PhD-FSTM-2022-029
The Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 22/03/2022 in Esch-sur-Alzette

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Najmeh SOROUSH

VERIFIABLE, SECURE AND PRIVACY-PRESERVING COMPUTATION

Dissertation defence committee

Dr Peter Y.A. Ryan, dissertation supervisor
Professor, Université du Luxembourg

Dr Ivan Visconti,
Professor, Université du Salerno, Italy

Dr Olivier Pereira,
Professor, Université Catholique de Louvain

Dr Gabriele Lenzini, Chairman
Professor, Université du Luxembourg

Dr Peter B. Rønne,Vice Chairman
Polish Academy of Sciences

Abstract

In this thesis, I present the research I conducted with my co-authors on numerous areas of verifiable, secure, and privacy-preserving computation during my doctoral studies at the University of Luxembourg, where Professor Peter Ryan advised me.

In the first part, I study the functional encryption scheme. In the standard setting of functional encryption, it is assumed both the Central Authority (CA) and the encryptors to run their respective algorithms faithfully. However, in the case of dishonest parties, the security of the cryptosystem may be violated. It means that dishonest parties can cause inconsistent results which may not be detected. In the first part, we improve on this situation by considering Inner-Product Encryption (IPE), a special case of functional encryption and a primitive that has attracted wide interest from practitioners and researchers in the last decade. Specifically, we construct the first efficient verifiable Inner Product Encryption (VIPE) scheme according to the inner-product functionality. As the next step, we construct a verifiable IPE that satisfies unconditional verifiability, whereas privacy relies on the standard assumption.

The second part of this thesis presents my research on e-voting protocols. I revisit the coercion-resistant e-voting protocol by Juels, Catalano and Jakobsson (JCJ) and, particularly, the attempts to make it usable and practical. In JCJ the user needs to handle cryptographic credentials and fake these in case of coercion. We present a hardware-independent protocol that can be implemented using a combination of a digitally stored cryptographic length key and a PIN only known by the voter. The long credential could be stored in several places or hidden via steganography. At the ballot casting phase, the software will input the digital key and the password to form the credential submitted with the vote. Depending on the level of coercion, the coerced voter can either fake the long credential or, for stronger levels of coercion, the voter can reveal the digitally stored credential to the coercer but fake the PIN. Due to our improved tally, the coercer will not know if he got faked credentials or PINs. On the other hand, since the voter memories the PIN is a high chance of users making a PIN typo error which will invalidate the vote and remain undetected. Note that naively giving feedback on the correctness of the PIN is not possible for coercion-resistance as it would allow the coercer to check whether he got a fake PIN or not. Instead, we will define a set of allowed PIN errors (e.g., chosen by the election administrator). We will consider a ballot valid if it has a correct PIN or an allowed PIN error but invalid for other PINs. At the tally phase, we construct protocols that secretly check whether a given PIN is in the set of allowed PINs and will sort out invalid ballots.

We also design another End-to-End verifiable e-voting scheme achieving coercion-resistance via deniable vote updating. We propose a new e-voting system that enables voters with an intuitive mechanism to update their possibly coerced vote in a deniable way. What is more, our e-voting system does not introduce any additional trust assumptions for end-to-end verifiability and vote privacy besides the standards. Moreover, we demonstrate that our e-voting system can be instantiated efficiently for practical elections. With these properties, our e-voting system has the potential to close the gap between theory and practice in coercion-resistant e-voting.

Acknowledgements

Without the help and support of many people, this research would not have been accomplished. At the end of my journey as a PhD student, I try to express my gratitude to everyone who has supported me academically and otherwise.

First and foremost, I would like to thank my supervisor, ***Peter Y. A. Ryan***, for accepting me as his student, giving me the opportunity to work for the APSIA group and, his advice and guidance during my studies. Most importantly, I would like to express my gratitude to him for his consistent support throughout the past four years.

I would like to thank ***Peter B. Roenne*** for patiently supervising me during the last years of my PhD. When it came to my ideas and details, he always pushed me to be as detailed as possible. Without this encouragement, my work would have been less precise in terms of scientific research. I admire his attention to all the detail and thoroughness, and I appreciate his patience in dealing with my mistakes.

My first steps into cryptography and research were guided by ***Vincenzo Iovino***. I would also thank Vincenzo for supervising me during the first two years of my PhD. I would like to thank him for sharing his idea with me and putting his trust in me to work on it.

I thank ***Dimiter Ostrev***. Working with him and his research attitude made me realize that I have to organize my mind and my idea when it come to the research. Dimiter is someone who is always a pleasure to work with.

I would also like to thank ***Alfredo Rial*** for patiently reading my long messy proof and guiding me through my first publication.

I am grateful to ***Johannes Müller*** for all the support, advice and worthwhile discussions while working together. I especially appreciate his early remarks and suggestions on my thesis.

I would also like to thank my jury members, ***Gabriele Lenzini***, ***Ivan Visconti***, ***Olivier Pereira*** and ***Peter Roenne*** for agreeing to evaluate my thesis. I thank both Ivan and Peter for our yearly CET meetings and their constructive feedbacks.

Research endeavours rarely happen without collaboration I would like to thank my collaborators and co-authors. ***Fatima El Orche***, ***Ehsan Estaji***, ***Kristian Gjøsteen***, ***Thomas Haines***, ***Vincenzo Iovino***, ***Johannes Muller***, ***Dimiter Ostrev***, ***Balazs Pejo***, ***Ivan Pryvalov***, ***Alfredo Rial***, ***Peter B. Roenne***, ***Peter Y. A. Ryan*** and ***Philip B. Stark***.

I would thank ***Naira Bardasaryan***, ***Jessica Giro***, ***Natalie Kirf***, ***Magali Martin*** and ***Catherine Violet*** for their administrative support.

I would like to thank ***Ehsan*** for being a good colleague and friend. Apart from collaboration, his company accelerated my learning process when I began exploring e-voting.

I thank ***Afonso***, ***Alireza***, ***Georgos***, ***Giuseppe***, ***Ivan***, ***Marjan***, ***Rafieh***, and ***Yan*** for their kindly help and support while I was writing my thesis, proofreading this manuscript and finding a much too high number of extra-camas!

I thank my office mate ***Jeroen*** for enduring my plants and messiness, it truly helped me feel comfortable in our office.

Beyond our work together, I also thank the APSIA group for fun and interesting conversations around tea, coffee or a Kwak beer: ***Aditya***, ***Afonso***, ***David***, ***Ehsan***, ***Ehsan***, ***Fatima***, ***Georgios***, ***Hao***, ***Ivan***, ***Jeroen***, ***Johannes***, ***Marie-Laure***, ***Marjan***, ***Masoud***, ***Petra***, ***Rafieh***, ***Wojciech*** and ***Yan***.

I would like to thank my family and dear friends in ***Khooneye Sohharabina*** and ***Leila*** who helped me start with my beloved journey. Without their support and encouragement, I could not have started my path.

And above all, I would like to express my deep gratitude to ***Shohreh*** and ***Zohreh***, for making me feel at home here.

Finally, I would like to gratefully acknowledge the generous funding of my Ph.D. by the Fonds National de la Recherche (FNR, Luxembourg National Research Fund) via the University of Luxembourg.

To the goldfish in Belval pond,

my companions, while my mind was wandering!

To M . M,

for inspiration!

Contents

Chapter 1

Introduction

“ Once Upon a Time,
there was a mystery”

Our story goes back far away, thousands of years ago, ancient Greece, ancient Rome, to Julius Caesar,

> *“If he had anything confidential to say, he wrote it in cipher, that is, by so changing the order of the letters of the alphabet, that not a word could be made out. If anyone wishes to decipher these, and get at their meaning, he must substitute the fourth letter of the alphabet, namely D, for A, and so with the others.”* [1]

And although he sadly was assassinated in Senate House of Pompey in Ides of March of 44 BC, the cryptographic adventure has been going on. That, after all, was only the beginning!

Contents

For a long time, the primary goal of cryptography, classified into classical and modern cryptography, was executing secure communications over an insecure channel between a sender and receiver. Furthermore, to establish a secure connection, the parties must agree on a secret key in advance or rely on a third entity for key distribution.

From a classical standpoint, cryptography is viewed as an art rather than a science. Back then, because of the lack of precise definitions, mathematical methods, and conceptual foundations, developing practical cryptosystems required creativity. Two well-known examples of classical cryptography are Caesar's cipher and the Enigma machine.

One of the oldest cryptography methods goes back to *Julius Caesar*. Coming from ancient history books[1], Julius Caesar invented an encryption method by shifting the letters of the alphabet three places forward: a became D, z became C, and so on. The receiver with the knowledge of the secret value, 3, easily replaced each letter with a letter three positions backwards to recover the original message.

Another well-known example is the *Enigma Machine*, a cipher device that Nazi Germany widely used during WWII. Despite the Germans' belief that the Enigma machine was so secure that it could transmit the most sensitive information, it was broken in 1932 by a team of mathematicians and cryptanalysts at the Polish Cipher Bureau. Another successful attempt to break more recent versions occurred at *Bletchley Park* in the United Kingdom. To decipher Enigma, the Bletchley team, led by *Alan Turing*, designed a decryption electro-mechanical device, ***Bombe***, that can be considered, among other electro-mechanical encrypting devices, the earliest prototype of what would become the modern computer we know today.

The history of breaking the cryptosystems that had been known as robust and secure schemes led to the development of cryptography from an art into a science, namely ***"Modern Cryptography."***

1.1 Modern Cryptography

The fundamental paper by Claude E. Shannon, *"Communication Theory and Secrecy Systems"* [237] published in 1949, is widely regarded as the foundation of modern cryptography. Shannon proposed a formal mathematical definition for perfect secrecy, also known as "Information-theoretic secrecy", which asserts that no one can gain knowledge from the ciphertext. Shannon also proved the existence of a perfectly secure cryptosystem by showing that the One-Time Pad Encryption Scheme, introduced by GS Vernam [252] in 1926, has perfect secrecy.

However, his work put an end to the possibility of designing a perfectly secure cryptosystem in the real world. He proved that in any cryptosystem, the secret key must be as long as the message itself to generate a ciphertext that properly hides all information about the message; moreover, the key cannot be used more than once.

Following that, a novel definition, *"Computational Secrecy,"* was presented to bypass the inherent limitation of a perfect secrecy notion. In contrast to the information-theoretic concept of secrecy, the definition of computational security incorporates two modifications: first, considering effective adversaries that run for some feasible amount of time and, secondly, allowing the adversary to break the scheme with negligible (very small) probability. Put simply; a cryptosystem is computationally secure if an efficient adversary cannot gain any knowledge from the ciphertext in a reasonable amount of time except for negligible probability.

[1]Suetonius, Life of Julius Caesar

As modern cryptography requires, the terms *"efficiency"* and *"small probability"* have been precisely defined with the help of mathematics and complexity theory concepts. In simple words, an efficient algorithm takes some value with length ℓ as input, and it terminates in a time and space polynomial in ℓ. A value is negligible in terms of ℓ when it is smaller than the absolute value of any inverse polynomial in ℓ. In addition to explicit terms such as efficiency, the definition of computational secrecy also indicates some subtle concepts. For example: *what does it means for an adversary to break a secure protocol? Recover the private key or recover only the original message from the ciphertext?* These notions also require precise and consistent definitions.

In the early years of modern cryptography, encryption schemes continued to have the same classical framework and syntax. Namely, most secure protocols were designed based on a secret key shared between the sender, Alice, and the receiver, Bob, and they guarantee the following three main purposes. They provide:

However, secure cryptosystems based on the shared secret key has their shortcomings, including challenges arising from secure key-distribution solutions and, in the following step, storing and managing a large number of keys. These concerns became more severe as technology advanced, particularly in telecommunications and computer science, which encouraged cryptographers to launch a new line of research into developing practical and secure key-exchange protocols. This led to the ground-breaking results: discovery of public-key encryption schemes.

1.1.1 The Dawn of the Public Key Encryption Scheme

"***We stand today on the brink of a revolution in cryptography.***"
1976, Diffie-Hellman.

Whitfield Diffie and Martin Hellman's pioneering paper [94] "New Directions in Cryptography", published in 1976, demonstrated an elegant approach to construct a shared key over an insecure channel, assuming the hardness of a mathematical problem. In brief, suppose that in the cyclic group $\mathbb{G}$ with the generator g, it is easy to compute g to power x but infeasible to compute the discrete logarithm of h ($x = \log_g h : g^x = h$). Then Alice and Bob can establish a secret value by performing steps in Figure 1.1.

This outstanding achievement paved the path for cryptographic primitives such as *Public Key Encryption Scheme*, (PKE for short) and *Digital Signatures Scheme.* In PKE schemes, each entity has two keys. The publicly-known key, a.k.a. the public key, enables individuals to encrypt their message m under the public key. And the private key, which allows the owner, to decrypt the ciphertext.and recover the original message. In contrast, in the signature scheme, the owner of the secret key signs some messages, m and later, anyone can validate the message's using the related public key.

1.2 Towards Advanced Cryptography

Despite the wide variety of cryptographic primitives, they all have adopted a common subtle concept; ensuring some property, either confidentiality, authenticity, or integrity on ***data***. Hence, in the next step, natural questions may be:

FIGURE 1.1: Diffie-Hellman Key Exchange Protocol

1. Alice and Bob select two random integers:

$$\text{Alice} : n_A\,,\ \text{Bob} : n_B.$$

2. They compute $\mathcal{A} = g^{n_A}$ and $\mathcal{B} = g^{n_B}$, sends it the other side while keeping n_A and n_B secret:

$$\begin{tikzcd}\text{Alice} \arrow[r, shift left, "\mathcal{A}"] & \text{Bob} \arrow[l, shift left, "\mathcal{B}"]\end{tikzcd}$$

3. They raise the other party's value to their secret number after receiving it:

$$\text{Alice} : \mathcal{K} = \mathcal{B}^{n_A}\,,\ \text{Bob} : \mathcal{K} = \mathcal{A}^{n_B}$$

4. The shared key would be identical to $\mathcal{K} = \mathcal{B}^{n_A} = \mathcal{A}^{n_B} = g^{n_A \cdot n_B}$, and the hardness of Diffie-Hellman problem guarantees that the shared key is secure.

i. *Is it possible to find a cryptosystem that ensures confidentiality, integrity and authenticity of some computations?*

ii. *Is it possible to evaluate a function on encrypted data without compromising the plain data?*

A more concrete example is an e-voting system in which voters cast an encrypted ballot rather than a plain ballot.

i. *How do we verify whether the outcome of an e-voting method is accurate, or some malicious authority manipulates some ballots?*

ii. *How can we tally encrypted ballots without having to open every single one in a verifiable way?*

In their seminal paper[128] in the late 1980s Shafi Goldwasser, Silvio Macali and Charles Rackoff positively answered the first questions by introducing one of the fascinating cryptographic primitives known as ***Zero-Knowledge Proof Systems***.

Furthermore, regarding the second question, the functional encryption scheme was developed independently in two papers [55, 214] by Dan Boneh, Amit Sahai, Bernet Water and Adam O'Neill. These two primitives constitute the foundation of our research into developing a ***Verifiable Secure and Privacy-Preserving Computation***, which is the primary topic of this thesis.

1.2.1 Zero-Knowledge Proof Systems

Zero-knowledge proof systems, introduced in the seminal work of Goldwasser, Micali, and Rackoff [128], are one of the fascinating and essential primitives in cryptography. However there is a contradiction hidden within the concept of Zero-Knowledge. At the same time, proof should be convincing; it must yield no knowledge beyond the validity of the proven statement. In other words, obtaining a zero-knowledge proof that a statement is true is equivalent to being told by a trusted party that the statement is true.

Goldreich, Micali, and Wigderson [123] state that the zero-knowledge proof is an innovative technique to force involved parties in a protocol to adhere to it while assuring that no secret information is leaked.

Zero-knowledge proof is most commonly used to prove NP-language membership; Namely, the prover, with the help of some auxiliary secret input w, which is called witness, proves that some x belongs to some language $\mathscr{L}$. Informally, the proof must meet three requirements:

i. ***Completeness:*** the verifier accepts the proof if x does belong to the language $\mathscr{L}$,

ii. ***Soundness:*** no [cheating] prover can falsely convince the verifier of accepting the proof for an invalid x. (*i.e.,* When x does not belong to the Language $\mathscr{L}$),

iii. ***Zero-Knowledge:*** the proof (including all interactions) should not reveal anything rather than the the statement's validity.

While formalizing the first two properties is straightforward, defining a formula that encapsulates the third is more challenging. Zero-Knowledge is strongly intertwined with the simulation paradigm, a fundamental concept in cryptography that underpins a variety of cryptographic security concepts, including semantic security.

Simply put, simulation is a method of creating an ideal world in which we have the security property by definition and then comparing it to the real world in which we want to prove the security of some primitive. For example, the concept of a zero-knowledge property compares what an adversary who receives the proof can learn to what an adversary who receives only *"the statement is valid"* . According to the definition, a proof system has the zero-knowledge property if both adversaries can learn approximately the same amount of information. This ensures a high level of security because the latter adversary learns nothing other than the statement's validity. This implies that when the adversary receives the proof in the real world, it learns nothing more than the statement's validity, which we must prove using the zero-knowledge proof system.

The simulator must extract the adversary's execution inputs and then generate a view consistent with these inputs. Furthermore, this view must be indistinguishable from the actual view. Section 2.4.1 will discuss the simulation paradigm in modern cryptography and compare it with game-based techniques.

As Zero-Knowledge Proof System is one of the essential ingredients we employ towards having ***"Verifiable Secure and Privacy-Preserving Computation,"*** we gather basic terminology and background of it in a brief survey in chapter 3.

1.2.2 Fine Grained Access to Information

Subsequent breakthroughs in modern cryptography stemmed from significant advances in machine learning, cloud computing, decentralizing and outsourcing computing initiatives. The issue that the areas mentioned above have in common is:

How to find a way to use the data while preserving individual privacy?

The above question put forth the idea of ***"fine-grained access"*** to the data rather than ***"all-or-nothing"***. We illustrate the notion with a concrete example.
Consider a central bank that audits and ranks local banks based on transaction quantity. As a result, all local banks are willing to share their financial turn-over (which is actually confidential information on their customer transactions) with the central bank

to facilitate financial discoveries such as predicting the future or finding better marketing methods without disclosing customers' private information due to confidentiality concerns.

If we encrypt the data traditionally (either with a public key or symmetric key cryptosystem), the central bank has access to all or none (all-or-nothing) of the data, depending on having or not having the key. It is important to note that the central bank does not require the raw data and simply needs to perform some computation on the raw data to obtain the necessary evaluation. As a result, an alternative and ideal approach for the central bank would be to have some restricted key that allows the holder to perform the computation through an encrypted database, learning just about the data's evaluation and nothing more about the raw data (fine-grained-access).

This is the concept underpinning the functional encryption scheme, the new encryption paradigm first presented by Sahai and Waters [229], formalized by Boneh, Sahai and Waters [55] and independently by O'Neill [214].

In a functional encryption scheme, in contrast to the public key cryptosystem, a decryption key, so-called, token, tok_f allows obtaining a function of the original message, $f(m)$ rather than the message itself. More specifically, in a functional encryption scheme for functionality $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, defined over *key space* $\mathcal{K}$, *message space* $\mathcal{X}$, and *output space* $\mathcal{Y}$, for each key $\in \mathcal{K}$, the owner of the key can obtain $F(k, x) = f_k(x)$ from the ciphertext of m computed under the master public key MPK, Figure 1.2.

FIGURE 1.2: Functional Encryption Scheme

• Master Public Key: MPC,	• $\mathsf{Enc}(\mathsf{MPK}, m) \to \mathsf{CT}$
• Master secret key: MSK,	• $\mathsf{TokGen}(\mathsf{MSK}) \to \mathsf{tok}_f$
• Functionality: $\mathcal{F} : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{Y}$	• $(\mathsf{CT}, \mathsf{tok}_f) \to f(m)$

1.2.3 Inner Product Encryption Scheme

Since the beginning, one of the most critical questions about functional encryption schemes has been,

> *For which functionality can we design an efficient, functional encryption system that provides a high level of security?*

Researchers have attempted to expand the functionality class supported by a functional encryption scheme despite some impossibility results. The path to developing a functional encryption scheme with the functionality of all polynomial-time algorithms, began with an identity-based encryption scheme and has a steppingstone, namely *"Inner Product Encryption Scheme"*.

Inner product encryption scheme (IPE for short) is a notable special case of functional encryption schemes and have attracted wide interest from both practitioners and researchers in the last decade, including privacy-preserving statistical analysis, where statistical analysis itself includes sensitive information and conjunctive/disjunctive normal form formulas.

In IPE, the message is associated with a pair $(m, \vec{x})$, with m being the *payload message* and $\vec{x} \in \mathbb{Z}_p^n$ the *attribute*, the token is associated with a vector $\vec{v} \in \mathbb{Z}_p^n$ and the functionality is defined as:

$$F(\vec{v},(m,\vec{x})) = f_{\vec{v}}(\vec{x},m) = \begin{cases} m \text{ if } \langle\vec{x},\vec{v}\rangle = 0 \mod \mathbb{Z}_p \\ \perp if \langle\vec{x},\vec{v}\rangle \neq 0 \mod \mathbb{Z}_p \end{cases}$$

In the first part of this thesis we present our *efficient* and *perfectly correct* inner product encryption scheme whose security is based on the standard assumption.

1.3 Verifiability in the Context of Functional Encryption

In a standard functional encryption scheme, it is implicitly assumed that the encryptors and the Central Authority (the owner of the master keys) that generate the tokens are trustworthy. Indeed, in the presence of any dishonest party (either the party that generates the token or the party that encrypts the message) the decryption outputs may be inconsistent, and this raises severe issues in practical applications.
In the preceding example, if a local bank delivers inaccurate encrypted data or a faulty token, resulting in incorrect evaluation. Even further, no other party will be able to identify this malicious operation due to the system's security *i.e.,* the inherent security will protect the harmful party.

As a concrete example, let's consider a case in which a local bank sends a list of encrypted data containing the amount of withdrawal from the bank account and deposit to the account for each of its customers, and it is a fact that the amount of deposit is always a greater number than the withdraw. This encrypted data is sent to two separate head offices, which evaluate the bank balance for distinct time periods. Furthermore, the local bank generates two tokens, $\mathsf{tok}_F, \mathsf{tok}_G$, for two functions, F and G. Function F evaluates the local bank's balance for months 3 and 4 and function G evaluates the balance for months $3, 4$ and 5.

If the local bank is not trustworthy, it can generate a faulty ciphertext or a defective token for at least one of these two centers, resulting in:

	1	2	3	4	5	6	7	8	9	10	11	12
WL: deposit for each month	56	34	91	62	10	73	55	91	102	40	78	109
DL: withdraw for each month	12	5	18	23	5	21	32	7	11	23	55	106

$$(\mathsf{MPK}, WL, DL) \mapsto \mathsf{CT}: \ (\mathsf{tok}_F, \mathsf{CT}) \mapsto 112, \ (\mathsf{tok}_G, \mathsf{CT}) \mapsto 94$$

As the numbers show, the two numbers are incompatible because the second number, which includes a month more, should be higher than the first. Unless these two parties compare their results, they will be unaware of the system's flaw.

To overcome this issue, Bardishayan [23] introduced the concept of verifiable functional encryption, which captures the basic requirement of output correctness. That is, even if the ciphertext is faulty generated (as well as the master keys and the token), the decryptor is guaranteed a meaningful sense of accuracy. In simple terms, the principle of verifiability in functional encryption schemes asserts that if the master public key MPK and a ciphertext CT pass a public verification test, it indicates that there exists some message m such that the following holds for all verified tokens (for function f):

$$f\colon (\mathsf{tok}_f, \mathsf{CT}) \mapsto f(m).$$

Indeed, verifiability and security (i.e., privacy) challenge each other. And developing a solution that satisfies both of these qualities consistently introduces the concept of verifiable functional encryption, a fascinating challenge for cryptographers.

In the first part of our thesis, we develop a verifiable inner product encryption scheme, employing our perfectly correct IPE along with a particular Zero-Knowledge proof system.

1.4 Verifiable Secure E-Voting Protocols

E-voting systems have been used for political elections in many countries, including Australia, Brazil, India, Estonia, and the United States. In such high-stakes elections, it is also critical to prevent voters from being coerced to vote or not vote for a specific candidate, abstain from voting or sell their votes. This is also reflected in the international standards of elections of the *UN Committee on Human Rights* [248] which requires that "[any] coercion of voters should be prohibited by penal laws, and those laws should be strictly enforced."

It is believed that developing an electronic voting system is a challenging and complicated task. Not only because of the vast and diverse aspect of an election (social aspect, networks, and complexity of the implementation) but also because of the multiplicity and diversity of the involved parties (a wide range from experts in computer science and security to people with no experience or knowledge in encryption or even computer science.)

On the one hand, a voting system must guarantee voter *privacy* while also protecting them from malicious influencers who want to swing an election or intrude on the election by coercing individuals to vote in a particular way (coercion-resistance). On the other hand, it must safeguard voters from themselves; precisely, it should not place voters in a situation where they can prove how they voted since this would expose them to coercion and vote-buying, *"receipt-freeness"*. Additionally, electronic voting must be transparent or verifiable *"E2E-Verifiability"*. At the end of the election, it must be demonstrated that all stages were followed the predetermined procedure honestly and that the stated result is the same result acquired from the counting of valid ballots. Furthermore, all of these stages must be implemented *practically*, *efficiently*, and *useable* for the general public, who may not have substantial experience in computers, security, or related areas.

There is a consensus in modern e-voting that all desired e-voting protocols must be; usable, efficient, secure (privacy-preserving) and verifiable. The first two properties convey the general concept:

i. ***Usability*** captures the fact that human voters must easily use the system to cast their votes, check that their ballots were counted, and run an anti-coercion strategy if necessary. Otherwise, even if the e-voting system is theoretically secure, it will be rendered completely insecure in practice.

ii. ***Efficiency*** expresses that for real elections, an e-voting system must be practically *efficient*; in particular, the voters' ballots should be small and be generated quickly, and it should be possible to announce the election result fast.

When it comes to verifiability and privacy in the context of e-voting protocols, each term takes on its meaning based on distinct but intertwined core properties. For example, privacy is defined by the following key properties: *ballot secrecy*, *receipt freeness*, and *coercion resistance*, and the threefold *vote-as-intended*, *recorded-as-cast*, and *tallied-as-recorded* define the concept of E2E verifiability.

1.4.1 Privacy in the context of E-Voting

Informally we can define the privacy notion in an e-voting scheme into the three key properties [220]:

i. ***Ballot secrecy:*** The voting mechanism must be designed to not expose the voter's choice, up to the election result.

ii. ***Receipt freeness:*** The voting system should not prove how voters voted to a third party.

iii. ***Coercion resistance:*** The voter should be able to vote for her preferred candidate even if she appears to be cooperating with a coercer.

Given that the threat of punishment does not always deter individuals from engaging in illegal activities during an election, the voting system must be designed in such a way that it supports the fundamental privacy requirements on a technical level or at the very least mitigates the possibility of conducting such attacks (see, e.g., [168, 80, 17, 79, 97, 232, 21, 259, 180, 154]).

For instance, we may ensure some degree of ballot privacy by utilizing a secure encryption scheme. Or we can protect voters against the coercer by offering a *counter-strategy* that allows voters to achieve their goal in the presence of an adversary instead of obeying the coercer. The coerced voter can vote for her preferred candidate by running the counter-strategy. However, due to some technical mechanisms, the coercer should not be able to distinguish whether the coerced voter followed his instructions (e.g., voted for the coercer's favourite candidate) or executed the counter-strategy.

The first two features, ballot secrecy and receipt freeness are comparable to the in-person election. at the same time the coercion-resistance property in electronic remote voting system is slightly different from the ordinary election. So, we provide an overview of counter-strategies that support a coercion-resistance voting protocol.

From a technical perspective, several approaches in the literature implement the concept of coercion resistance *fake credentials*, *masking*, and *deniable vote updating*. The three most commonly used approaches are:

i. ***Fake credentials*** technique provides each voter with a unique and secret credential C. A voter uses C to submit her vote when she is not under coercion. Otherwise, if a voter is under coercion, she can create a so-called *fake credential* C^* to submit her coerced vote. Since the voter's fake credential is invalid, the voting authorities will secretly remove the respective vote.

ii. ***Masking choices*** provide each voter with a unique and secret mask $\hat{m}$. A voter uses $\hat{m}$ to blind their actual vote $\hat{v}$ when not coerced. Otherwise, if a voter is coerced to vote for a different choice v, then it computes a false mask m such that the resulting blinded vote is still a vote for its real choice $\hat{v}$.

iii. ***Deniable vote updating*** scheme permits the voter to overwrite her submitted ballot, which she may have cast under coercion, such that no one else, including a possible coercer, can see whether or not the voter has subsequently updated her vote. The voting system should not provide evidence on how a voter voted to a third party.

1.4.2 Verifiability in the Context of E-Voting

According to documented cases, many e-voting systems include weaknesses that allow for election results to be tampered with (see, e.g., [95, 244, 261, 147, 243]). To overcome these vulnerabilities, a comprehensive investigation has been conducted into developing an electronic voting system with *end-to-end verifiability* security properties. End-to-end verifiability features imply that if the election protocol passes a particular procedure, the published election result is correct, i.e., corresponds to the votes cast by the voters, even if voting devices and servers have programming flaws or are malicious.

It is important to note that, as with other security protocols, there is a trade-off between verifiability and privacy. However, much successful research has been conducted on the verifiability of secure voting protocols, whether for postal or electronic elections.

As early instances of verifiable voting schemes, we can refer to Punchscan and Optical Scan [76], introduced by David Chaum which employ visual cryptography and optical scanning to enable voters to verify their votes' accuracy and tally result.

A scratch-off surface is used in Ben Adida's Scratch and Vote technology. Another technique was used in the Prêt à Voter system, invented by Peter Ryan *et al.* [226]. Instead, each ballot form is a random permutation of the primary candidate list. This permutation is encrypted to provide voter privacy, and in the tally phase, it is used to retrieve the original permutation and also serves as the voter verification receipt. Ron Rivest developed the ThreeBallot voting protocol to give some of the benefits of a cryptographic voting system without the use of cryptography. Numerous modern verifiable voting methods are theoretically compatible with paper-based and electronic elections. Additional examples are available in [8, 78, 39].

Although the preceding examples demonstrate robust and reliable techniques for ensuring verifiability, most verifiable electronic voting protocols developed in the last few years have relied on a zero-knowledge proof systems, which is also our approach for implementing a secure and verifiable electronic voting protocol.

1.5 Contributions and Outline of Thesis

We now describe the structure of this thesis and summarize the main contributions. The contributions of this thesis are twofold. The first part, which mainly relies on our results in [242], is concerned primarily with verifiability in functional encryption.

The first part of my PhD research resulted in constructing an efficient verifiable inner product encryption scheme from bilinear maps. Towards this goal, we build a perfectly correct IPE scheme that may be of independent interest. To our knowledge, all IPE schemes known in literature do *not* satisfy perfect correctness, while our perfectly correct IPE scheme is based on standard assumptions over bilinear groups.

The second part of my PhD research is devoted to verifiable, secure voting systems. My result contains two practical and secure electronic voting techniques that tolerate human error while maintaining a high level of privacy and resistance to coercion.

We expand below on the three contributions developed in this thesis and outline some of their implications. The results discussed below have been mainly taken from papers [242, 98, 227] presented respectively at PKC-2020, E-Vote-ID-2021 and E-Vote-ID-2021. Further our contribution in chapter 8 is under submission.

1. Najmeh Soroush, Vincenzo Iovino, Alfredo Rial, Peter B. Rønne and Peter Y. A. Ryan. "Verifiable Inner Product Encryption Scheme" In: *Public-Key Cryptography*

- PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part I. ed. by Aggelos Kiayias et al. Vol. 12110. Lecture Notes in Computer Science. Springer, 2020, pp. 65–94. DOI `10.1007/978-3-030-45374-9_3`. URL: `https://doi.org/10.1007/978-3-030-45374-9_3`

2. Ehsan Estaji, Thomas Haines, Kristian Gjøsteen, Peter B. Rønne, Peter Y. A. Ryan and Najmeh Soroush. "Revisiting Practical and Usable Coercion-Resistant Remote E-Voting." In: *Electronic Voting - 5th International Joint Conference, E-Vote-ID 2020, Bregenz, Austria, October 6-9, 2020, Proceedings.* Ed. by Robert Krimmer, Melanie Volkamer, Bernhard Beckert, Ralf Küsters, Oksana Kulyk, David Duenas-Cid and Mikhel Solvak. Vol. 12455. Lecture Notes in Computer Science, Springer, 2020, pp. 50–66. DOI: `10.1007/978-3-030-60347-2_4`

3. Peter Y. A. Ryan, Peter B. Roenne, Dimiter Ostrev, Najmeh Soroush, Fatima-Ezzahra El Orche and Philip B. Stark. "Who Was that Masked Voter? The Tally Won't Tell!" In: *Electronic Voting - 6th International Joint Conference, E-Vote-ID 2021, Virtual Event, October 5-8, 2021, Proceedings.* Ed. by Robert Krimmer, Melanie Volkamer, David Duenas-Cid, Oksana Kulyk, Peter B. Rønne, Mihkel Solvak and Micha Germann. Vol. 12900. Lecture Notes in Computer Science. Springer, 2021, pp. 106–123. DOI: `10.1007/978-3-030-86942-7_8`. URL: `https://doi.org/10.1007/978-3-030-86942-7_8`

1.5.1 Perfect Inner Product Encryption Scheme

The first part, largely based on our results in [242], primarily focuses on developing a perfectly correct IPE scheme based on standard assumptions over bilinear groups.

We need to build an IPE scheme with perfect correctness to instantiate the verifiable inner product encryption scheme. Our starting point to construct a perfectly correct IPE scheme is the IPE scheme of Park [217], which only enjoys statistical correctness. The reason for choosing this IPE is that it is conceptually simple, and its security is based on standard assumptions over bilinear groups. However, to make Park's scheme compatible with the Badrinarayanan *et al.*'s transformation we need to solve several technical challenges, in particular:

i. The master public key needs to be verifiable.

ii. The scheme has to satisfy perfect correctness.

This requires substantial modification of all main algorithms: setup, token generation, encryption, and decryption.

1.5.1.1 Verification of Algorithm's Outputs.

A VIPE scheme requires public verification algorithms that can verify the outputs of the setup, encryption, and token generation algorithms, in particular check whether these algorithms were run honestly. In more detail, if any string (master public key, ciphertext or token) passes the corresponding verification algorithm, it means it was a proper output of the corresponding algorithm (setup, encryption, or token generation). Each party who runs the setup, encryption or token generation algorithm needs to provide a proof that it executed the algorithm honestly without revealing harmful information about the secret parameters or the randomness used in the algorithm.

Usually, non-interactive Zero-Knowledge (NIZK) proofs are used in this context. However, unfortunately, NIZK proofs cannot be used for verifiable FE as they rely on a trusted CRS (Common Reference String) or random oracles. So instead, we aim at *perfect verifiability* that holds despite any collusion and computing power. The transform of Badrinarayanan *et al.* cleverly solves the issue by employing non-interactive witness-indisting uishable proof systems (NIWI for short).

Following the transform of [23], our VIPE consists of four instances of an IPE scheme. In the VIPE's encryption algorithm we first run the IPE's encryption algorithm four times to generate four ciphertexts and then we prove that all these four ciphertexts are the encryption of the same message or that some other trapdoor predicate is satisfied (the latter is needed for message indistinguishability and will be detailed later).

For the sake of argument, let us assume the VIPE scheme consists only of two (instead of four) parallel perfectly correct IPE scheme instantiations IP and $\hat{\mathsf{IP}}$. The master public key of the Park's scheme [217] contains a component $\Lambda = \mathbf{e}(g, g')$ in which g is public but g' needs to be kept secret. An honestly computed ciphertext CT in IP includes $\mathsf{ct}_1 = g^{-s}$ and $\mathsf{ct}_7 = \Lambda^{-s} \cdot m$ among its components (we here ignore the other components). We first provide proof that CT (resp. $\hat{\mathsf{CT}}$ in $\hat{\mathsf{IP}}$) is well-formed. Then we need to prove that the two ciphertexts are both encryptions of the same message M (i.e., $m = \hat{m} = M$). We reduce the problem to proving that the following property holds, (See 4.5 for more detail):

$$\frac{\mathsf{ct}_7}{\hat{\mathsf{ct}}_7} = \frac{\mathbf{e}(g, g')^{-s} \cdot m}{\mathbf{e}(\hat{g}, \hat{g}')^{-\hat{s}} \cdot \hat{m}} = \frac{\mathbf{e}(\hat{\mathsf{ct}}_1, \hat{g}')}{\mathbf{e}(\mathsf{ct}_1, g')} = \frac{\mathbf{e}(\hat{g}^{\hat{s}}, \hat{g}')}{\mathbf{e}(g^s, g')}$$

However, since g' and $\hat{g}'$ are not public, the party who runs the encryption algorithm would be unable to prove this property. We solve this issue in the following way: We add to the master public key of IP two elements g_1, g_2 (and $\hat{g}_1, \hat{g}_2$ for $\hat{\mathsf{IP}}$) satisfying

$$\Lambda = \mathbf{e}(g, g') = \mathbf{e}(g_1, g_2)\,,\ \hat{\Lambda} = \mathbf{e}(\hat{g}, \hat{g}') = \mathbf{e}(\hat{g}_1, \hat{g}_2).$$

Then, we define the new secret variables $\mathcal{X}_3 = g_1^s, \hat{\mathcal{X}}_3 = \hat{g}_1^{\hat{s}}$ and add the following equations:

$$\mathsf{ct}_7^{-1} \cdot \hat{\mathsf{ct}}_7 = \mathbf{e}(\mathcal{X}_3, g_2) \cdot \mathbf{e}(\hat{\mathcal{X}}_3, \hat{g}_2)^{-1}, \quad \mathbf{e}(g, \mathcal{X}_3) = e(\mathsf{ct}_1, g_1), \quad \mathbf{e}(\hat{g}, \hat{\mathcal{X}}_3) = \mathbf{e}(\hat{\mathsf{ct}}_1, \hat{g}_1).$$

It is easy to see that these equations are satisfied if and only if $m = \hat{m}$, which the encryptor can prove. Having modified Park's scheme, we thus have to prove that the modified scheme is indistinguishable-secure. This is done in Section 4.7 in which we reduce the Security of the scheme to the Decision Linear assumption.

1.5.1.2 On Perfect Correctness Property

The correctness property in IPE[2] captures the fact that the decryption's outcome is equal to the inner product of the two vectors. In contrast, the decryption algorithm outputs the payload message in predicate IPE schemes if only the inner product is zero. As a result, the decryption algorithm must determine whether to output the result of the calculation or $\perp$ symbol.

In the big picture, the encryption and decryption algorithms work as follows:

$$\mathsf{Enc}(\mathsf{MPK}, \overrightarrow{x}, m) \rightarrow \mathsf{CT},\ \mathsf{Dec}(\mathsf{Tok}_{\overrightarrow{v}}, \mathsf{CT}) \rightarrow m^* = m \odot (\mathsf{random})^{\langle \vec{x}, \vec{v} \rangle},$$

[2]We refer to the IPE functionality of Katz, Sahai and Waters [173].

in which random is some random value from the underlying group $\langle \mathbb{G}, \odot \rangle$ that depends on the randomness used by the token generator and encryption algorithms. Because in predicate-only-IP, there is no payload message m, the outcome of the decryption algorithm would be equal to

$$1 = \mathsf{random}^{\langle \vec{x}, \vec{v} \rangle} = \mathsf{random}^0$$

if two vectors are orthogonal and would be some random number if they are not orthogonal.

Technically, in the predicate-IPE case, the result of the computation in the decryption algorithm would be equal to the original message, m, if $\langle \vec{x}, \vec{v} \rangle = 0$, and a random message, m', if it is not. However, the primary issue here is how the decryption algorithm should decide whether to send m' or $\perp$ without knowing vectors $\vec{x}$ and $\vec{v}$.

Several solutions to this problem are provided in the literature for resolving this issue. For instance, the *Bloom Filters* method was used by Boneh and Water in [56]. In this approach, rather than using the entire set $\mathcal{M}$ as a message space, it considers a subset $\mathcal{M}'$ with the specific structure (for example, consider a subset of M whose binary representation starts with k zeros). Then in the decryption algorithm, at the end of the computing steps, if the result, say m^*, does not have that structure; the algorithm outputs the error; otherwise, it outputs m^* as the original message. However, in this method, the probability of false-positive, that is, $m^* \neq m$, but the decryption algorithm outputs it as the original message instead of $\perp$, is always greater than zero. Moreover, the probability of a false result depends on the size $\mathcal{M}$ and $\mathcal{M}'$. This indicates that we must either consider a small message space $\mathcal{M}'$ or accept a high probability of error, both of which might be unacceptable.

To our knowledge, all IPE schemes[2] known in the literature have a negligible probability of error which makes cheating possible and so not directly usable to construct verifiable functional encryption and functional commitments for the IPE functionality.

In more detail, in most pairing-based IPE schemes the encryption and decryption algorithms work as follows:

$$\mathsf{Dec}\big(\mathsf{Tok}_{\overrightarrow{v}}, \mathsf{CT} = \mathsf{Enc}(\overrightarrow{x}, m)\big) \longrightarrow m^* = m \cdot \mathbf{e}(g,h)^{(\lambda_1 s_1 + \lambda_2 s_2)\langle \vec{x}, \vec{v} \rangle}$$

in which λ_1, λ_2 are random values used in the token generation algorithm and s_1, s_2 are random values used in the encryption algorithm. To decide whether to accept the output of the decryption or not, the naive attempt would be the following. Generate two ciphertexts $\mathsf{ct}, \mathsf{ct}'$ with two independent random values $\{s_i\}, \{s'_i\}$, decrypt both ct and ct' to get M and m_2^* and if $m_1^* = m_2^*$ accept the result, or output $\perp$ otherwise. In more detail:

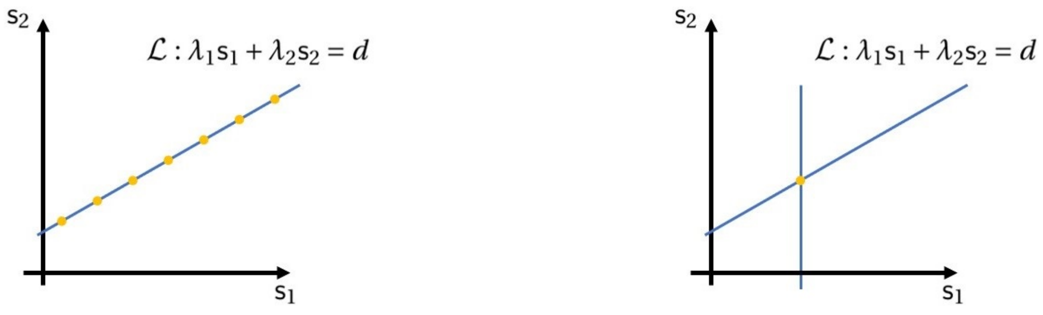
$$m_1^* = m \cdot \mathbf{e}(h,g)^{(\lambda_1 s_1 + s_2 \lambda_2)\langle \vec{x}, \vec{v} \rangle}, \; m_2^* = m \cdot \mathbf{e}(h,g)^{(\lambda_1 s'_1 + s'_2 \lambda_2)\langle \vec{x}, \vec{v} \rangle}$$

However, in case $\langle \vec{x}, \vec{v} \rangle \neq 0$ there is non-zero probability for which:

$$\lambda_1 s_1 + s_2 \lambda_2 = \lambda_1 s'_1 + \lambda_2 s'_2 \neq 0 \Rightarrow m_1^* = m_2^* \neq m$$

As shown in figure 1.3, every point on the line $\mathcal{L} : \lambda_1 x + \lambda_2 y = d$ makes the equation equal to zero, so even if we consider two ciphertexts, as long as the randomness of the encryption algorithm is chosen independently, an error is probable. To avoid this issue, we choose the random values so that the above equality can never occur (figure 1.3 the

FIGURE 1.3: The left diagram shows the points that result to incorrect output in decryption algorithm and the right diagram shows the single point that result to correct output in decryption algorithm


s_2
$\mathcal{L} : \lambda_1 s_1 + \lambda_2 s_2 = d$
s_1
s_2
$\mathcal{L} : \lambda_1 s_1 + \lambda_2 s_2 = d$
s_1

intersection of two lines). To do so, in the encryption algorithm we choose non-zero random values $s_1, \ldots, s_2$ and $s'_1, \ldots, s'_2$ such that $s_1 \neq s'_1$, and $s_2 = s'_2$. In this case, we have:

$$\begin{aligned} &\lambda_1 s_1 + s_2 \lambda_2 = \lambda_1 s'_1 + \lambda_2 s_2 \\ \Rightarrow &\lambda_1 (s_1 - s'_1) = 0 \\ \Rightarrow &(\lambda_1 = 0) \vee (s_1 = s'_1) \end{aligned}$$

Based on the way λ_1, s_1, s'_1 have been chosen, neither $(\lambda_1 = 0)$ nor $(s_1 = s'_1)$ may happen, hence the decryption algorithm outputs m if and only if $\langle \vec{x}, \vec{v} \rangle = 0$. The resulting IPE scheme satisfies perfect correctness as wished and we prove that it is still selectively indistinguishability-secure under the DLin Assumption 6. When constructing a VIPE scheme from such an IPE scheme, these additional constraints in the encryption and token generation procedures will correspond to more constraints in the proofs of correct encryption and token generation.

Furthermore, an additional challenge we will have to address is that some of the proofs in the Badrinarayanan*et al.* transform are for relations that consist of a generalized form of disjunction. Thus standard techniques to implement disjunctions for GS proofs cannot be directly applied.

1.5.1.3 Motivating Applications

IPE has numerous applications, including Anonymous Identity-Based Encryption [59], Hidden-Vector Encryption [56], and predicate encryption schemes supporting polynomial evaluation [173]. As shown in [23], making FE schemes verifiable enables more powerful applications. As an example, we can mention, VIPE can be used to construct what we call "*Polynomial Commitment Scheme*" (detailed in 5.2.1) which corresponds to a functional commitment of Badrinarayanan *et al.* for the polynomial evaluation *predicate*. The same construction can easily be adapted to construct functional commitments for the predicate-IP.

1.5.2 Revisiting Practical and Usable Coercion-Resistant E-Voting

In the second part of our research, we revisit the seminal coercion-resistant e-voting protocol by Juels, Catalano and Jakobsson (JCJ) and the attempts to make it usable and practical. In JCJ the user needs to handle cryptographic credentials and be able to fake these in case of coercion. In a series of three papers Neumann et al. analyzed the usability of JCJ and constructed and implemented a practical credential handling system using a smart card that unlocks the true credential via a PIN code, respectively fake the credential via faking the PIN. Finally we present several attacks and problems with the security of this protocol, especially an attack on coercion-resistance due to information leakage from the removal of duplicate ballots.

Another problem, already stressed but not solved by Neumann *et al.* is that PIN typos happen frequently and would invalidate the casting vote without the voter detecting this.

We construct different protocols which repair these problems. Further, the smart card is a trusted component that can invalidate cast votes without detection and can be removed by a coercer to force abstention, i.e. presenting a single point of failure. Hence we choose to make the protocols hardware-flexible *i.e.,* also allowing the credentials to be stored by ordinary means, but still being PIN-based and providing PIN error resilience. Finally, one of the protocols has a linear tally complexity to ensure an efficient scheme also with many voters.

In more detail, we investigate a hardware-independent protocol that can be implemented using a combination of a digitally stored cryptographic length key and a PIN only known by the voter. The long credential could be stored in several places or even hidden via steganography. At the ballot casting phase, the software will take as input the digital key and the password to form the credential submitted with the vote. Depending on the level of coercion, the coerced voter can either fake the long credential or, for stronger levels of coercion, the voter can reveal the digitally stored credential to the coercer but fake the PIN. Due to our improved tally, the coercer will not know if he got faked credentials or PINs. On the other hand, since the voter memorizes the PIN, there is a high chance of users making a PIN typo error which will invalidate the vote and remain undetected.

Note that naively giving feedback on the correctness of the PIN is not possible for coercion-resistance as it would allow the coercer to check whether he got a fake PIN or not. Instead, we define a set of allowed PIN errors (e.g., chosen by the election administrator). A ballot is valid if it has either a correct PIN or an allowed PIN error but invalid for other PINs. And our approach to implementing the protocol is based on polynomial evaluation, which allows us to determine whether or not a PIN is legitimate efficiently. This is accomplished by generating a list

$$\mathsf{ErrorList}_a = \{a_1 = a, a_2, \ldots, a_k\}$$

of approved PINs based on the user's PIN a. From this, we generate the following polynomial which has all $\mathsf{ErrorList}_a$ members as its root:

$$\mathsf{poly}_{\mathbf{pin}}(x) = \prod_{i=1}^{k}(x - a_i) = \sum_{i=0}^{k} p_i x^i.$$

To check the validity of the PIN, typed by the voter, it is then sufficient to check whether the polynomial value on this **pin** is equal to zero or not.

It is obvious that this polynomial must be kept secret at all times to prevent an adversary from recovering the PIN by factorizing it. As a result, we must operate with encrypted polynomials, which brings us to our next challenge: polynomial evaluation under this encryption. Namely, given the polynomial encryption as its encrypted coefficient,

$$\mathsf{poly}_{\mathbf{pin}}(x) = \sum_{i=0}^{k} p_i x^i \Rightarrow \mathsf{Enc}(\mathsf{poly}_{\mathbf{pin}})(x) = \sum_{i=0}^{k} \mathsf{cp}_i x^i,$$

as well as a ciphertext $\mathsf{CT}_{\mathbf{pin}} = \mathsf{Enc}(\hat{a})$ that is the encryption of the entered pin, we need to compute $\mathsf{Enc}(\mathsf{poly}_{\mathbf{pin}}(\hat{a}))$.

Therefore, in the next step, we have to find a way to prove publicly that the individual voter's polynomial is correctly evaluated without endangering the coercion-resistance. For example this would e.g., rule out voters evaluating the polynomials on the voter side only. Furthermore, for the sake of the coercion-resistance property, the protocol is constructed so that the tally phase will secretly check whether a given PIN is in the set of allowed PINs and will remove invalid ballots.

In chapter 7, we present a full description of our protocol with three instantiations of the protocol. We also provide a detailed security analysis of our protocol based on the KTV computational model introduced in [182] and the bPRIV privacy notion introduced in [46].

1.5.3 Deniable Vote Updating

A deniable vote updating scheme enables each voter to overwrite her previously submitted ballot, which she may have cast under coercion, such that no one else, including a possible coercer, can see whether or not the voter has subsequently updated her vote. Among others, this technique is employed in the hybrid e-voting system used for national elections in Estonia [154] and formerly in Norway . Here, voters can overwrite electronically cast ballots by submitting a physical ballot at the polling station. There are also solutions to deniable vote updating for completely remote e-voting systems, most notably [180, 197, 98].

In an e-voting system with deniable vote updating, "the vote casting process is no different from simpler voting systems that do not ensure coercion resistance" and "even in case of coercion, the concept of voting again to overwrite the vote cast under coercion would most probably fit into the mental models of the voters" [179].

Despite these significant advantages, existing e-voting systems with deniable vote updating have fundamental restrictions. For example, the deniable vote updating techniques proposed in [9, 196] require that the voters' submitted ballots are secretly compared pairwise[3], leading to quadratic complexity in the number of voters. This property is undesirable for elections with medium-size or larger electorates. More specifically, as demonstrated in [197], in an election with 180,000 voters, the solution by [9] would require more than one core *year* to do the comparisons.

The only *scalable* e-voting systems with deniable vote updating are [180, 197]. What these techniques have in common is that several indistinguishable dummy ballots hide the voters' re-voting pattern. Unfortunately, although [180] follows the concept of deniable vote updating, the counter-strategy proposed in [180] is cumbersome for human voters. If a voter in [180] wants to update a choice v submitted under coercion,they need to memorize v, invert it, multiply the result with her truly favorite choice $\hat{v}$, and submit a ballot for $v^{-1} \cdot \hat{v}$. The procedure becomes even more complex if the coercer asks the voter to submit several choices, each one updating the one submitted before. It is questionable whether human voters are actually able to execute this complex counter-strategy and thus whether [180] provides a sufficient level of coercion-resistance in practice.

[3]The latter however achieves everlasting privacy

In contrast to [180], the counter-strategy voters have to run in [197] is as easy as it could possibly be. In fact, if a voter was coerced to submit a vote for v (or even a sequence of votes), she can simply, at any later point of the submission phase, cast a vote for her truly favorite choice $\hat{v}$ (without having to memorize any previously cast a vote). On the downside, [146] demonstrated that [197] does not provide a reasonable level of security because there exists a single voting authority in [197] that needs to be trusted for all security properties, i.e., verifiability, privacy, and coercion-resistance. According to [146], this security issue is intrinsic to the approach taken in [197], and could thus, if possible, only be resolved by fundamental modifications.

Altogether, we can conclude that, to date, there does not exist a practically efficient e-voting system in the literature which is provably secure and provides human voters with a simple counter-strategy to defeat coercion. Therefore to overcome the unsatisfying state-of-affairs described, we propose a remote e-voting system that satisfies all of the following properties:

- Voters can deniably update their votes intuitively, resulting in some level of coercion-resistance.
- E2E verifiability and vote privacy is provably guaranteed without any additional trust assumptions besides the standards.
- Large-scale real-world elections can be realized efficiently.

1.5.4 Risk-Limiting Tallies

In this part of our research, we consider elections that publish anonymized voted ballots or anonymized cast-vote records for transparency or verification purposes, investigating the privacy implications, coercion, vote selling and exploring how partially masking the ballots can alleviate these issues.

Risk Limiting Tallies[4] (RLT) [164] is a technique that reveals only a random sample of ballots to mitigate some coercion threats while ensuring that the required confidence level in the election result is achieved. The intuition behind the technique is simple: Since only a random subset of ballots are revealed, the coerced voter can always claim that they followed all of the instructions, but their vote was masked. Put simply, there is plausible deniability.

Here we show how these ideas can be generalized and made more flexible and effective by masking at a finer level of granularity: at the level of the components of ballots. In particular, we consider elections involving complex ballots, where RLT may be vulnerable to pattern-based vote-buying. We propose various measures of verifiability and coercion-resistance and investigate how different sampling/masking strategies perform against these measures. Finally using methods from coding theory, we analyze signature attacks, bounding the number of voters who can be coerced.

1.5.5 Outline

The rest of this manuscript is organized as follows:

Part One: The first part of this thesis includes our research in the *"Verifiable Functional Encryption Scheme"*.

[4]This method can be applied for standard elections as well as for e-voting

- *Building Blocks; Functional Encryption Scheme:* In Chapter 2, we provide the foundation for our thesis by reviewing fundamental mathematical, algorithmic, and computational models and cryptographic primitives.
- *A Brief Survey on Zero-Knowledge Proof Systems:* Chapter 3 combines a brief survey and a preliminary chapter for our work. Since the concept of Zero-Knowledge proof system plays a significant role in our research, a short study on the Zero-Knowledge proof system is presented in chapter 3 to provide the reader with a deeper understanding of the subject. This survey includes formal definitions of zero-knowledge proofs and some of its variants. We also go in more detail over some of the Zero-Knowledge systems that we use the most of our study, such as the Sigma-protocol and Groth-Sahai NIWI proof system.

  It is important to mention although the majority of chapter 3 is devoted to results on the Zero-knowledge proof system, for the sake of clarity, we present a part of our contribution from [242] in section 3.12, rather than following the publication's chronological order.
- *Perfect Inner Product Encryption Schemes:* In Chapter 4 we present our perfect Inner Encryption scheme and its security proof based on the standard assumption.
- *Verifiable IPE:* In Chapter 5 which contain some part of our contribution from [242] we present the detail of the transformation from IPE to the Verifiable IPE.

Part Two: In the second part of this thesis, we present our research on secure and *"Verifiable e-Voting Protocols"*.

- *Building Blocks; e-Voting Protocol:* In chapter 6 we gather technical preliminaries related to the e-voting protocols we use in the second part of this thesis.
- *Practical and Usable Coercion-Resistant Remote E-Voting:* This Chapter is mostly based on our results in, *"Revisiting Practical and Usable Coercion-Resistant Remote E-Voting"* [98].
- *A New Technique for Deniable Vote Updating:* Chapter 8 presents our research to develop a deniable vote updating protocol. We stress that our research presented in this chapter is under submission.
- *Risk-Limiting Tallies:* This chapter discusses our contribution in *Who Was that Masked Voter? The Tally Won't Tell!* [227], we discuss certain risk-reducing tally strategies for elections with complex ballots.

Part I

Verifiable Functional Encryption Schemes

Chapter 2

Building Blocks; Verifiable Functional Encryption Schemes

"When the ancients discovered 'Phi', they were certain they had stumbled across God's building blocks for the world."
Dan Brown

The first part of this thesis includes our results in ***"Verifiable Functional Encryption Scheme"***. In this section, we introduce the notations and review the fundamental concepts and mathematical background that will be used throughout the first part. We also describe and discuss cryptographic primitives and the computational assumptions that will be used in our research.

Contents

2.1 Mathematical Notions and Notations

Sets and Numbers: We let $\mathbb{Z}$ denote the set of all integers and $\mathbb{N}$ all positive integers throughout the thesis. For any integer $n \in \mathbb{N}$, we present by $[n]$ the set $\{1, 2, \ldots, n\}$ and by $\{0,1\}^n$ the set of all bit-strings with length n.

The set of strings with arbitrary length, including the empty string ε, is presented by $\{0,1\}^*$ and the length of a string $s \in \{0,1\}^*$ is denoted by $|s|$. We use $|.|$ in several cases. If x is a string, $|x|$ shows the length of x, for a number p, $|p|$ shows the length of its binary representation. Moreover, in the case of a set, $|A|$ denotes the number of elements in the set.

For two integers $a, b \in \mathbb{N}$, we let $\mathsf{gcd}(a,b)$ and $\mathsf{lcm}(a,b)$ denote the *greatest common divisor* and the *least common multiple* for a and b, receptively, and we call a and b co-prime numbers if $\mathsf{gcd}(a,b) = 1$ equivalently, $\mathsf{lcm}(a,b) = a \times b$.

Euler and Carmichael's function: For positive integer $n \in \mathbb{N}$, the Euler's Phi function, $\phi(n)$, is the number of co-prime integers with n:

$$\phi(n) = \left|\{m \in \mathbb{N} : \mathsf{gcd}(n,m) = 1\}\right|,$$

and Carmichael's function, $\lambda(n)$ is the smallest positive integer such that:

$$a^{\lambda(n)} = 1 \mod n.$$

For a specific case, $n = p \times q$ where p and q are two distinct prime numbers, we have the following formulas:

$$\phi(n) = (p-1)\cdot(q-1), \qquad \lambda(n) = \mathsf{lcm}(p-1, q-1).$$

We denote by χ_S the characteristic function of the set S; $\chi_S(x) = 1$ if $x \in S$ and $\chi_S(x) = 0$ if $x \notin S$.

Cyclic groups and Bilinear maps: A commutative group $\langle \mathbb{G}, \cdot \rangle$ with respect to the action "$\cdot$" is a cyclic group if there exists a group member $g \in \mathbb{G}$ such that

$$\mathbb{G} = \langle g \rangle = \{1_{\mathbb{G}} = g^0, g, g^2, \ldots, g^n\}$$

where $1_{\mathbb{G}}$ is the neutral element of $\mathbb{G}$ with respect to the action "$\cdot$". In other words, every element $h \in \mathbb{G}$ can be represented as a power of g, namely for all $h \in \mathbb{G} : h = g^w$ for some integer w. According to a well-known theorem in group theory; for any h in finite group $\mathbb{G}$ of order n; $h^n = 1_{\mathbb{G}}$. Throughout this thesis, we only consider the finite group; $|\mathbb{G}| = n$ and we refer to $\mathbb{G}$ as a prime-order (composite-order) group if n is a prime (composite) number.

Definition 1 (Bilinear map [53])**.** *A bilinear map, $\mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{G}_T$, is defined over a pair of groups $\mathbb{G}_1, \mathbb{G}_2$ into the target group $\mathbb{G}_T$ with three properties:*

i. ***Bilinear:*** *For all $a, b \in \mathbb{Z}$ and $u \in \mathbb{G}_1, v \in \mathbb{G}_2$, $\mathbf{e}(u^a, v^b) = \mathbf{e}(u,v)^{ab}$.*

ii. ***Non-degenerate:*** *Consider any two generators g_1 and g_2 for groups $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, then $\mathbf{e}(g_1, g_2) \neq 1_{\mathbb{G}_T}$.*

iii. ***Computable:*** *There exists an efficient algorithm to compute the map.*

Additional Note. Cryptography relies heavily on two classes of cyclic groups: multiplicative subgroups of finite fields and (subgroups of) elliptic curves supplied with bilinear maps both in symmetric settings when $\mathbb{G}_1 = \mathbb{G}_2$ and asymmetric settings when $\mathbb{G}_1 \neq \mathbb{G}_2$ (See [201, 175, 174]). The use of pairing friendly elliptic curves for cryptography has been initiated by[53, 166, 167]. We refer to [111, 27, 110, 239] for further details on the subject.

While this work is concerned with generic bilinear maps, it is worth noting that significant research has been conducted on efficient bilinear mappings over elliptic curves that may be used to instantiate (efficiently) our results, such as Weil Pairing introduced by André Weil [258], Tate pairing [107] and one of the most efficient pairings, optimal pairing proposed by Frederik Vercauteren [251].

Modular Arithmetic: For any integer $n \in \mathbb{N}$, the set $\mathbb{Z}_n = \{0, 1, ..n-1\}$ with respect to addition modulo n (*i.e.,* where $a + b := [a + b \mod n]$) is a commutative group of order n, $\langle \mathbb{Z}_n, +_n \rangle$. Furthermore, we denote by $\langle \mathbb{Z}_n^*, \times_n \rangle$ the multiplicative group with respect to multiplication modulo n (*i.e.,* $a \times_n b := [a \times b \mod n]$). In fact, $\mathbb{Z}_n^*$ includes all integers in Z_n that are invertible modulo n. (*i.e.,* $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n : \gcd(a, n) = 1\}$). We will often abuse these notations and write $\mathbb{Z}_n$ for $\langle \mathbb{Z}_n, +_n \rangle$, and $\mathbb{Z}_n^*$ for $\langle \mathbb{Z}_n^*, \times_n \rangle$.

With respect to the above notations, we remind the following theorems from *Number Theory*:

$$\left|\mathbb{Z}_n^*\right| = \phi(n), \left|\mathbb{Z}_{n^2}^*\right| = \phi(n^2) = n\phi(n),$$
$$\forall a \in \mathbb{Z}_{n^2}^* : a^{\lambda(n)} = 1 \mod n\,,\; a^{n\cdot\lambda(n)} = 1 \mod n^2.$$

Legendre Symbol, Jacobi Symbol, and Quadratic Residue: For prime number p and integer $s \in \mathbb{Z}_p$, the ***Legendre Symbol*** of x modulo p is defined as follows:

$$\left(\frac{x}{p}\right) = \begin{cases} 0 \text{ If } x = 0, \\ 1 \text{ if } x = y^2 \text{ for some } y \in \mathbb{Z}_p^* \\ -1 \text{ otherwise} \end{cases}$$

For positive integer $n = p_1^{a_1} \dots p_k^{a_k}$, where p_i are distinct prime numbers and $a_i \in \mathbb{N}$, the ***Jacobi Symbol*** of x modulo n is:

$$\left(\frac{x}{n}\right) = \prod_{i=1}^{k} \left(\frac{x \mod p_i}{p_i}\right)^{a_i}.$$

We denote by $\mathbb{J}_n$ the subgroup of $\mathbb{Z}_n^*$ with Jacobi symbol 1, which has order $\phi(n)/2$, and it is the largest cyclic group contained in $\mathbb{Z}_n^*$. Furthermore, we let $\mathbb{QR}_n$ present a set of quadratic residues modulo n, which include integer x such that there exist $y \in \mathbb{Z}_n^* : x = y^2$. Using *Chinese Remainder Theorem* (**CRT**), we see that if x is a quadratic residue modulo n then $\left(\frac{x}{n}\right) = 1$. The converse is only true if n is a prime number. $\mathbb{QR}_n$ is also a cyclic subgroup of $\mathbb{Z}_n^*$ of order $\phi(n)/4$.

Probability and Distributions: For an arbitrary set $\mathcal{S}$, by $x \overset{\$}{\leftarrow} \mathcal{S}$ we indicate that x is chosen uniformly at random from $\mathcal{S}$. We use $\mathbf{U}_n$ as a random variable, uniformly distributed over $\{0, 1\}^n$. For any probability distribution $\mathcal{D}$, we let $x \overset{\mathcal{D}}{\longleftarrow} \mathcal{S}$ denotes choosing

x from $\mathcal{S}$ according to the probability distribution $\mathcal{D}$. We often use the term *random* to mean *uniformly at random* and sometimes, when it is clear from the context, we drop the distribution in $\xleftarrow{\mathcal{D}}$. We denote by $\Pr[X = s]$ the probability of a random variable X taking value s, and $\Pr_{x\leftarrow\mathcal{D}}[f(s) = \alpha]$ to denote the probability that $f(x)$ is equal to α, while s is sampled according to the distribution $\mathcal{D}$.

Definition 2 (Statistical Distance [105])**.** *Consider two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ over some finite set $\mathcal{S}$ and two random variables X_0 and X_1 sampled according to $\mathcal{D}_0$ and $\mathcal{D}_1$ respectively. The statistical distance between two distributions is defined as:*

$$\mathsf{d}_{\mathsf{static}} := \frac{1}{2}\sum_{s\in\mathcal{S}}\left|\Pr_{x\leftarrow\mathcal{D}_0}[x = s] - \Pr_{x\leftarrow\mathcal{D}_1}[x = s]\right|$$

2.2 Algorithms

By algorithm, we mean a "Turing Machine Program" which is classified into two types: deterministic and probabilistic algorithms.

The *probabilistic* algorithm (which we often use in this work) incorporates an additional Turing tape with random bits (a.k.a. random coins). In the probabilistic algorithm (Turing Machines), we assume that the random tape chooses a bit-string *uniformly* at random.

A different sort of randomized algorithm, the *non-uniform* probabilistic algorithm, has access to an advise string. As a result, the random coin is picked according to some distribution rather than the uniform distribution. While this makes the algorithm more robust than the (uniform) ones, it is, in fact, unrealistic. Therefore, we consider only algorithms of uniform complexity throughout our work, those that do not access the advice string.

A randomized Turing Machine, $\mathbf{A}$, runs in time $t_{\mathbf{A}}(|s|)$ if for all string s, $\mathbf{A}(s)$ halts within $t_{\mathbf{A}}(|s|)$ steps. For example, suppose we fix the input's length ($s \in \{0,1\}^n$) and consider its running time as a random variable (dependent on its coin tosses). In that case, we call the algorithm expected in polynomial time if the expectation of this random variable is polynomial in n. In contrast, if there exist some polynomial poly such that bounds the algorithm's running time on input s ($t_{\mathbf{A}}(s) < \mathsf{poly}(|s|)$), then we say that $\mathbf{A}$ is a (strict) polynomial-time algorithm. In our research, PPT stands for a probabilistic polynomial-time algorithm that implicitly considers only the uniformly randomized ones.

Non-uniformly Polynomial-Time Algorithms are polynomial algorithms that have access to some extra input (polynomial length string in input length) $\mathbf{A}(s) = B(s, \mathsf{r}_{\mathsf{poly}(|s|)})$ which B is a PPT algorithm.

For algorithm $\mathbf{A}$, we let $y \leftarrow \mathbf{A}(s)$ denote $\mathbf{A}$'s execution on input x with fresh random coins that outputs y as a result. Sometimes we explicitly indicate the random coin used by the algorithm by writing $y \leftarrow \mathbf{A}(s; \mathsf{random})$; we separate the input and randomness by ";". We stress that the symbol "random" represents the concept of randomness in the algorithm not an exact value. In other words, the random values in these two algorithms, $y_1 \leftarrow \mathbf{A}(s_1; \mathsf{random})$ and $y_2 \leftarrow \mathbf{A}(s_2; \mathsf{random})$ are not necessarily the same value, although we use the same notation.

Interactive Turing Machines: An interactive Turing Machine (ITM) is a Turing Machine that communicates with other Turing Machines using its two additional tapes.ITM has one communication input tape to receive messages and one output tape to send messages to other machines. We use the following notation regarding interactive algorithms:

- $\langle \mathbf{A}(x), \mathbf{B}(y)\rangle(z)$: Two algorithms, **A** and **B**, with inputs x and y and common input z, interacting with each other.
- $z \leftarrow \mathbf{A}(x)^{\rightleftharpoons \mathbf{B}(y)}$: The algorithm **A** takes x as input and outputs z after interacting with algorithm B with input y. x may or may not be equal to y. If y is not equal to x, it implicitly means that **A** (**B**) does not have knowledge about **B**'s (**A**) inputs.
- $\big[\mathbf{A}(s), \mathbf{B}(y)\big](z)$: The probability distribution of history descriptions generated by the interaction of **B** with **A** on common input z and private inputs x and y for each algorithm.

Definition 3 (**View of an interactive protocol** [250])**.** *Consider the interactive algorithms $\langle \boldsymbol{A}(x), \boldsymbol{B}(y)\rangle(z)$ with two parties **A** and **B** on common input z and private inputs x and y. Then **A**'s view of the interaction with **B**, denoted by $\mathcal{V}iew\big[\boldsymbol{A}(x)^{\rightleftharpoons \boldsymbol{B}(y)}\big](z)$, is a random variable $\langle \boldsymbol{A}, \boldsymbol{B}\rangle[x] = (m_1, m_2, \ldots, m_t; \mathsf{random})$ consisting of all the messages $m_1, \ldots, m_t$ exchanged between **A** and **B** together with the sub-string of* random *containing all the random bits that **A** has read during the interaction.*

2.3 A Background from Complexity Theory

"***P versus NP;***
a gift to mathematics, from computer science."
Steve Smale.

Following from the book, *The Foundations of Cryptography* [119], "Efficient computations correspond to computations that can be carried out by probabilistic polynomial-time Turing machines".

In this section we recall some definitions and concepts from the complexity theory. We begin by introducing three well-known complexity classes, **P**, **NP**, and **PSPACE**, and then inject the fourth class that has a significant impact on cryptography.

2.3.1 P, NP and PSPACE

We consider a language $\mathscr{L}$ as a set of bit-strings and we say a Turing Machine decide (recognize) the language if:

i. ***Completeness:*** For every $x \in \mathscr{L}$, the machine accepts x, outputs 1.

ii. ***Soundness:*** For every string $x \notin \mathscr{L}$ the algorithm rejects x , outputs 0.

Moreover, we say algorithm **A** decide the language $\mathscr{L}$ with a 2-sided error if for every statement $x \in \mathscr{L}$, the completeness and soundness properties, occur with some failure probability.

BPP-Class includes all language that can be decided in polynomial-time with two side-error, defined as follows:

Definition 4 (Bounded-Probability Polynomial Time)**.** *The complexity class **BPP** contains all language $\mathscr{L}$ which is recognized by a probabilistic polynomial-time Turing Machine M such that:*

i. ***Completeness:*** *For every $x \in \mathscr{L}$, the machine accepts x with probability of at least $\frac{2}{3}$:*

$$\forall x \in \mathscr{L} : \Pr[\, A(x) = 1\,] \geq \frac{2}{3}.$$

i. **Soundness:** *For every string $x \notin \mathscr{L}$ the algorithm rejects x , outputs 0 with a probability of at least $\frac{2}{3}$:*

$$\forall x \notin \mathscr{L} : \Pr[\, A(x) = 0\,] \geq \frac{2}{3}.$$

P-Class includes all languages $\mathscr{L}$ that are recognized by a deterministic polynomial-time algorithm:

$$\mathbf{P} = \{\mathscr{L} : \text{for some deterministic polynomial-time algorithm } \mathbf{A}^{\text{deterministic}}: \\ \mathbf{A}^{\text{deterministic}}(x) = 1 \iff \chi_{\mathscr{L}}(x) = 1\}$$

NP-Class includes all languages for which there are two algorithms; a computationally unbounded algorithm and a polynomial-time algorithm. The first algorithm takes $x \in \mathscr{L}$ as input and generates w, a bit-string of size polynomial in the $|x|$. The second algorithm is a boolean deterministic polynomial algorithm that outputs accept (or 1) if x belongs to the language and outputs reject (or 0) if x does not belong to the language.Formally we have the following definition:

Definition 5 (Complexity Class NP[119])**.** *A language $\mathscr{L}$ is in **NP**-class if there exists a Boolean relation $R_{\mathscr{L}}$ and a polynomial* poly(.) *such that $R_{\mathscr{L}}$ can be recognized in (deterministic) polynomial time, and $x \in \mathscr{L}$ if and only if there exists a w such that $|w| \leq$* poly($|x|$) *and $(x, w) \in R_{\mathscr{L}}$. We call w a witness for the membership of the statement x in language $\mathscr{L}$.*

Examples: Consider a graph G (big graph) and the following two questions:

1. Is G an Eulerian Graph? *i.e., Does G have a path that visits every edge exactly once?*
2. *Is G a Hamiltonian graph?* i.e., Does G have a path that visits every vertex exactly once?

Based on these questions, we can define two languages, $\mathscr{L}_1$ and $\mathscr{L}_2$:

1. $\mathscr{L}_1 = \{G : G$ is an Eulerian Graph$\}$
2. $\mathscr{L}_2 = \{G : G$ is a Hamiltonian Graph$\}$

The first language belongs to class **P** because, according to a well-known study in *Graph Theory*, a connected graph has an Eulerian cycle if and only if every vertex has an even degree. In contrast, the second language belongs to the **NP**-class, because there is no *known* deterministic algorithm that can decide whether or not a graph is Hamiltonian. Moreover, having a Hamiltonian cycle of graph G, one can verify the correctness of the witness; particularly, a deterministic algorithm R($x = G, w =$ cycle) exists which indicates the Hamiltonian problem belongs to the **NP**-class.

PSPACE-Class: The third notable category is the **PSPACE**-class, which includes all languages recognized by a Turing Machine in a polynomial amount of space. It should be noted that there is no time constraint in languages in **PSPACE**. It is believed that **NP** and **P** are strictly contained in **PSPACE** and **NP** respectively:

$$\mathbf{P} \subset \mathbf{NP} \subset \mathbf{PSPACE}$$

2.3.2 Interactive Proof Systems

A remarkable connection between the **NP** and **PSPACE** (assuming that **NP** $\neq$ **PSPACE**) derives from another complexity class, **IP** that incorporates some relaxations relative to the **NP**-class. Tolerating a tiny amount of errors by injecting some randomness into the algorithms and allowing algorithms to interact, namely, sending some challenging value, and receiving the response adaptively. Formally, we define the interactive proof as follows:

Definition 6 (**Interactive Proof Systems** [128, 129]). *An interactive proof system for a language $\mathscr{L}$ (or its corresponding relation $R_{\mathscr{L}}$) is a two-party game between the prover and the verifier. The verifier executes a probabilistic polynomial algorithm* Verify *and the prover executes a computationally unbounded strategy* Prove*, satisfying the following properties:*

- ***Efficiency:*** $\langle \mathsf{Prove}, \mathsf{Verify} \rangle (x)$ *is polynomially bounded.*
- ***Completeness:*** *For every* $(x, w) \in R_{\mathscr{L}}$*: the* Verify *algorithm accepts after interacting with the prover on common input x, with probability at least* $1 - \frac{1}{\mathsf{poly}(|x|)}$ *for some polynomial* poly.
- ***Soundness:*** *For some polynomial* poly*, it holds that for every $x \notin \mathscr{L}$ and every potential strategy* Prove^**, the verifier, rejects with a probability of at least* $\frac{1}{\mathsf{poly}(|x|)}$.

The class of languages having interactive proof systems is denoted by ***IP****.*

In the original definition of an interactive proof system in [128] they consider an interactive Turing machines; we use the definition that was introduced in [129].

Additional Note. It is clear that all languages in **NP** have a one-round interactive proof system that merely involves the prover sending (x, w) to the verifier. The verifier performs the polynomial-time algorithm relation and accepts or rejects the result based on whether or not $\mathrm{R}(x, w) = 1$. Therefore, it is natural to ask whether these new class, **IP**, is strictly more powerful than **NP** in terms of recognition of additional languages.

Much research has been done addressing the first issue, and the result illustrates a fundamental relation between **NP** and **PSPACE**. Adi Shamir in [235] by using a novel proof technique, *"arithmetization"* and Goldreich in [198] using some **NP**-complete reduction (permanent of a matrix) showed that **IP** = **PSPACE**. Assuming **NP** is not equal to **PSPACE**, this result shows that the class **IP** strictly includes **NP**.

Another relevant question is what complexity class we obtain by relaxing a single criterion of the **NP**-class. In terms of error tolerance, some works do not require perfect completeness and allow for failures of less than $\frac{1}{3}$. Goldreich *et al.* established two interesting theorems in [108]:

Theorem 2.3.1. *If $\mathscr{L}$ has an n-round interactive proof system, then $\mathscr{L}$ has an $n+1$-round interactive proof system with perfect completeness.*

Theorem 2.3.2. *If $\mathscr{L}$ has an interactive proof with perfect soundness, then $\mathscr{L}$ is in an* **NP** *language.*

This result shows the necessity and sufficiency of the relaxations. Without tolerating any soundness failure, the **IP** collapsed to **NP**, and the two-sided error version equals the one-sided. Moreover, in terms of interaction, the non-interactive form of **IP** was studied in [20] and it was proved that it contains **NP**, but the converse is unknown.

Additional Note. In terms of interaction, interactive proof systems come in various flavors. The non-interactive form of **IP** was studied in [20] and proved it contains **NP**, but the converse is unknown. Arthur-Merlin games, also introduced by [20], are restricted cases in which it is required that a verifier sends only the outcome of the coin it tosses. This type of interactive proof is known as a public coin interactive proof, as opposed to a private coin interactive proof, in which the verifier may keep its internal state secret. Interestingly it has been proved [129] that this restricted case has essentially the same power as the general case. In other words, each interactive proof system can be considered as a public-coin interactive proof system. [1] More precisely, Goldwasser and Sipser [122] prove that a public-coin interactive proof system can recognize any language with an n-round private-coin interactive proof system with $n+2$-rounds.

2.4 Provable Security

In their outstanding book ***Introduction to Modern Cryptography*** [172], Katz and Lindell characterized modern cryptography by its threefold principles: *formal definitions, precise assumptions* and *proofs of security.* The first step towards modern cryptography is establishing a formal definition. This specification provides a comprehensive view of our model and helps us answer the essential questions:

What do we wish to safeguard? Is it the key? Is it the whole message or just some partial information that allows us to discover what kind of adversary we are trying to protect from? Are we trying to safeguard our system against what kind of adversary? How powerful are our adversaries, and what resources does the adversary possess?

Answering these questions helps us establish the basic steps of an approach so-called provable approach, put simply, is a security that can be proved, and it has the following steps:

1. Define the system you wish to protect and define the security goal precisely.
2. Define your adversarial power, resources and capabilities.
3. Specify the assumption.
4. Prove that if the assumptions hold true, the adversary with the described power cannot breach your system.

In this section, we present our definition, concept, and the security model that we use in our research, the approach which is called Provable security.

[1]This is not the case in Zero-Knowledge proof systems

2.4.1 Computational Secrecy

Perfect secrecy requires that no information about an encrypted message is revealed, even to an adversary with unlimited computational power. However, as Shannon pointed out, the price of achieving perfect secrecy is prohibitively high in the real world. In addition, while perfect secrecy is a worthy objective, it is also unnecessarily strong. Likewise, a cryptosystem with small information leakage to a computationally bounded adversary is still considered secure in real-world applications. These considerations lead us to define another concept of secrecy: *computational security.*

There are two approaches to computational security:

i. ***The concrete approach*** to computational security estimates the maximal security level of the system based on the adversary's resources (e.g., time). A scheme is (t, b)-secure if any adversary with a time limit of at most t succeeds in violating system security with a probability of at most b. This approach is most applicable in practice, whereas we employ an asymptotic in the theoretical study.

ii. ***The asymptotic approach*** is rooted in complexity theory, and it equates efficiency with a probabilistic polynomial-time algorithm and slight probability with negligible functions (Definition 7). In complexity theory, we usually measure the algorithm's running time based on the length of its input.

Definition 7. *A function* $\mathsf{negl}(.) : \mathbb{N} \mapsto \mathbb{R}$ *is a negligible function if for any polynomial* poly *there exits a number* $n^* \in \mathbb{N}$ *such that:*

$$\forall n > n^* : |\mathsf{negl}(n)| < \frac{1}{\mathsf{poly}(n)}$$

The dual of negligible function is an *overwhelming* function. We say f is overwhelming if $1 - f(.)$ is a negligible function.

Security Parameter. Although the asymptotic approach relies on complexity, it is dangerous to tie the security failure to the input length from the security point of view, as we may need to guarantee high security even for short statements. For this purpose, we use the concept of ***security parameter***, a positive integer $\ell \in \mathbb{N}$, in the asymptotic approach. We bind all algorithms in a cryptosystem together by using the security parameter as their common input. This allows us to investigate the system's security comprehensively. In order to respect the notion and terminology of complexity theory, we use a unary representation of security parameters as input to the algorithms.

Therefore to put the computational secrecy in formal definition, we will have the following:

Definition 8 (**Computational Secrecy**)**.** *A cryptosystem has computational secrecy if any PPT adversary succeeds in breaking the scheme with at most negligible probability in security parameter:*

$$\Pr\left[\mathsf{Succ}_{\mathcal{A}}(1^{\ell})\right] < \mathsf{negl}(\ell)$$

After establishing the computational framework, proceed by defining other concepts required for our security model.

Adversary: While we correspond an efficient method that runs in probabilistic polynomial time to ensure a higher level of security, an adversary is considered a non-uniform probabilistic algorithm. We distinguish active and passive adversaries

- *Passive adversary* attempts to obtain secret information while (honestly) following the cryptographic protocol. For example, honest voters cast their ballot on the public bulletin board in an election. Some malicious party tries to learn some voters' choice by observing the published information but do not intervene. However, security against passive adversaries does not address all possible situations.

- *Active adversary* forces parties to run the protocol in an unexpected way. A concrete example of this could be a coercer in an election who forces a voter by casting a ballot with a particular pattern. Such executions may cause additional information leakage, making the protocol less secure. Such adversaries are referred to as "active adversaries" since they disobey the established procedure.

Oracle Access: Sometimes, along with inputs and random coins, we provide algorithms with access to oracles. Oracles are a type of black-box that accepts some inputs and provides some output. They are intended to represent the possibility that an algorithm may obtain the answers to some queries without stating how they are computed. Given an oracle $\mathcal{O}$, we use $A(x)^{\rightleftharpoons\mathcal{O}(y)}$ to denote that the algorithm A is given oracle access to $\mathcal{O}$ when it is executed on the inputs x and y.

We highly rely on the concept of indistinguishable distributions in this computational secrecy. We differentiate three types of distribution indistinguishability:

1. ***Perfect Indistinguishability:*** Two distributions, $\mathcal{X} = \{\mathcal{X}_\ell\}_{\ell\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\ell\}_{\ell\in\mathbb{N}}$ are perfectly indistinguishable if from the point of view of any PPT adversary they look identical:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathcal{X},\mathcal{Y}}(1^\ell) = \left|\Pr\left[\,\mathcal{A}(x) = 1 \mid x \xleftarrow{\$} \mathcal{X}_\ell\,\right] - \Pr\left[\,\mathcal{A}(x) = 1 \mid x \xleftarrow{\$} \mathcal{Y}_\ell\,\right]\right| = 0$$

2. ***Statistically Indistinguishability:*** Two distributions ensemble $\mathcal{X} = \{\mathcal{X}_\ell\}_{\ell\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\ell\}_{\ell\in\mathbb{N}}$ are statistically indistinguishable if their statistical distance(Definition 2) is negligible:

$$\mathsf{d}_{\mathsf{static}}(\mathcal{X}_\ell, \mathcal{Y}_\ell) < \mathsf{negl}(\ell)$$

3. ***Computationally Indistinguishability:*** Two distributions ensemble $\mathcal{X} = \{\mathcal{X}_\ell\}_{\ell\in\mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\ell\}_{\ell\in\mathbb{N}}$ are computationally indistinguishable if the advantage of any PPT adversary is negligible:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathcal{X},\mathcal{Y}}(1^\ell) = \left|\Pr\left[\,\mathcal{A}(x) = 1 \mid x \xleftarrow{\$} \mathcal{X}_\ell\,\right] - \Pr\left[\,\mathcal{A}(x) = 1 \mid x \xleftarrow{\$} \mathcal{Y}_\ell\,\right]\right| < \mathsf{negl}(\ell)$$

We only consider computational security throughout this thesis, so we usually omit the computational word.

There are two methods to prove the security of some primitives in the computational model: the *Simulation-Based* [191] and the *Game-Based* [238] methods.

2.4.2 Simulation-Based Security

In a simulation-based paradigm, we prove the primitive's security by designing a simulator with some *ideal functionality*. Consider a real-world electronic election system in which we hope no malicious party may learn voter choices. Imagine an ideal model in which a thoroughly trusted party has secure channels to all voters, receives all voters' choice through some secure channels, and only publishes the outcome. As is obvious, a perfect model is secure by definition. We demonstrate a simulator that provides the same output as the adversary in the real world while interacting with the ideal "functionality" in the ideal world. In such a scenario, the system in the real world has the same level of security as the scheme in the ideal world, which is secure by definition [191].

2.4.3 Game-Based Security

The game-based method is characterized by a game centered on some generic primitive that played between two parties: the challenger $\mathcal{C}$, who has access to some secret values, and the adversary, $\mathcal{A}$ who only has access to some oracles plus whatever the challenger discloses during the game. The challenger challenges the adversary with a specific goal in mind. While interacting with the challenger, the adversary attempts to respond to the challenge. If the adversary achieves the predetermined goal, we say he has won the game.

For example, in an interactive game in which the challenger tosses a coin, a goal could be that the adversary was correctly guessing the outcome of coin-tossing. Clearly, the adversary can guess correctly in this scenario, at least with a probability of $\frac{1}{2}$ (random guessing). However, the interaction between the adversary and the challenger may help the adversary win the game with advantage of more than $\frac{1}{2}$. This is where we establish the security level of the generic primitive. By determining how much *better* the adversary performs than a trivial adversary that merely guesses the coin outcome.

More precisely, the game-based paradigm includes the following two steps:

1. ***Reduction Mechanism:*** Relates the security property of some cryptosystems to mathematical problems. As Goldwasser and Micali laid the foundations of provable security in [127], a cryptosystem or cryptographic protocol is provably secure if an adversary who breaches the system's security can be turned into an adversary who breaks a challenge that is conjectured to be hard. Therefore the reduction does not establish any level of security for the scheme on its own. However, assuming that no efficient algorithm can solve the underlying problem (challenge) with a high probability, the reduction ensures that the probability of breaching the cryptosystem is negligible.

   As a result of the above explanation, the reduction mechanism may be summarized in the following phases.

   (a) Consider a hard problem, X, for which no efficient (PPT) algorithm is believed to exist.

   (b) Assume an effective adversary $\mathcal{A}$ that attacks the cryptosystem C with a success probability of s.

   (c) Develop an efficient algorithm $\mathcal{B}$ that attempts to solve problem X by using adversary $\mathcal{A}$ as a subroutine. In reality, $\mathcal{B}$ simulates an instance of scheme C for $\mathcal{A}$ as resulting in; As far as $\mathcal{A}$ can tell, it appears to be engaging with C.

   (d) Prove that when $\mathcal{A}$ succeeds in breaking the instance of C (that $\mathcal{B}$ is simulating), then $\mathcal{B}$ should be able to solve the instance x of X.

   Taken together, the above suggests that $\mathcal{B}$ solves X with non-negligible probability, which contradicts our assumption that no efficient algorithm solves the problem with non-zero probability. As a result, the scheme's security will be proved.

Additional Note. We benefit from this approach in two ways. Firstly, it eliminates many insecure protocols. Additionally, it allows cryptanalysis to concentrate on investigating a limited number of specific, well-defined problems. As examples of the problems considered hard problems, we can name factorization and discrete logarithm, the inverse of exponentiation in some specific cyclic groups or the shortest vector in a lattice.

2. ***Game Hopping Mechanism:*** After associating our primitive security property with a hard problem, we design a sequence of slightly different games, each computationally close to the previous one. Specifically, the adversary cannot distinguish between the two unless it solves the difficult problem that we assumed no efficient adversary could solve. This contradiction implies that the adversary could not distinguish between two consecutive games. Further, the game should be constructed to capture the desired security property for the primitive that we wish to establish.

Experiments and Games. Consider an experiment (game) as an interactive game between a challenger and an adversary (or a distinguisher) in which after some rounds of interactions between $\mathcal{A}$ and $\mathcal{C}$, the challenger selects bit β. The adversary must output bit β' (in fact, guess the bit β) based on all information he gets during the execution. The adversary would win the game if $\beta = \beta'$.

$$\mathsf{Succ}_{\mathcal{A}}^{\mathsf{Exp}}(1^{\ell}) = |\Pr\left[\beta = \beta'\right]$$

Advantage. We define $\mathcal{A}$'s advantage as the ability of the adversary to guess the bit correctly or win the game better than random guesses, $\frac{1}{2}$.

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Game}}(1^{\ell}) = |\Pr\left[\mathcal{A} \mapsto 1 \mid \mathcal{C}^{\mathsf{Game}} \mapsto 1\right] - \Pr\left[\mathcal{A} \mapsto 1 \mid \mathcal{C}^{\mathsf{Game}} \mapsto 0\right]$$

Additional Note. In some certain experiments, the adversary is required to output some information, not a single bit. These games are mostly used in computational problems such as factorization problems. (See 2). In contrast the above experiments, which are related to a challenging bit, are mostly used in the decisional problem (See 7).

2.5 One Way Functions

Considering the basic concepts of computational complexity theory, now is the time to define one of the most important cryptographic concepts. Therefore we introduce one of the essential concepts in modern cryptography theory, the One-Way Function.

Definition 9. *[(Strongly) One-Way function:] Function $f : \{0,1\}^* \mapsto \{0,1\}^*$ is a (Strongly) one-way if it is efficiently computable and hard to invert:*

- ***Efficiently computable:*** *There exists a deterministic polynomial-time algorithm that for any $x \in \{0,1\}^*$ compute $f(x)$.*
- ***Hard to invert:*** *For any PPT algorithm $\mathcal{A}$, the following probability is negligible;*

$$\Pr_{x \leftarrow \mathbf{U}_n, \mathsf{random}}\left[y \leftarrow \mathcal{A}(f(x), 1^{\ell}; \mathsf{random}) \mid f(x) = y\right] < \mathsf{negl}(\ell)$$

Suppose we replace the PPT property of algorithm $\mathcal{A}$ with a non-uniformly probabilistic algorithm (a family of polynomial circuits). In that case, we get a more powerful concept known as a Non-Uniformly Strong One-Way Function. Obviously if f is one-way in a non-uniform way, then it is also one-way. The converse is not necessarily true, although it is widely believed that non-uniformly one-way exists, and all examples in the preceding subsections are believed to be non-uniformly one-way functions [119]. A weaker notion of one-way function results from using the following formula in definition9:

$$\Pr_{x \leftarrow \mathbf{U}_n, \mathsf{random}}\left[y \leftarrow B(f(x), 1^{\ell}; \mathsf{random}) \mid f(x) \neq y \right] = 1 - \mathsf{negl}(\ell)$$

Not every weak $One - Way$ is a strong One-Way function, but interestingly, it is proved that if there exists a weak One-Way then there exists a strong One-Way function.

Theorem 2.5.1. *Weak one-way functions exist if and only if strong one-way functions exist. [119]*

Formulating one-way functions as in Definition 9 is appropriate for abstract discussion. However, to conduct more robust research, we require some modifications:

- Rather than considering one-way functions acting on an infinite domain (*i.e.,* $\{0,1\}^*$*), we consider infinite sets of functions; each is operating on a finite domain.*
- *In many cryptosystems, we must have a unique preimage. Hence we prefer only to consider one-way permutation rather than one-way function.*
- *Also, to retrieve the encrypted message from the ciphertext, we need to be able to efficiently invert y and find $x : f(x) = y$ with the help of some additional input, called* **trapdoor**.

The above requirements bring us to the definition of Trapdoor Permutation Collection:

Definition 10. ***Collection of Trapdoor Permutations One-Way Function [119]:*** *Let $\bar{I}$ be a set of bit-string, $\bar{I} \subset \{0,1\}^*$ and $\bar{I}_n = \bar{I} \cap \{0,1\}^*$. A collection of permutations with indices in $\bar{I}$ is a set $\{f_i : D_i \mapsto D_i\}$ such that each f_i is one-to-one on the corresponding domain D_i. Such a collection is called a trapdoor permutation if there exist four probabilistic polynomial-time algorithms I, D, F and F^{-1} such that the following five conditions hold:*

- ***Index and trapdoor selection**: For every n;*

$$\Pr\left[I(1^n) \in \bar{I}_n \times \{0,1\}^* \right] > 1 - 2^{-n}$$

- ***Selection in the domain:** For any $n \in \mathbb{N}$ and $i \in \bar{I}_n$:*
  1. $\Pr[D(i) \in D_i] > 1 - 2^n$.
  2. *Conditioned on $D(i) \in D_i$, the output is uniformly distributed in D_i. That is for every $x \in D_i$: $\Pr[D(i) = x \mid D(i) \in D_i] = \frac{1}{|D_i|}$. without loss of generality $D_i \subset \{0,1\}^{\mathsf{poly}(|i|)}$.*

- ***Efficient evaluation**: For every n*

$$\Pr\left[F(i,x) = f_i(x) \right] > 1 - 2^{-n}.$$

- ***Hard to invert:*** *Let I_n be a random variable describing the distribution of the first element in the output of $I(1^n)$, and $X_n := D(I_n)$. Then for every probabilistic polynomial-time algorithm $\mathcal{A}$, every positive polynomial* $\mathsf{poly}(.)$ *and all sufficiently large n:*

$$\Pr\left[\mathcal{A}(I_n, f_{I_n}(X_n)) = X_n\right] < \frac{1}{\mathsf{poly}(n)}.$$

- ***Inverting with trapdoor:*** *For every $n \in \mathbb{N}$, any pair (i, t) in the range of $I(1^n)$ such that $i \in \bar{I}_n$ and every $x \in D_i$:*

$$\Pr\left[F^{-1}(t, f_i(x))\right] > 1 - 2^{-n}.$$

2.5.0.1 The RSA Collection as a One-Way Permutation:

One of the most well-known collections of one-way trapdoor permutations, is the RSA family, proposed in [222].Following definition 10, we describe the RSA collection as follows:

Definition 11. *The RSA trapdoor one-way collection, $RSA = \{RSA_{N,e}\}_\ell$ is collection of one-way trapdoor permutation for index set, $I = (N, e)$, function $RSA(N, e, x) = x^e \mod n$ and domain $D_{(N,e)}$ where $N = p \cdot q$ for two prime numbers p and q with length $\frac{1}{2} \cdot \log_2 N$ and $\mathsf{gcd}(e, \phi(N)) = 1$. The function of index $n = (N, e)$ has domain $D_\ell = \{1, \ldots, N\}$ and maps $x \mapsto x^e \mod N$. Using the fact that e is relatively prime to $\phi(N) = (p-1)\cdot(q-1)$, it can be shown that the function is in fact a permutation over its domain. Hence, the RSA collection is a collection of permutations. $RSA.I, RSA.D, RSA.F$:*

- ***Index and trapdoor selection**: The index set is $I = (N, e)$ where $N = pq$ for two (large) prime numbers p and q, $2^{\ell-1} \le p < q < 2^\ell$. The number e is a co-prime with $\phi(N)$;* $\mathsf{gcd}(e, \phi(N)) = 1$ *which is uniformly selected among the admissible possibilities.*

- ***Selection in domain:** We set $D_{(N,e)} = \{1, 2, \ldots, N\}$ and algorithm D_{RSA} selects (almost) uniformly an element in the set $D_{(N,e)}$.*

- ***Efficient evaluation**: $RSA(N, e, x) = x^e \mod n$,*

- ***Hard to invert:** It is proved that the RSA function is hard to invert without the trapdoor (See 1).*

- ***Inverting with trapdoor:** The corresponding trapdoor would be d such that $e \cdot d = 1$* $\mod \phi(n)$. *With the help of number theory, we know that*

  $RSA.F_{(N,e)}(x) = x^e = y \implies y^d = x^{e\cdot d = 1[\mod \phi(N)]} = x[\mod N]$

In order to avoid heavy notation, instead of the above description, we denote by $\mathcal{G}^{\mathsf{RSA}}(1^\ell)$ the PPT algorithm which generates the RSA tuple (p, q, N, e, d) such that p and q are two strong prime numbers and

$$2^{\ell-1} \le q < 2^\ell,\ N = p \times q,\ e \in \mathbb{Z}_n, \mathsf{gcd}(e, \phi(n)) = 1,\ e \cdot d = 1 \mod \phi(N).$$

Assumption 1. *It is believed that, for any PPT algorithm $\mathcal{A}$ the success probability in Figure 2.1 is negligible in ℓ.*

FIGURE 2.1: $\mathsf{Exp}^{\mathsf{rsa}}_{\mathcal{A}}(1^{\ell})$, RSA Experiment

Challenger steps:	***Adversary steps:***	***Success probability:***
1.$(e,p,q) \leftarrow I_{RSA}(1^{\ell})$		
2.$(x) \leftarrow D_{(N,e)}$	4.$x' \leftarrow A(1^{\ell},(N,e),y)$	$\mathsf{Succ}^{\mathsf{rsa}}_{\mathcal{A}}(1^{\ell}) = \Pr\left[x'^{e} = y\right]$
3.$RSA.F_{(N,e)}(x) = y$		

2.6 Computational Assumptions

This section reviews the classical computational assumptions that will underpin the remainder of this thesis, which we divide into two broad categories: discrete-logarithm-based assumptions and factorisation-based assumptions.

2.6.1 Factorization-Based Assumptions

Factorization Problem: The hardness of factorization, intuitively, states the following, for any PPT adversary $\mathcal{A}$ who is given the number $n = pq$, the product of two random primes p and q, the probability of generating two number p' and q' such that $n = p' \times q'$ is negligible in terms of $\ell = |p|$.

It is easy to see that the probability would not be negligible if either p or q is a small prime number. Hence we need to define a precise setting for factorization assumption. One of the standard settings is the one where it is assumed that p and q are strong prime numbers (Blum primes), namely, $p' = \frac{p-1}{2}$ and $q' = \frac{(q-1)}{2}$ are also prime numbers. Here we consider the general setting.

Assumption 2. *Factorization assumption holds true if the following probability (success probability) for any PPT adversary $\mathcal{A}$ is negligible.*

$$\begin{aligned}\mathsf{Succ}^{\mathsf{factorization}}_{\mathcal{A}}(\ell) =&\Pr\left[\mathcal{A}(n) \mapsto (p',q') \mid n \leftarrow \mathcal{G}^{\mathsf{RSA}}(1^{\ell}), n = p' \times q'\right] \\ &<\mathsf{negl}(\ell)\end{aligned} \tag{2.1}$$

Decisional Composite Residuosity Problem; (DCR) states that that given the RSA number n and a random integer x, it is hard to decide whether x is an n-residue modulo n^2.

Assumption 3. *DCR assumption holds true if the following probability (success probability) for any PPT adversary $\mathcal{A}$ is negligible.*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{DCR}}_{\mathcal{A}}(\ell) =&\big|\Pr\left[\mathcal{A}(n,x) = 1 \mid n \leftarrow \mathcal{G}^{\mathsf{RSA}}(\ell), z \leftarrow \mathbb{Z}_n, x = z^n\right] \\ &- \Pr\left[\mathcal{A}(n,x) = 1 \mid n \leftarrow \mathcal{G}^{\mathsf{RSA}}(\ell), x \leftarrow \mathbb{Z}_n\right]\big| < \mathsf{negl}(\ell)\end{aligned}$$

2.6.2 Discrete Logarithm-Based Assumptions

Discrete Logarithm Problem; (DL). Let $\mathcal{G}_{DL}(1^{\ell})$ be a PPT algorithm that generates a group $\mathbb{G} = \langle g \rangle$ with some random generator g. Discrete Logarithm problem states that given $(\mathbb{G}, g, h \in \mathbb{G})$ it is hard to compute α such that $h = g^{\alpha}$. Formally:

Assumption 4. *The discrete logarithm assumption holds for* $(\mathbb{G}, g) \leftarrow \mathcal{G}_{DL}$ *if for all non-uniform polynomial-time adversary* $\mathcal{A}$*, we have:*

$$\left|\Pr\left[\mathcal{A}(\mathbb{G},(g,h)) = \alpha' | (\mathbb{G},g) \leftarrow \mathcal{G}_{DL}(1^{\ell}), \mathsf{r} \xleftarrow{\$} \mathbb{Z}, h = g^{\mathsf{r}}\right]\right| < \mathsf{negl}(\ell)$$

Decisional Diffie-Hellman Problem; (DDH). Let $(\mathbb{G}, g, p) \leftarrow \mathcal{G}_{\mathsf{DDH}}(1^{\ell})$ be a PPT algorithm that generates a group setting. The decisional Diffie-Hellman problem (DDH) states that given $(g, g^{\alpha}, g^{\beta}, Z) \in \mathbb{G}^4$ for random integers α, β it is hard to tell whether $Z = g^{\alpha\beta}$ or a random element in $\mathbb{G}$.

Assumption 5. *The decisional Diffie-Hellman assumption holds for* $\mathcal{G}_{\mathsf{DDH}}$ *if for all non-uniform polynomial-time adversary* $\mathcal{A}$ *we have:*

$$\begin{aligned}&\big|\Pr[\mathcal{A}(\mathbb{G},(g,g^{\alpha},g^{\beta},Z)) = 1 \mid (p,\mathbb{G},,g) \leftarrow \mathcal{G}_{\mathsf{DDH}}(1^{\ell});\\ &Z = g^{\alpha\beta}, g \xleftarrow{\$} \mathbb{G}, (\alpha,\beta) \leftarrow \mathbb{Z}_p] -\\ &\Pr[\mathcal{A}(\mathbb{G},(g,g^{\alpha},g^{\beta},Z)) = 1 \mid (p,\mathbb{G},,g) \leftarrow \mathcal{G}_{\mathsf{DDH}}(1^{\ell});\\ &g \xleftarrow{\$} \mathbb{G}, (\alpha,\beta,\mathsf{r}) \leftarrow \mathbb{Z}_p, (Z = g^{\mathsf{r}})]\big| < \mathsf{negl}(\ell)\end{aligned}$$

Decisional Linear Problem; (DLin). Let $\mathcal{G}_{\mathsf{DLin}}(1^{\ell})$ be a PPT algorithm generates a bilinear group setting, that takes security parameter ℓ as input and outputs ℓ-bit prime p, the descriptions of two groups $\mathbb{G}$ and $\mathbb{G}_T$ of order p and a bilinear map $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. The decisional linear problem (DLin) introduced by Boneh, Boyen and Shacham [51] states that given $(g, g^{\alpha}, g^{\beta}, g^{r\alpha}, g^{s\beta}, Z) \in \mathbb{G}^6$ for random integers α, β, r, r it is hard to tell whether $Z = g^{r+s}$ or a random element in $\mathbb{G}$.

Assumption 6. *The decisional linear assumption holds for* $\mathcal{G}_{\mathsf{DLin}}$ *if for all non-uniform polynomial-time adversary* $\mathcal{A}$ *we have:*

$$\begin{aligned}&\big|\Pr[\mathcal{A}(\mathbb{G},(g,g^{\alpha},g^{\beta},g^{r\alpha},g^{s\beta},Z)) = 1 \mid (p,\mathbb{G},\mathbb{G}_T,\mathbf{e},g) \leftarrow \mathcal{G}_{\mathsf{DLin}}(1^{\ell});\\ &Z = g^{r+s}; (\alpha,\beta,r,s) \leftarrow \mathbb{Z}_p] -\\ &\Pr[\mathcal{A}(\mathbb{G},(g,g^{\alpha},g^{\beta},g^{r\alpha},g^{s\beta},Z)) = 1 \mid (p,\mathbb{G},\mathbb{G}_T,\mathbf{e},g) \leftarrow \mathcal{G}_{\mathsf{DLin}}(1^{\ell})\\ &Z = g^{\mathsf{r}}; (\alpha,\beta,r,s,\mathsf{r}) \leftarrow \mathbb{Z}_p]\big| < \mathsf{negl}(\ell)\end{aligned}$$

The Decisional Bilinear Diffie-Hellman Problem; (DBDH). For a bilinear group seeting $(\mathbb{G}, \mathbb{G}_T, g, \mathbf{e})$, DBDH assumption states the hardness for PPT adversaries of solving the following problem. On input $(g, g^{\alpha}, g^{\beta}, g^{\gamma}, Z) \in \mathbb{G}^4 \times \mathbb{G}_T$, decide whether $Z = \mathbf{e}(g, g^{\alpha\beta\gamma})$ or Z is a random element in $\mathbb{G}_T$.

Assumption 7. *The decisional linear assumption holds for* $\mathcal{G}_{\mathsf{DBDH}}$ *if, for all non-uniform polynomial-time adversary* $\mathcal{A}$*, we have:*

$$\begin{aligned}&\big|\Pr[\mathcal{A}(g,g^{\alpha},g^{\beta},g^{\gamma},Z) = 1 \mid (\mathbb{G},\mathbb{G}_T,g,\mathbf{e}) \leftarrow \mathcal{G}_{\mathsf{DBDH}}(1^{\ell});\\ &Z = \mathbf{e}(g,g^{\alpha\beta\gamma}); (\alpha,\beta,\gamma) \leftarrow \mathbb{Z}] -\\ &\Pr[\mathcal{A}(g,g^{\alpha},g^{\beta},g^{\gamma},Z) = 1 \mid (\mathbb{G},\mathbb{G}_T,g,\mathbf{e}) \leftarrow \mathcal{G}_{\mathsf{DBDH}}(1^{\ell})\\ &Z = \mathbf{e}(g,g)^{\mathsf{r}}; (\alpha,\beta,\gamma,\mathsf{r}) \leftarrow \mathbb{Z}]\big| < \mathsf{negl}(\ell)\end{aligned}$$

Symmetric External Diffie-Hellman Problem; (SXDH). The setup algorithm $\mathcal{G}_{\mathsf{SXDH}}(1^{\ell})$ generate a prime order bilinear map, $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2)$. In addition, SXDH assumption states the hardness for PPT adversaries of solving the following problem.

On input $(g_b, g_b^{\alpha}, g_b^{\beta}, Z) \in \mathbb{G}_b^4$, decide whether $Z = \mathbf{e}(g_b, g_b^{\alpha\beta})$ or Z is a random element in $\mathbb{G}_T$ for $b \in 1, 2$. Formally:

Assumption 8. *We say the SXDH [204] assumption holds for group generator $\mathcal{G}_{\mathsf{SXDH}}$ if for all non-uniform polynomial-time adversary $\mathcal{A}$ and $b \in \{1,2\}$. The following two distribution are computationally indistinguishable:*

$$\begin{aligned}&\big|\Pr\,[\mathcal{A}((g_b, g_b^{\alpha}, g_b^{\beta}, Z)) = 1 \mid (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e}, g_1, g_2) \leftarrow \mathcal{G}_{\mathsf{SXDH}}(1^{\ell});\\ &Z = e(g_1, g_2^{\alpha\beta}); (\alpha, \beta) \leftarrow \mathbb{Z}] -\\ &\Pr\,[\mathcal{A}(g_b, g_b^{\alpha}, g_b^{\beta}, Z) = 1 \mid (\mathbb{G}, \mathbb{G}_T, g, \mathbf{e}) \leftarrow \mathcal{G}_{\mathsf{SXDH}}(1^{\ell})\\ &(Z = \mathbf{e}(g_1, g_2)^{\mathsf{r}}); (\alpha, \beta, \mathsf{r}) \leftarrow \mathbb{Z}]\big| < \mathsf{negl}(\ell)\end{aligned}$$

Decisional Subgroup Problem; (DSub). Let $\mathcal{G}_{\mathsf{DSub}}(1^{\ell})$ be a PPT algorithm that generates a bilinear group setting, that takes security parameter ℓ as input and outputs two distinct ℓ-bit prime numbers p and q the descriptions of two groups $\mathbb{G}$ and $\mathbb{G}_T$ of order $n = p \times q$ and a bilinear map $\mathbf{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$:

$$\mathsf{pp}^{\mathsf{DSub}} = (n, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g) \leftarrow \mathcal{G}_{\mathsf{DSub}}(1^{\ell}).$$

The decisional subgroup problem (DSub) introduced by Boneh, Goh and Nissim [51] states that, without knowing the factorization of the group order n, deciding if an element h is in a subgroup of $\mathbb{G}$ or not is hard, namely given $(\mathsf{pp}^{\mathsf{DSub}}, h = g^{\alpha})$ for random integer α it is hard to tell whether $h^q = 1$ or not.

Assumption 9. *The decisional subgroup assumption holds for $\mathcal{G}_{\mathsf{DSub}}$ if for all non-uniform polynomial-time adversary $\mathcal{A}$ we have:*

$$\begin{aligned}&\big|\Pr\,[\mathcal{A}(\mathsf{pp}^{\mathsf{DSub}}, Z = g^{p\alpha}) = 1 \mid \mathsf{pp} \leftarrow \mathcal{G}_{\mathsf{DSub}}(1^{\ell}), \alpha \xleftarrow{\$} \mathbb{Z}]\\ &-\Pr\,[\mathcal{A}(\mathsf{pp}^{\mathsf{DSub}}, Z = g^{\alpha}) = 1 \mid \mathsf{pp} \leftarrow \mathcal{G}_{\mathsf{DSub}}(1^{\ell}), \alpha \xleftarrow{\$} \mathbb{Z}]\big| < \mathsf{negl}(\ell)\end{aligned}$$

2.7 Cryptographic Primitives

This section recalls some cryptographic primitives that we will use later.

2.7.1 Commitment Scheme

Commitment schemes are one of the essential components in many cryptographic protocols. Informally, the commitment scheme is a two-party two-phase protocol between a sender and the receiver. The first phase is known as the commit phase, and the second phase is the opening phase. It is required that the receiver does not gain any knowledge (at least no knowledge of the sender's value) in the commitment phase, even if the receiver tries to cheat. Additionally, the commit phase should "bind" the sender to a unique value which means that in the opening phase, the receiver accept only this value.

Definition 12. ***Commitment Scheme*** *is a tuple of PPT algorithms,* $\langle \mathsf{SetUp}, \mathsf{Commit}, \mathsf{Verify} \rangle$ *including the following steps:*

- ***Set up algorithm**,* $\mathsf{SetUp}(1^\ell)$ *takes as input the security parameter and generates some public parameters* pp *that includes the message space* $\mathcal{M}$*, the commitment space* $\mathcal{C}$*, the opening space* $\mathcal{O}$ *and the random space* $\mathcal{R}$*.*

- *In the **commitment phase**, the sender runs the commitment algorithm on inputs, public parameter,* $m \in \mathcal{M}$ *and* $\mathsf{random} \in \mathcal{R}$ *to generate the commitment-opening pair,* $(\mathsf{com}, \mathsf{d}) \in \mathcal{C} \times \mathcal{O}$*.*

- *In the **opening phase**, the deterministic algorithm* $\mathsf{Verify}(\mathsf{pp}, \mathsf{com}, \mathsf{d}, m)$ *outputs a bit. It outputs* $b = 1$ *if* d *is indeed an opening for* com *respect to the message* m*. Otherwise it reject and outputs* 0*.*

Security Requirements. A commitment scheme requires three security properties: Correctness, hiding and binding properties defined as follows:

- **Correctness:** A commitment scheme is correct if for all $(\mathsf{com}, \mathsf{d}) \leftarrow \mathsf{Commit}(\mathsf{pp}, m; \mathsf{random})$ in which $\mathsf{pp} \leftarrow \mathsf{SetUp}(1^\ell)$, $m \in \mathcal{M}$ and $\mathsf{random} \in \mathcal{R}$, then verify algorithm outputs 1, namely $\mathsf{Verify}(\mathsf{pp}, \mathsf{com}, \mathsf{d}, m) = 1$

- **Hiding:** A commitment scheme has a hiding property if the advantage in experiment 2.2 for any PPT adversary $\mathcal{A}$ would be negligible in terms of the security parameter.

- **Binding:** A commitment scheme is binding if $\mathsf{Succ}_{\mathcal{A}}(1^\ell) = \mathsf{negl}(\ell)$, for any PPT adversary $\mathcal{A}$.

Additional Note. It is worth mentioning that both properties can be characterized as perfect, statistical, or computational properties that define different flavours for commitment schemes; Perfect, statistical and computational [218].

- ***Perfectly hiding*** means that commitments to any two different messages m_0 and m_1 are identical. In contrast, *computationally* (*statistically*) hiding means that commitments of any two distinct messages are computationally (statistically) indistinguishable.

- ***Perfectly binding*** means no unbounded adversary can open a given commitment in two different ways, while computationally binding guarantees that no polynomial-time adversary can open a commitment in two different ways.

It is worth noticing that a commitment scheme can have perfect hiding and computational binding or the opposite. It is, however, impossible to have perfect hiding and perfect binding at the same time.

It is proved that a (computationally hiding, statistically binding and via versa) commitment scheme does exist if One-Way function [151, 203, 209, 148, 49]. A weaker construction comes from pseudorandom permutation [203, 63, 86, 88]. A stronger result shows that "*If there exist non-uniformly One-Way functions, then there exists a bit-commitment scheme for which the secrecy condition also holds with respect to polynomial-size circuits [119].*"

Variants. There are some types of commitment schemes used widely in practice:

- ***Equivocating Commitment Scheme***, which allows for a trapdoor that enables the opening of commitments to arbitrary messages [30].
- ***Homomorphic Commitment Scheme***, which takes the groups $(\mathcal{M},+),(\mathcal{C},\oplus)$ as the message space and commitment space and has an extra PPT algorithm Evaluation with the following property [140]:

$$\text{For } \beta=0,1; m_0,m_1\in\mathcal{M}; (\mathsf{c}_\beta,\mathsf{d}_\beta)\leftarrow \mathsf{Comm}(m_\beta;\mathsf{random}_\beta),\\ \mathsf{c}^*\leftarrow \mathsf{Evaluation}(\mathsf{pp},\mathsf{c}_0,\mathsf{c}_1):\\ \mathsf{Verify}(\mathsf{pp},\mathsf{c}^*,m_0+m_1)=1$$

  This is widely used in SAT problem.
- ***Polynomial Commitment:*** Using a polynomial commitment scheme (see also [170]), Alice may publish a commitment to a polynomial $\mathsf{poly}(x)$ with coefficients in $\mathbb{Z}_p$. If later Bob wants to know $\mathsf{poly}(m)$ for some value m, that is the evaluation of the polynomial at some point; he sends m to Alice, who replies with the claimed evaluation y and a proof that $y=\mathsf{poly}(m)$. The proof guarantees that the claimed evaluation is consistent with the committed polynomial. We require the scheme to be *perfectly binding.*
- ***Concurrent non-Malleable Commitment:*** This is a scheme with the non-malleable property. Namely any PPT adversary $\mathcal{A}$ who has received some commitments, $\mathsf{com}_1=\mathsf{Commit}(m_1),\ldots,\mathsf{com}_n=\mathsf{Commit}(m_n)$, still is unable to generate a new valid commitment c^* for some message m' related to the $m_1,\ldots m_n$ without knowing m_i [218].

Pedersen Commitment Scheme. Our research uses, one of the most well-known commitment schemes, the Pedersen commitment scheme [219], with perfectly hiding and computationally binding property and additively homomorphic. In the Pedersen Scheme, the message space is a cyclic group $\mathbb{G}$ of a prime order p and with two generators (g,h).

Considering a cyclic group $(\mathbb{G},.)$ of prime order p and two generators (g,h), the Pedersen scheme associates the message space, random space, and commitment space with G, $\mathbb{Z}_p$ and $\mathbb{Z}_p$, respectively.

- $\mathsf{Setup}(1^\ell)$ generates the public parameters; $\mathsf{pp}=(p,\mathbb{G},(g,h))$ such that $|p|=\ell$.
- $(\mathsf{c},\mathsf{d})\leftarrow\mathsf{Comm}(\mathsf{pp},m;\mathsf{random})$ such that $\mathsf{c}=g^m\cdot h^{\mathsf{random}}$ and $\mathsf{d}=\mathsf{random}$
- $\mathsf{Verify}(\mathsf{pp},\mathsf{c},\mathsf{d},m)$ accepts the opening if and only if $(g^m\cdot h^{\mathsf{d}}=\mathsf{c})$

2.7.2 Public Key Encryption Schemes

In the conventional method of secure communication, also known as a symmetric key encryption scheme, Alice and Bob use the same secret key for both encryption and decryption algorithms. As a result, they need to share this secret key in advance and keep it secret for as long as they wish to communicate secretly. Diffie and Hellman, in their seminal paper [94], introduced the notion of a public-key encryption scheme, in which, in contrast to symmetric encryption schemes, two distinct keys are used for encryption and the decryption algorithm.

FIGURE 2.2: $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{hiding}}(1^{\ell})$: Commitment Experiment

Challenger steps:		***Adversary steps***
$\mathsf{pp} \leftarrow \mathsf{SetUp}(1^{\ell})$		
	$\xrightarrow{(\mathsf{pp})}$	
		$(m_0, m_1) \leftarrow \mathcal{A}(1^{\ell}, \mathsf{pp})$
	$\xleftarrow{(m_0, m_1)}$	
$\beta \xleftarrow{\$} \{0,1\}$		
$(\mathsf{c}, \mathsf{d}) \leftarrow \mathsf{Commit}(\mathsf{pp}, m_{\beta})$		
	$\xrightarrow{(\mathsf{c})}$	
	$\xleftarrow{(\beta^*)}$	

$$\mathsf{Succ}_{\mathcal{A}}^{\mathsf{hiding}}(1^{\ell}) = \Pr[\,\beta = \beta^*\,], \mathsf{Adv}_{\mathcal{A}}(1^{\ell}) = |\mathsf{Succ}_{\mathcal{A}}(1^{\ell}) - \frac{1}{2}|$$

In a public-key encryption scheme, the public key operates as an encryption key; anyone who knows that public key can encrypt a message. The private key operates as a decryption key; only the owner of the private key can recover the original message. Surprisingly, the encryption key (public key) is useless to an adversary attempting to decipher ciphertexts encrypted with that key. As a result, anyone can publish or broadcast the public key without fear of an eavesdropper for secure communication, which implies that a public-key encryption system allows for private communication without a private channel for key distribution [172].

Definition 13. *A public-key encryption scheme (PKE for short) is a tuple of three probabilistic polynomial-time algorithms* $\Pi^{\mathsf{pke}}(\mathsf{Kgen}, \mathsf{Enc}, \mathsf{Dec})$ *applied to three sets, the keyspace* $\mathcal{K}$*, the message space* $\mathcal{M}$*, and the ciphertext space* $\mathcal{C}$*, such that:*

- $\mathsf{Kgen}(1^{\ell}) \to (\mathsf{PK}, \mathsf{SK})$*: The key generation algorithm takes the security parameter,* ℓ*, as an input and returns a pair of keys* $(\mathsf{PK}, \mathsf{SK})$ *from the keyspace* $\mathcal{K}$*. We refer to the first component,* PK*, as a public key, which defines a message space,* $\mathcal{M}$*, and the second* SK *as private (or secret key). It requires that both keys have a length polynomial in terms of the security parameter.*

- $\mathsf{Enc}(\mathsf{PK}, m) \to \mathsf{ct}$ *: The encryption algorithm, is a probabilistic polynomial-time algorithm that takes as input a message* $m \in \mathcal{M}$ *and the public key and returns a ciphertext* $\mathsf{ct} \in \mathcal{C}$*.*

- $\mathsf{Dec}(\mathsf{SK}, \mathsf{ct}) \to \mathcal{M} \cup \{\perp\}$ *: The decryption algorithm is a deterministic algorithm that takes the ciphertext* ct *and the secret key* SK *as inputs and returns the message* m' *from the message space or* $\perp$ *as denoting failure.*

Security Requirements. A public-key encryption scheme requires to have the following properties:

- ***Correctness:*** The output of the decryption algorithm is the original message except with negligible probability over the randomness of key-generation and encryption algorithms:

- ***Security:*** When it comes to security in the context of a public-key encryption scheme, we need to discuss the "security guarantee" and "the adversarial model," namely, the adversary's power. Then we can have a precise definition for different flavour of the security notions such as adaptive versus non-adaptive, chosen-plaintext vs chosen-ciphertext attacks.

The basic notion of security for a PKE is semantic security which says, no PPT adversary, given two messages m_0 and m_1, and a ciphertext ct, can guess ct is encryption of m_0 or m_1 better than a random guess.

Semantic security is the most basic security concept in a public-key system, which states that an adversary who chooses messages m_0 and m_1, and receives the ciphertext ct cannot distinguish whether ct is the encryption of m_0 or m_1. Therefore, considering the adversarial power, we have the following security notions [31]:

- **IND-CPA Security against an adaptive adversary:** Adaptive adversary chooses messages m_0 and m_1 after receiving the public-key of the scheme. A public-key encryption scheme is indistinguishable against chosen plaintext attacks (IND-CPA for short) if the advantage of any PPT adversary in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{cpa-A}}(1^{\ell})$ in Figure 2.3 would be negligible in terms of the security parameter.

- **IND-CPA Security against a non-adaptive adversary** guarantees a weaker notion of security since a non-adaptive adversary should choose messages m_0 and m_1 before receiving the public-key of the scheme. A public-key encryption scheme is indistinguishable against chosen plaintext attacks (IND-CPA for short) if the advantage of any PPT adversary in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{cpa-nA}}(1^{\ell})$ would be negligible in terms of the security parameter, Figure 2.3.

  Note that the only difference between $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{cpa-A}}(1^{\ell})$ and $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{cpa-nA}}(1^{\ell})$ is the sequence in which steps 1 and 2 are carried out.

- **IND-CCA Security against an adaptive adversary** ensures a high level of security in which the adversary has access to a decryption oracle. It means that the adversary can query some string ct to the decryption oracle and receive some message m such that $\mathsf{Dec}(\mathsf{ct}) = m$ or $\perp$. An adaptive adversary can query after determining the message m_0 and m_1, whereas non-adaptive adversaries can only access to the decryption oracle before selecting the messages. A public-key encryption scheme is indistinguishable against chosen plaintext attacks (IND-CPA for short) if the advantage of any PPT adversary in an experiment in Figure 2.4 would be negligible in terms of the security parameter.

- **IND-CCA Security against a non-adaptive adversary.** A public-key encryption scheme is indistinguishable against chosen plaintext attacks (IND-CPA for short) if the advantage of any PPT adversary in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{cca-nA}}(1^{\ell})$ would be negligible in terms of the security parameter, Figure 2.4.

- **Non-Malleable Encryption Scheme.** A public-key encryption scheme is non-malleable if the advantage of any PPT adversary in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{nm-cpa}}(1^{\ell})$ would be negligible in terms

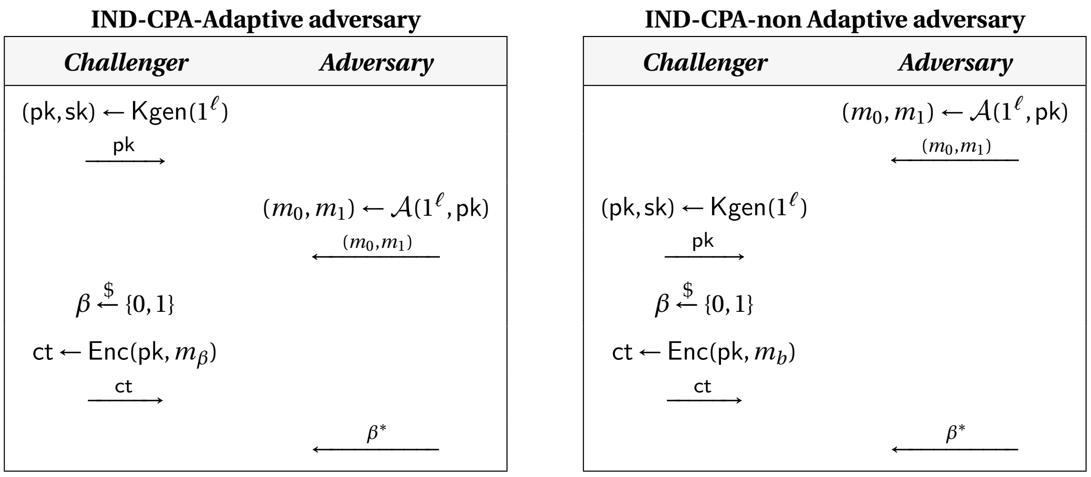


FIGURE 2.3: Experiments for CPA security

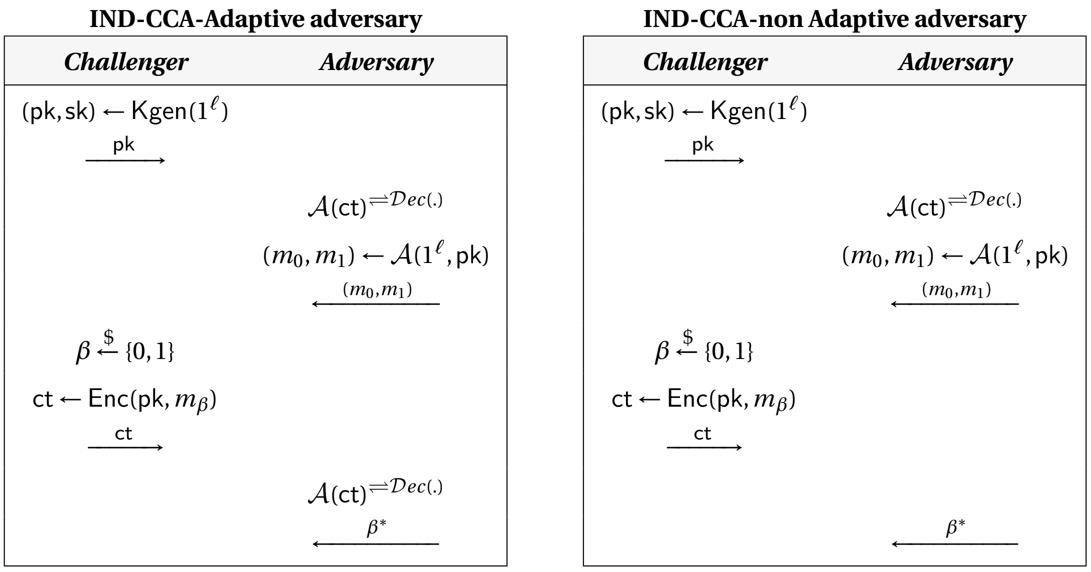


FIGURE 2.4: Experiments for CCA security

of the security parameter, Figure 2.5.

$$\mathsf{Adv}^{\mathsf{nm-cpa}}_{\mathcal{A}}(\ell) = \Bigg|\Pr\left[\mathcal{A}\big((\mathsf{Enc}(m_0),\pi_{\mathsf{enc}})\big) \mapsto 0 \;\Bigg|\; \begin{array}{r}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Kgen}(1^\ell)\\ (m_0,m_1) \leftarrow \mathcal{A}(1^\ell,\mathsf{pk})\\ \mathcal{A}(\mathsf{pk})^{\rightleftharpoons \mathcal{D}ec(.)}\end{array}\right] - \Pr\left[\mathcal{A}\big((\mathsf{Enc}(m_1),\pi_{\mathsf{enc}})\big) \mapsto 0 \;\Bigg|\; \begin{array}{r}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Kgen}(1^\ell)\\ (m_0,m_1) \leftarrow \mathcal{A}(1^\ell,\mathsf{pk})\\ \mathcal{A}(\mathsf{pk})^{\rightleftharpoons \mathcal{D}ec(.)}\end{array}\right]\Bigg| < \mathsf{negl}(\ell) \tag{2.2}$$

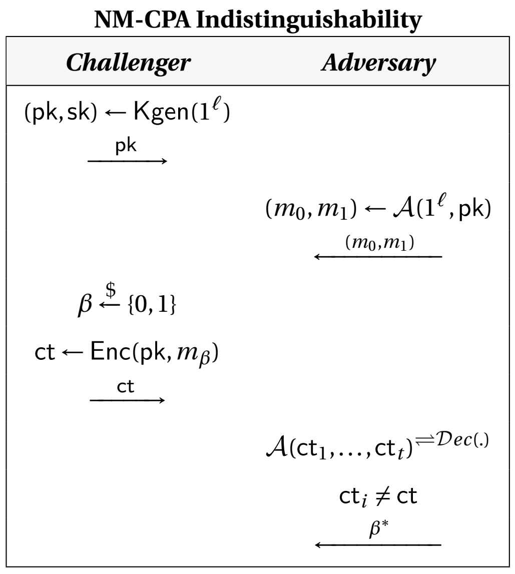

NM-CPA Indistinguishability
Challenger
Adversary
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Kgen}(1^\ell)$
$\xrightarrow{\mathsf{pk}}$
$(m_0, m_1) \leftarrow \mathcal{A}(1^\ell, \mathsf{pk})$
$\xleftarrow{(m_0, m_1)}$
$\beta \xleftarrow{\$} \{0,1\}$
$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_\beta)$
$\xrightarrow{\mathsf{ct}}$
$\mathcal{A}(\mathsf{ct}_1, \dots, \mathsf{ct}_t)^{\Rightarrow \mathcal{D}ec(.)}$
$\mathsf{ct}_i \neq \mathsf{ct}$
$\xleftarrow{\beta^*}$

FIGURE 2.5: Experiments for Non-Malleable CPA security

Through our research we use some public-key encryption schemes. Here we recall these schemes and their security properties.

2.7.2.1 ElGamal Encryption Scheme

In 1985, ElGamal [112] introduced a practical and powerful (partially) homomorphic encryption scheme which has a IND-CPA security based on the hardness of decisional Diffie-Hellman assumption 5. The original version of ElGamal is defined on the multiplicative group $\mathbb{Z}_p^*$ for some prime number p. Still, other variants of the scheme have been proposed, instantiated on different groups in which the discrete logarithm assumption holds, such as ElGamal in elliptic curve group [174] and CRT-ElGamal in a subgroup of $\mathbb{Z}_n^*$ where n is an RSA-number [156], ElGamal adaptations in Class Groups [67, 66, 64]. ElGamal in a hashed version [65]. We refer to [72] for more information about the ElGamal Encryption scheme. Here we recall the original version of the ElGamal encryption scheme [112].

ElGamal encryption scheme. Consider the group generator algorithm that on input the security parameter, output a tuple $\mathcal{G}$ of a prime number p with $|p| = \ell$, a cyclic group $\mathbb{G}$ with generator g:

$$\mathcal{G} = (p, \mathbb{G}, g) \leftarrow \mathsf{GroupGen}(1^\ell).$$

Then the ElGamal PKE scheme is composed of the following algorithms for the message space $\mathcal{M} = \mathbb{G}$:

- **Key Generation:**
  1. Run $\mathsf{GroupGen}(1^\ell)$ to generate $\mathcal{G} = (p, \mathbb{G}, g)$.
  2. Select a random integer x from $\mathbb{Z}_p^*$.
  3. Set $h = g^x$.
  4. Set $\mathsf{pk}_{\mathsf{EG}} = (\mathbb{G}, p, g, h)$ and $\mathsf{sk}_{\mathsf{EG}} = x$.
  5. Return $\mathsf{Key}_{\mathsf{EG}} = (\mathsf{pk}_{\mathsf{EG}}, \mathsf{sk}_{\mathsf{EG}})$.
- **Encryption:** For message $m \in \mathbb{G}$:

1. Select a random number $\mathsf{random} \in \mathbb{Z}_p^*$.
2. Set the ciphertext $\mathsf{ct} = (g^{\mathsf{random}}, h^{\mathsf{random}} \cdot m)$.
3. Return ct.

- **Decryption:** For ciphertext $\mathsf{ct} = (\alpha, \beta) \in \mathbb{G} \times \mathbb{G}$:

1. Compute $m' = \frac{\beta}{\alpha^x}$.
2. Return m'.

The security of the system is the result of the following theorem.

Theorem 2.7.1. *The semantic security of the ElGamal encryption is equivalent to the decision Diffie-Hellman problem. [247]*

Homomorphic Property. Consider two ciphertexts $\mathsf{ct}_1 = \mathsf{Enc}(m_1)$ and $\mathsf{ct}_2 = \mathsf{Enc}(m_2)$ for some (unknown) randomness $\mathsf{r}_1, \mathsf{r}_2$:

$$\begin{cases} \mathsf{ct}_1 = (\alpha_1, \beta_1) = (g^{\mathsf{r}_1}, h^{\mathsf{r}_1} g^{m_1}), \\ \mathsf{ct}_2 = (\alpha_2, \beta_2) = (g^{\mathsf{r}_2}, h^{\mathsf{r}_2} g^{m_2}), \end{cases} \Longrightarrow u_1 \times u_2 = g^{\mathsf{r}_1} \times g^{\mathsf{r}_2}, \beta_1 \times \beta_2 = h^{\mathsf{r}_1} \cdot m_1 \times h^{\mathsf{r}_2} \cdot m_2,$$

$$\alpha_1 \times \alpha_2 = g^{\mathsf{r}_1 + \mathsf{r}_2}, \beta_1 \times \beta_2 = h^{\mathsf{r}_1 + \mathsf{r}_2} \cdot (m_1 \cdot m_2)$$
$$\mathsf{ct}_1 \times \mathsf{ct}_2 = \mathsf{Enc}(m_1 + m_2)$$

re-Encryption Property. Additionally, ElGamal scheme, allows reEncrypting a ciphertext without knowledge of the message and the randomness:

$$\mathsf{ct} = \mathsf{Enc}(m; \mathsf{random}) = (\alpha, \beta) = (g^{\mathsf{random}}, h^{\mathsf{random}} \cdot m)$$
$$\Longrightarrow \mathsf{ct}^* = (\alpha \times g^{\mathsf{r}}, \beta \times h^{\mathsf{r}}) = (g^{\mathsf{random}+\mathsf{r}}, h^{\mathsf{random}+\mathsf{r}} \cdot m) = \mathsf{Enc}(m; \mathsf{random} + \mathsf{r})$$

2.7.2.2 Paillier Encryption Scheme

Paillier public-key cryptosystem is a partially homomorphic encryption scheme, and it has IND-CPA security based on the hardness of the decisional composite residuosity assumption 3. The message space in Paillier encryption is associated with $\mathbb{Z}_n^*$ where n is an RSA-number, $n = p \cdot q; (p, q, e) \leftarrow \mathcal{G}^{\mathsf{RSA}}(1^\ell)$ and the cipher text space $\mathcal{C}$ is $\mathbb{Z}_{n^2}^*$. Consider two functions $\lambda(n) = \mathsf{lcm}(p-1, q-1)$ and $Ł(x) = \frac{x-1}{n}$, Paillier encryption scheme is described as follows:

- **Key Generation:**

1. Run $\mathcal{G}^{\mathsf{RSA}}(1^\ell)$ to generate two prime numbers p and q.
2. Set $n = p \cdot q$.
3. Select a random integer (generator) from $\mathbb{Z}_{n^2}^*$.
4. Set $\mu = \left(Ł(g^{\lambda(n)}[\mod n^2])\right)^{-1}[\mod n]$.
5. Set $\mathsf{pk}_{\mathsf{paillier}} = (n, g)$ and $\mathsf{sk}_{\mathsf{paillier}} = (\lambda(n), \mu)$.
6. Return $\mathsf{Key}_{\mathsf{paillier}} = (\mathsf{pk}_{\mathsf{paillier}}, \mathsf{sk}_{\mathsf{paillier}})$.

- **Encryption:** For message $m \in \mathbb{Z}_n$:
  1. Select a random number $\mathsf{random} \in \mathbb{Z}_n^*$.
  2. Set the ciphertext $\mathsf{ct} = g^m \cdot \mathsf{random}^n [\mod n^2]$.
  3. Return ct.
- **Decryption:** For ciphertext $\mathsf{ct} \in \mathbb{Z}_{n^2}^*$:
  1. Compute $m' = \frac{Ł(\mathsf{ct}^{\lambda(n)}[\mod n^2])}{Ł(g^{\lambda(n)}[\mod n^2])}[\mod n]$.
  2. Return m'

The security of the system is the result of the following two theorems [216]:

Theorem 2.7.2. *Paillier encryption scheme is one-way if and only if the Computational Composite Residuosity assumption holds.*

Theorem 2.7.3. *Paillier is semantically secure, (**IND-CPA**) if and only if the Decisional Composite Residuosity assumption holds.*

Homomorphic property. Consider two ciphertexts $\mathsf{ct}_1 = \mathsf{Enc}(m_1)$ and $\mathsf{ct}_2 = \mathsf{Enc}(m_2)$ with two (unknown) random numbers $\mathsf{r}_1, \mathsf{r}_2$:

$$\begin{cases} \mathsf{ct}_1 = g^{m_1}\mathsf{r}_1^n[\mod n^2], \\ \mathsf{ct}_1 = g^{m_1}\mathsf{r}_1^n[\mod n^2], \end{cases} \Longrightarrow \mathsf{ct}_1 \times \mathsf{ct}_2 = g^{m_1}\mathsf{r}_1^n \times g^{m_1}\mathsf{r}_1^n[\mod n^2] = g^{m_1+m_2}(\mathsf{r}_1\mathsf{r}_2)^n \mod n^2$$
$$\mathsf{ct}_1 \times \mathsf{ct}_2 = g^{(m_1+m_2)}(\mathsf{random})^n = \mathsf{Enc}(m_1 + m_2) \tag{2.3}$$

re-Encryption property. Additionally, the Paillier scheme, allows re-Encrypting a ciphertext without knowledge of the message and its randomness:

$$\begin{aligned} \mathsf{ct} = \mathsf{Enc}(m; \mathsf{random}) = g^m \mathsf{random}^n [\mod n^2] \\ \Longrightarrow \mathsf{ct}^* = \mathsf{ct} \times \mathsf{r}^n = g^m(\mathsf{random} \times \mathsf{r})^n[\mod n^2] = \mathsf{Enc}(m; \mathsf{random} + \mathsf{r}) \end{aligned} \tag{2.4}$$

Multi-Party Computation comparison. One of the interesting properties of Paillier encryption scheme that we use in our research is that it allows an efficient multi-party computation protocol to compare and hence sort ciphertexts by plaintext values without decryption [192]. Furthermore, this algorithm is linear in the bit lengths, i.e., logarithmic in the security parameter, and can be made public verifiable [184].

2.7.2.3 BBS Linear Encryption Scheme

Boneh *et al.* presented a linear encryption (LE) scheme based on the Decision Linear assumption in [51]. The BBS scheme has the following algorithms:

- **Key Generation:**
  1. Run $\mathsf{GroupGen}(1^\ell)$ to generate $\mathcal{G} = (p, \mathbb{G}, g)$.
  2. Choose generators $u, v, h \in \mathbb{G}$ and integers $x, y \in \mathbb{Z}_p$ such that $u^x = v^y = h$.
  3. Set $\mathsf{pk}_{\mathsf{BBS}} = (\mathbb{G}, u, v, h)$ and $\mathsf{sk}_{\mathsf{BBS}} = (x, y)$.

4. Return $\mathsf{Key}_{\mathsf{BBS}} = (\mathsf{pk}_{\mathsf{BBS}}, \mathsf{sk}_{\mathsf{BBS}})$.

- **Encryption:** For message $m \in \mathbb{G}$:
  1. Select random numbers $\mathsf{r}_1, \mathsf{r}_2 \in \mathbb{Z}_p$.
  2. Set the ciphertext $\mathsf{CT} = (u^{\mathsf{r}_1}, v^{\mathsf{r}_2}, m \cdot h^{\mathsf{r}_1+\mathsf{r}2})$.
  3. Return CT.
- **Decryption:** For ciphertext $\mathsf{CT} = (\alpha, \beta, \gamma) \in \mathbb{G}^3$:
  1. Compute $m' = \frac{\gamma}{\alpha^x \cdot \beta^y}$.
  2. Return m'

The BBS is semantically secure against a chosen-plaintext attack, assuming Decision Linear assumption (see 6) holds in group $\mathbb{G}$ [51].

The security of the system is the result of the following theorem [216]:

Theorem 2.7.4. *The semantic security of the linear encryption BBS is the result of the decision Linear problem.*

2.7.3 Hash Functions

As mentioned in 2.4, random oracles do not exist in the real world; hence a realistic and practical instantiation of the Random Oracle Model is required. In modern cryptography, the random oracle is instantiated with a particular cryptographic primitive known as a hash function which maps an arbitrary finite-length string to a fixed-length string. In practice, hash functions are commonly employed for protocol verification, such as message integrity verification. Another application is, converting an interactive protocol into a non-interactive variant, such as the Schnorr proof system [233](a non-interactive version of the Sigma protocol). Merkle-Damgard is the most well-known high-level structure for hash functions used in constructing SHA family hash functions [82, 157].

There are two types of hash functions: the first one that depends only on the bit-string and the public parameters, known as ***unkeyed hash functions***, and the second type, ***keyed hash function*** that requires additional input, a key, to generate the hash value [200].

Unkeyed Hash Function. From a structural perspective, the subclass of unkeyed hash functions, also known as modification detection codes (**MDC**s), can be classified according to the nature of the operations that comprise their internal compression functions. As a result, we can classify iterated hash functions into three major categories: block cipher-based hash functions, customized hash functions, and hash functions based on modular arithmetic.

Keyed Hash Function. One of the well-known methods for the keyed hash function is the Message Authentication Code (**MAC**) which is explicitly designed for message authentication. However, prior to 1995, relatively few MAC algorithms were proposed compared to the enormous number of MDC algorithms, possibly because the original ideas, which were extensively implemented in practice, were appropriate.

Definition 14 (Collision-Resistant Hash Function [200]). *A collision-resistance hash function family is a tuple of PPT algorithms* $\Pi^{\mathsf{hash}} = \langle \mathsf{Setup}, \mathsf{hash} \rangle$ *domain space* $\mathcal{D} = \{0,1\}^*$.

- ***Set up*** *is a PPT algorithm that takes as input the security parameter and generates public parameter and the private hash key,* $(\mathsf{pp}, \mathsf{key}_{\mathsf{hash}}) \leftarrow \mathsf{Setup}(1^{\ell})$. *The public parameter also defines the domain space,* $\mathcal{D}$ *as a subset of* $\{0,1\}^{\ell}$.

- ***The hash*** *is a deterministic PT algorithm that on inputs public parameter, hash key and some string* $s \in \{0,1\}^*$, *outputs the hash value* $\mathsf{h} \in \mathcal{D}_{\mathsf{hash}}$.

Security Requirements. Following from [200] a cryptographic hash function requires to have the following properties:

- **Preimage resistance** states that it is computationally infeasible to find any input that hashes to a any pre-specified output. Formally the following success probability is negligible for any PPT adversary $\mathcal{A}$.

$$\mathsf{Succ}^{\mathsf{pre-image}}_{\mathsf{hash},\mathcal{A}}(\ell) = \Pr\left[\mathsf{hash}(x) = \mathsf{h} \mid \begin{array}{c}(\mathsf{pp},\mathsf{key}) \leftarrow \mathsf{Setup}^{\mathsf{hash}}(1^{\ell}),\\ x \leftarrow \mathcal{A}(1^{\ell},\mathsf{pp},\mathsf{key}_{\mathsf{hash}},\mathsf{h})\end{array}\right]$$
$$< \mathsf{negl}(\ell)$$

- **Second preimage resistance** states that it is computationally infeasible to find any second input which the same output as any specified input, $\mathsf{hash}(x) = \mathsf{hash}(y)$. Formally the following success probability is negligible for any PPT adversary $\mathcal{A}$:

$$\mathsf{Succ}^{\mathsf{2-pre-image}}_{\mathsf{hash},\mathcal{A}}(\ell) = \Pr\left[\begin{array}{c}\mathsf{hash}(x) = \mathsf{hash}(y),\\ x \neq y\end{array} \mid \begin{array}{c}(\mathsf{pp},\mathsf{key}) \leftarrow \mathsf{Setup}^{\mathsf{hash}}(1^{\ell}),\\ (x,y) \leftarrow \mathcal{A}(1^{\ell},\mathsf{pp},\mathsf{key}_{\mathsf{hash}})\end{array}\right]$$
$$< \mathsf{negl}(\ell)$$

- **Collision resistance** states that even with a free choice of inputs, finding two distinct inputs x and y with the same hash-value is computationally infeasible. Formally the following success probability is negligible for any PPT adversary $\mathcal{A}$.

$$\mathsf{Succ}^{\mathsf{collision}}_{\mathsf{hash},\mathcal{A}}(\ell) = \Pr\left[\begin{array}{c}\mathsf{hash}(x) = \mathsf{hash}(y),\\ x \neq y\end{array} \mid \begin{array}{c}(\mathsf{pp},\mathsf{key}) \leftarrow \mathsf{Setup}^{\mathsf{hash}}(1^{\ell}),\\ (x,y) \leftarrow \mathcal{A}(1^{\ell},\mathsf{pp},\mathsf{key}_{\mathsf{hash}})\end{array}\right]$$
$$< \mathsf{negl}(\ell)$$

Additional Note. We note that Collision resistance implies second-preimage resistance of hash functions, but it does not guarantee preimage resistance.

2.7.4 Signature Schemes

Definition 15. *A signature scheme with message space* $\mathcal{M}$ *is a tuple of PPT algorithms* $\Pi^{\mathsf{signature}} = \langle \mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify} \rangle$ *where:*

- ***Set up*** *is a probabilistic polynomial algorithm that takes as input the security parameter and generates public parameter* $(\mathsf{pp}) \leftarrow \Pi^{\mathsf{sign}}.\mathsf{Setup}(1^{\ell})$. *The public parameter also defines the message space* $\mathcal{M}$ *and rang space* $\mathcal{D}$. *We exclude the* pp *from the algorithm's inputs while it is implicitly included to avoid heavy notation.*

- ***Key generation*** *is a probabilistic algorithm that on input security parameter and the public parameters, outputs a pair of matching keys; The private key is used for signing and the public verification key.* $(\mathsf{sk}_{\mathsf{sign}}, \mathsf{pk}_{\mathsf{verify}}) \leftarrow \mathsf{Kgen}(1^{\ell})$.

- ***Signature*** *is also a probabilistic polynomial algorithm that takes as input a message* $m \in \mathcal{M}$ *and the private signing key* $\mathsf{sk_{sign}}$ *and generates a signature* σ
- ***Verify*** *is a deterministic algorithm that takes the verification key, some message* $m \in \mathcal{M}$ *and its associated signature* σ_m *and output 1 or 0.*

Security Requirements. The basic notion of security for a signature scheme is unforgeability, which states that no PPT adversary can generate a valid signature for message m without having the private signing key, $\mathsf{sk_{sign}}$. However, when we take into account the adversary power, more profound notions of unforgeability emerge [50]:

- **Unforgeable under chosen message attacks against an adaptive adversary versus non-adaptive:** In unforgeable security notion, the adversary has access to a signing oracle; $\mathcal{A}(.)^{\rightleftharpoons \mathcal{S}ign(\mathsf{sk_{sign}})}$. It means that the adversary can query some message $m_1, \ldots, m_q$ to the signing oracle and receive the valid signatures; $\{(m_i, \sigma_i)\}_{i \in [q]}$. An ***adaptive*** adversary can query after receiving the verification key, pk_{verify} while a ***non-adaptive*** adversary is forced to output messages $m_1, \ldots m_q$ it wants to see signed, before obtaining the verification key. A signature scheme is existential unforgeable under chosen-message attacks adaptive (non-adaptive) , $\mathsf{UF-cma}$ if the success probability of any PPT adversary $\mathcal{A}$ in experiment 2.6 is negligible.
- **Strongly unforgeable versus unforgeable under chosen message attacks:** The term ***strongly*** unforgeable refers to an adversary winning if it outputs a message with a valid signature such that the pair (m, σ_m) did not appear in the query phase. In contrast, unforgeable indicates that the message m should not appear during the query phase. A signature scheme is Strongly existential unforgeable under chosen-message attacks, sUFcma if the success probability of any PPT adversary $\mathcal{A}$ in experiment 2.6 is negligible.
- **Existential unforgeability notions regarding q-bounded chosen-message-attack:** In addition to the previous notions, this weaker security notion, for signatures scheme, is presented in [50], which is the same security notions above, except that the adversary is restricted to at most q signature queries.

FIGURE 2.6: $\mathsf{Exp}^{\mathsf{sign}}_{\mathcal{A}}(1^\ell)$: Signature Experiment

Challenger	***Adversary***
$\mathsf{pp^{sign}Setup}(1^\ell)$	
$(\mathsf{sk_{sign}}, \mathsf{pk_{verify}}) \leftarrow \mathsf{Kgen}(1^\ell)$	
	$\mathcal{A}(m_i)^{\rightleftharpoons \mathcal{S}ign(.)}$
	$(m^*, \sigma) \leftarrow \mathcal{A}(1^\ell, \{(m_i, \sigma_i)\})$

Success Probability:

$$\mathsf{Succ}^{\mathsf{sign}}_{\mathcal{A}}(1^\ell) = \Pr\left[\mathsf{Verify}(\mathsf{pk_{sign}}, m^*, \sigma^*) = 1\right] \begin{cases} (m^*, \sigma^*) \notin (m_i, \sigma_i) : \text{ for EUF} \\ (m^*) \notin m_i : \text{ for strong EUF} \end{cases}$$

Chapter 3

A Brief Survey on Zero-Knowledge Proof Systems

“ The purpose of life is to conjecture and prove”
Paul Erdos

If you have heard the story of Alibaba and the forty Baghdad thieves, you may be aware that there was a cave and a treasure. And the cave’s door was sealed with a large stone, and the only way to enter was to know the secret phrase ***“Open Sesame!”***

And Alibaba was aware of the word. That is why the thief master ordered him to reveal the word! A tricky situation! Indeed, it did not seem wise for Alibaba to reveal the word because it is most likely the thieves would kill him after that. But on the other hand, if he did not reveal the word, they might have believed he did not know it in the first place, and he would be killed again.

So what could he do not to die!?

We have heard more tales of Alibaba after his adventures with the forty thieves of Baghdad, so we are guessing he went with the third solution. He had convinced them that he was aware of the ***“Open Sesame!”*** without telling the ***“Open Sesame!”*** However, how did he accomplish this?

We believe he was clever enough to use a ***“Zero-Knowledge proof System”***. A beautiful concept that conceals a contradiction at its core that interwoven the notions of clarity and mystery!

Contents

A Zero-Knowledge proof system is one of the fascinating tools in cryptography. However, there is a contradiction hidden within the concept of Zero-Knowledge; while proof should be convincing, it must yield no knowledge beyond the validity of the statement being proven. In other words, obtaining Zero-Knowledge proof that a statement is true is equivalent to being told by a trusted party that the statement is true.

Zero-Knowledge proof is introduced in the seminal work of Goldwasser, Micali, and Rackoff [128], and became one of the essential underlying primitives in cryptography. According to Goldreich, Micali, and Wigderson [125], the Zero-Knowledge proof is an innovative technique to force involved parties in a protocol to adhere to it while assuring that no secret information is leaked.

Zero-Knowledge is one of the essential ingredients we employ towards having verifiable, secure computations. This section, will gather basic terminology and background knowledge we use in our research.

3.1 Zero-Knowledge proof Systems

Recall the definition of an Interactive proof system in which two parties, the *prover* in charge of the algorithm Prove and the *verifier* in charge of the algorithm, Verify, interact with each other on some common input $(x, \mathscr{L})$. Prover within an n-round interaction, tries to convince the verifier that $x \in \mathscr{L}$. We motivate a Zero-Knowledge system as an interactive proof system by which the verifier gains "no knowledge" beyond the statement's validity. Before going to the formal definition, it is necessary to clarify the meaning of ***"gaining no knowledge"***.

Information versus Knowledge. To begin, we want to emphasize that while knowledge (as stated below) and information (from an information theory perspective) are often used interchangeably, we distinguish between the two. Then to better understanding the concept, we will look into interactive Zero-Knowledge proof systems for language L_1 and L_2 (See 2.3.1).

Consider the first scenario for language, $\mathscr{L}_1$. When Bob asks Alice if graph G is Eulerian or not, whatever Alice tells Bob about the Eulerian path, Bob could have easily obtained it by running some linear time algorithm. Therefore, here we say Bob does not gain any knowledge in this interaction. On the other hand, in the interactive game for language $\mathscr{L}_2$, Hamiltonian graph, if Alice proves to Bob that G has a Hamiltonian cycle, that would be the knowledge that Bob gains in the game. That is because Bob does not have an efficient algorithm to recognize the "Hamiltonian Graph" by himself. In fact, Bob gains knowledge only if he receives some outcome from the interaction with Alice that is infeasible for him to compute. To summarise according to [119], knowledge is tied to computational difficulty; something is knowledge if it can be computed by an efficient algorithm given limitless processing resources, whereas information is not.

Based on the above discussion, informally, we can define Zero-Knowledge property as follow: We say an interactive proof system has a Zero-Knowledge property if what can be computed by an arbitrary feasible adversary (e.g., a verifier) from the interactive game on input x can be computed by an arbitrary feasible algorithm that is only given the input x.

Definition 16. *[**Interactive Zero-Knowledge Proof System**]An n-round interactive Zero-Knowledge proof system, for the language $\mathscr{L}$ is a protocol between prover* (Prove) *and a PPT verifier* (Verify) *with the following properties:*

- **Completeness:** *For very $x \in \mathscr{L}$, the verifier always outputs 1 (accept the proof) after interacting with the prover:*

$$\forall x \in \mathscr{L} : \Pr\Big[\,[\mathsf{Verify}^{\rightleftharpoons \mathsf{Prove}}(x)] \rightarrow 1\Big] = 1$$

- **Soundness:** *For any $x \notin \mathscr{L}$, and any potential cheater prover, the verifier output* 0 *(reject the proof) with overwhelming probability:*

$$\forall x \notin \mathscr{L} : \Pr\Big[\,[\mathsf{Verify}^{\rightleftharpoons \mathsf{Prove}^*}(x)] \rightarrow 1\Big] = \mathsf{negl}(|x|)$$

- **Zero-Knowledge:** *For any $x \in \mathscr{L}$ there exist a PPT* simulator *algorithm,* Sim, *such that the following two distributions are identical:*

$$\Big\{\mathcal{V}iew[\mathsf{Verify}^{\mathsf{Prove}}](x)\Big\}_{x \in \mathscr{L}} \equiv \Big\{\langle \mathsf{Sim} \rangle (x)\Big\}_{x \in \mathscr{L}}$$

Additional Note. According to the above definition, the zero-knowledge property requires the existence of an algorithm simulating the view of any verifier. Honest-verifier zero-knowledge proof system is a weaker concept that only requires the existence of a simulator for a single verifier, which is the honest verifier specified in the protocol specification.

Additional Note. To evaluate the robustness of the Zero-Knowledge proof system's definition, that is, to determine if it is too weak or too strong, we first note that every programming language has a trivial proof. On the other hand, triple theorems demonstrate why each property in the zk definition is required:

Theorem 3.1.1 ([119])**.** *Suppose that $\mathscr{L}$ has a unidirectional Zero-Knowledge proof system, then $\mathscr{L} \in$* ***BPP****.*

Theorem 3.1.2 ([119])**.** *Suppose that $\mathscr{L}$ has a Zero-Knowledge proof system in which the verifier program is deterministic, then $\mathscr{L} \in$* ***BPP****.*

Theorem 3.1.3 ([119])**.** *Suppose that $\mathscr{L}$ has an auxiliary-input Zero-Knowledge proof system in which the prover program is deterministic, then $\mathscr{L} \in$* ***BPP****.*

Negative results. To analyze the upper bound for **IP** we first consider another type of **IP** the so-called *Arthur-Miller* game, which is a simplified version of a 3-round interactive proof system with the following steps:

1. The verifier sends a random string to the prover.
2. The prover responds with some string.
3. Based on a deterministic computation, the verifier on common inputs and two strings deterministically accept or reject the proof .

By **AM** we refer to the class of language $\mathscr{L}$ that can be recognized by an Arthur-miller game.It is proved that if an arbitrary language $\mathscr{L} \in$ **coNP** has an Arture-Miller proof system (**coNP** $\subset$ **AM**), then it would be unlikely that the polynomial-time hierarchy would collapse. Hence with the help of the following theorem we can define an upper bound for the class **ZKP**.

Additional Note. It is believed that **coNP** in not contained in **AM** (**NP** is not contained in **coAM**). In fact if **coNP** $\subseteq$ **AM** then polynomial-time hierarchy would collapse which is unlikely. On the other hand we have the following theorem:

Theorem 3.1.4 ([119])**.** *If there exists a statistical (almost-perfect) Zero-Knowledge proof system for a language $\mathscr{L}$, then $\mathscr{L} \in$* ***coAM****. In fact $\mathscr{L} \in$* ***coAM*** *$\cup$* ***AM***

Therefore, we conclude that if some **NP**-complete language $\mathscr{L}^*$ has a statistical Zero-Knowledge proof system, then it implies every language in **NP** has the statistical Zero-Knowledge proof system:

$$\begin{aligned}\exists \mathscr{L}^{\text{complete}} \in \mathbf{SZK} &\xRightarrow{\mathscr{L} < \mathscr{L}^{\text{complete}}} (\forall \mathscr{L} \in \mathbf{NP} : \mathscr{L} \in \mathbf{SZK}) \\ &\Longrightarrow \forall \mathscr{L} \in \mathbf{NP} : \mathscr{L} \in \mathbf{coAM} \\ &\Longrightarrow \mathbf{NP} \subset \mathbf{coAM}\end{aligned}$$

The above argument shows that there exists some language that they do not possess perfect Zero-Knowledge proof system.

3.2 Zero-Knowledge Proof System for NP-language

We now need to consider the following key question:

Do Zero-Knowledge proof systems exist? Additionally, assuming this is the case, for which languages may we have a zero proof system?

The brief answer is *"yes"* to the first question. Furthermore, it is clear from the definitions of the complexity classes **P** and **BPP** that we have a trivial Zero-Knowledge proof system for each language in these classes. Because for the languages in **BPP** class, any PPT verifier can recognize the language by itself. Thus, the interesting question is:

For which languages is a non-trivial zero proof system possible?

Intensive and fascinating research has been conducted to answer this question positively, assuming that a one-way function does exist.

On the other hand, it is shown in [215] that unless very weak one-way functions exist, Zero-Knowledge proofs can be given only for languages in **BPP**, which establish the necessity of the one-way function for non-trivial ZK.

To demonstrate that ZK exists for all NP-languages, the authors in [124] construct a ZK for some **NP**-complete language (such as, Graph coloring or Hamiltonian Graph languages) and then conclude that ZK exists for all NP-languages using the Karp reduction. Here is a brief summary of the proof:

FIGURE 3.1: The Zero-Knowledge proof system for Graph 3-Colouring [119]

• Setting:	$\mathcal{G} = (G, V, E, n) : V = \{1, \dots, n\}, E = \left\{(i, j) : \text{ vertix } i \text{ connect to } j\right\}$
	$\mathrm{R}_g = \left\{(x, w) : x = \mathcal{G}, w = \phi : V \mapsto \left\{\text{blue}, \text{red}, \text{green}\right\}\right.$
	$(i, j) \in E \implies \phi(i) \neq \phi(j)$
• Inputs:	Prover x, w, Verifier: x
Prover commitment:	
	1. Pick a random permutation σ over $\left\{\text{blue}, \text{red}, \text{green}\right\}$
	2. For $i = 1, \dots n$ commitment to the value $\sigma(\phi(i))$
	3. Compute $\mathsf{Com} = \mathsf{Com}(\sigma(\phi(i)))$
	Prover $\xrightarrow{\mathsf{Com}}$ Verifier
Verifier challenge	
	Pick $e = (i, j) \overset{\$}{\leftarrow} E$
	Verifier $\xrightarrow{e}$ Prover
Prover respond:	
	Decommit to i and j by sending $\sigma(\phi(i)), \sigma(\phi(j))$
	Prover $\xrightarrow{z=(\sigma(\phi(i)),\sigma(\phi(j)))}$ Verifier
• Verification:	$\mathsf{Verify}(x, \mathsf{Com}, e, z)$ accepts if and only if $\sigma(\phi(i)) \neq \sigma(\phi(j))$

Theorem 3.2.1. *[119] The protocol demonstrated in Figure 3.1 is a Zero-Knowledge proof system, assuming the hiding and binding property of the commitment scheme.*

Consider that, there is a reduction to 3-colouring language, by applying the Karp reduction, we have the following theorem:

Theorem 3.2.2. *If one-way functions exist, then every* **NP***-language has a zero-knowledge interactive proof system.*

3.3 Zero-Knowledge Proof Systems; Variants

Modifying the requirement of a Zero-Knowledge proof system yields new variations of the system.

3.3.1 Simulator with Auxiliary Input

In the original definition, the simulator does not take any auxiliary input [128]. In contrast, in the revisited definition [126], they relaxed the definition for a Zero-Knowledge property by considering a simulator with auxiliary input, $\mathsf{Sim}(x, \mathsf{AuxInput})$, that has an indistinguishable description from the actual protocol:

Zero-Knowledge property. For any $x \in \mathscr{L}$, there exists a PPT algorithm; Sim called a simulator, such that the following two distributions are identical:

$$\left\{\mathcal{V}iew[\mathsf{Verify}^{\rightleftharpoons \mathsf{Prove}}](x)\right\}_{x\in\mathscr{L}} \equiv \{\langle\mathsf{Sim}\rangle(x, \mathsf{AuxInput})\}_{x\in\mathscr{L}}$$

This relaxation of the simulator is widely used, and a lot of the literature uses the revisited definition to introduce the Zero-Knowledge proof system. Practically all known Zero-Knowledge proofs are auxiliary-input Zero-Knowledge proofs. (Some examples of the original version can be found in [121] and some with a non-black-box simulator [25])

3.3.2 Perfect, Statistical, Computational ZK

Recall that three variants of indistinguishability; Perfect, statistical and computational indistinguishability(see 2.4.1). Each indistinguishable interpretation provides a different form of Zero-Knowledge that has been widely investigated in the literature. [123]

- Perfect Zero-Knowledge (**PZK**): It requires that the following distributions to be identical:

$$\left\{\mathcal{V}iew[\mathsf{Verify}^{\rightleftharpoons \mathsf{Prove}}](x)\right\}_{x\in\mathscr{L}} \equiv \left\{\langle\mathsf{Sim}\rangle(x, \mathsf{AuxInput})\right\}_{x\in\mathscr{L}}$$

- Statistical Zero-Knowledge, almost-perfect (**SZK**): It requires the statistical distance of the following distributions to be negligible:

$$\{\langle\mathsf{Prove}, \mathsf{Verify}\rangle(x)\}_{x\in\mathscr{L}} \overset{\mathsf{static}}{=} \{\langle\mathsf{Sim}\rangle(x, \mathsf{AuxInput})\}_{x\in\mathscr{L}}$$

- Computational Zero-Knowledge (**CZK**): It requires that the two probability ensemble to be indistinguishable by any PPT adversary:

$$\left|\Pr\left[\mathcal{A}(\{\langle P, V\rangle(1^\ell)\}) \mapsto 1\right] - \Pr\left[\mathcal{A}(\{\langle\mathsf{Sim}\rangle(1^\ell, x, \mathsf{AuxInput})\}) \mapsto 1\right]\right| < \mathsf{negl}(\ell)$$

The class **PZK**, **SZK** and **CZK** are defined as all languages that have a perfect, statistical and computational, respectively, Zero-Knowledge proof system with a polynomial number of rounds (in its input length). Although **CZK** systems are the most liberal notion, they are very expressive and offer significant Zero-Knowledge guarantees. It is proven that assuming one-way functions exist, and every NP-language has a computational Zero-Knowledge proof system [124], followed by a stronger result proven in [158, 34], which state that:

$$\mathbf{BPP} \subset \mathbf{PZK} \subseteq \mathbf{SZK} \subset \mathbf{CZK} = \mathbf{IP} = \mathbf{PSPACE}$$

3.3.3 Expected Polynomial-Time Simulators

Following [26] in the context of Zero-Knowledge, efficiency has also been interpreted to imply polynomial on the average, *i.e.,* the expected polynomial-time algorithm. Suppose we fix the algorithm's input and consider the algorithm's running time as a random variable (dependent on its coin tosses). In this case, we call the algorithm expected in polynomial time its random variable the expectation is polynomial.

As mentioned in [120, 188], it is shown that this approach is quite problematic since it is not model-independent and is not closed under algorithmic composition. However, suppose the simulator runs in an expected polynomial (expectation is taken over the coin tosses of the simulator) rather than strict polynomial time. In this case, we have a new variant that we call *Expected Polynomial-Time Simulators* Zero-Knowledge proof system. This yields the following formal definition:

Definition 17. *If for a Zero-Knowledge proof system* $\langle \mathsf{Prove}, \mathsf{Verify} \rangle$ *the Zero-Knowledge property holds with respect to an expected polynomial-time simulator. Namely, for every* $x \in L$ *the random variables* $\left\{\mathcal{V}iew[\mathsf{Verify}^{\rightleftharpoons \mathsf{Prove}}](x)\right\}_{x \in \mathscr{L}}$ *and* $\left\{\langle \mathsf{Sim} \rangle (x, \mathsf{AuxInput})\right\}_{x \in \mathscr{L}}$ *are identically distributed, we call proof system, Zero-Knowledge with expected polynomial-time simulators.*

3.3.4 Knowledge Tightness

Knowledge tightness is a security measure specific to the Zero-Knowledge property, and intuitively, it measures the "real security" of the proof system. In other words, it quantifies how much harder the verifier must work while not interacting with the prover to compute anything that it can compute after interacting with the prover. Thus, knowledge tightness is the ratio between the simulator's running time and the verifier's running time in the real interaction simulated by the simulator.

Definition 18 (Knowledge Tightness [119]**).** *Let* $t : \mathbb{N} \leftarrow \mathbb{N}$ *be a function. We say that a Zero-Knowledge proof for language* $\mathscr{L}$ *has knowledge tightness if there exists a polynomial* $\mathsf{poly}()$ *such that for every probabilistic polynomial-time verifier there exists a simulator* Sim *such that for all sufficiently long* $x \in \mathscr{L}$ *we have:*

$$\frac{t_{\mathsf{Sim}}(x) - \mathsf{poly}(|x|)}{t_{\mathsf{Verify}}(x)} \leq t(|x|),$$

where $t_{\mathsf{Sim}}(x)$ *denotes the expected running time of the simulator on input* x *and* $t_{\mathsf{Verify}(x)}$ *denotes the running time of the verifier on input* x*.*

Notably, the Zero-Knowledge property does not guarantee polynomial knowledge tightness, even though all known Zero-Knowledge proofs and, more broadly, all Zero-Knowledge properties using a single simulator with black-box access to verifier have polynomial knowledge tightness [119].

3.3.5 Arguments; Computationally Sound ZK

In the interactive proof system, we relax the soundness property in the following way: rather than requiring that it is impossible to fool the verifier into accepting false statements, we require that it be infeasible. This property is referred to as computational soundness, and the proof system that possesses it is referred to as an argument proof system (or sound proof system). Compared to the proof system, the arguments proof system has several theoretical and practical advantages. Theoretically, it is demonstrated that *Perfect* Zero-Knowledge computationally sound proof systems can be constructed for all NP-languages under some reasonable assumption. Additionally, computationally sound proof systems are significantly more efficient than conventional proof systems in practice.

Definition 19. *An interactive system* $\langle \mathsf{Prove}, \mathsf{Verify} \rangle$ *is called a computationally sound proof system or an argument for a language* $\mathscr{L}$ *if both prover and verifier are polynomial-time with auxiliary input with the following property:*

1. ***Completeness**:* $\forall x \in \mathscr{L}\ \exists w \in \{0,1\}^*\ s.t. \forall z \in \{0,1\}^*$:

$$\Pr\left[\, \langle \mathsf{Prove}(w), \mathsf{Verify}(z) \rangle (x) = 1 \,\right] = 1$$

2. ***Computational Soundness**: For every polynomial-time interactive machine B and all sufficiently long* $x \notin \mathscr{L}$ *and every y and z:*

$$\Pr\left[\, \langle \mathsf{Prove}(y), \mathsf{Verify}(z) \rangle (x) = 1 \,\right] \leq \frac{1}{3}$$

3. ***Zero-Knowledge:** Same as definition 16.*

3.4 Proof of Knowledge

We distinguish two languages, $\mathscr{L}_1$ and $\mathscr{L}_2$ for the cyclic group $\mathbb{G} = \langle g \rangle$ in which the discrete logarithm problem is hard:

$$\begin{aligned} \mathscr{L}_g &: \mathrm{R}_g = \left\{(x, w) : x = (\mathbb{G}, g, h), h = g^w\right\} \\ \mathscr{L}_{\mathsf{ddh}} &: \mathrm{R}_{ddh} = \left\{(x, w) : x = \big((g, h), (u, v)\big), u = g^w, v = h^w\right\} \end{aligned} \tag{3.1}$$

We say $\mathscr{L}_g$ is a trivial language. However, for every element $h \in \mathbb{G}$, a $w \in \mathbb{Z}_p$ such that $h = g^w$ exists due to the cyclic nature of $\mathbb{G}$. Therefore, each random statement h is considered a valid statement with R_g. In other words, the proof, π does not establish the validity (which is self-evident); rather, it establishes the asserting the knowledge of some witness, not merely its existence. On the other hand, since NOT every tuple (u, v) is a valid statement in language $\mathscr{L}_{\mathsf{ddh}}$, a prover can construct proof even without knowing the precise witness. The idea of PoK distinguishes two scenarios: in the first, the prover is aware of the validity but not necessarily of the witness, whereas in the second, the proof establishes that the prover is aware of the witness.

In a formal framework, we capture the concept of proof of knowledge by using an efficient algorithm that uses the proof system as a black-box and outputs a valid witness, which we define with the help of the concept message-specification functions. Furthermore, the next-message function captures the fact that the extractor has fine-grained oracle access to the prover algorithm.

Definition 20 (**Message-Specification Function**[119]). *Denote by $P_{x,y,r}(\bar{m})$ the message sent by machine P on common input x, auxiliary input y and randomness* r *after receiving the message m it is called the message specification function of machine P.*

An oracle machine having access to the function $P_{x,y,r}$ will present machine P's knowledge on (x, y, r). This oracle is referred to as an extractor, and its task is to finding w, a witness for x. The extractor's running-time must be inversely related to the corresponding acceptance probability.

Definition 21 (**Zero-Knowledge proof of Knowledge**). *An (n-round) interactive Zero-Knowledge proof system, for the language $\mathscr{L}$ is a Zero-Knowledge Proof of Knowledge (PoK) if it has a knowledge extraction property (with knowledge error κ), detailed as follows:*

Knowledge Extraction property: *If there exists an efficient algorithm* Extr *and a polynomial* poly *such that for any statement x, the oracle algorithm* $\mathsf{Extract}^{\rightleftharpoons \mathsf{Prove}_{x,w;\mathsf{r}}}$ *runs in expected polynomial time and satisfies:*

$$\Pr\left[\ w^* \leftarrow \mathsf{Extract}(x)^{\rightleftharpoons \mathsf{Prove}_{x,w;\mathsf{r}}} \mid R(x, w^*) = 1\ \right] \geq \frac{1-\kappa}{\mathsf{poly}(|x|)}$$

Additional Note. The original definition for proof of knowledge considers the extractor with polynomial-expected running time. We obtain the strong PoK property definition by replacing the extractor that strictly runs polynomial.

Applications. PoK has a wide variety of applications in real-world protocols; for example, we can use PoK to develop cryptographic primitives such as non-oblivious commitment schemes, non-malleable CPA, and CCA-secure cryptosystem. Additionally, it has a wide range of applications in mutual disclosure of same data and e-voting protocols.

We summarize the result for the PoK proof system in the following theorem:

Theorem 3.4.1 ([119]). *Assuming the existence of (non-uniformly) One-Way functions, every NP-relation has a Zero-Knowledge system for proofs of knowledge. Furthermore, inputs not in the corresponding language are accepted by the verifier with exponentially vanishing probability.*

3.5 Sigma Protocol

Sigma protocol is one of the most well-studied and popular Zero-Knowledge-proof systems, which is a three-round interactive, honest-verifier Zero-Knowledge proof of knowledge for an NP-relation R.

Definition 22 (**Sigma-Protocol** [233]). *A Sigma protocol for an NP-relation R is a public-coin, 2-party interactive, honest-verifier and proof of knowledge that has the following three rounds:*

1. *On input $(x, w) \in R$, the prover sends the commitment of values r to the verifier:*
2. *On input x, the verifier sends a uniformly random challenge e to the prover.*
3. *The prover responds to the challenge e, by sending $f(x, w, r, e)$ where f is some public function.*

As a concrete example of the Sigma Protocol, we present the Schnorr Protocol.

The Schnorr Protocol. Consider the group $\mathbb{G} = \langle g \rangle$ with order p for some prime number p and some random generator g such that the discrete logarithm is a hard problem in $\mathbb{G}$ 4. Then the following is a sigma protocol:

FIGURE 3.2: The Schnorr Protocol

• Setting:	$\mathcal{G} = (G, g, p) : \mathbb{G} = \langle g \rangle; \lvert G \rvert = p,$
	$\mathrm{R}_g = \{(x, w) : x = (\mathcal{G}, g, h), h = g^w\}$
• Inputs:	$\mathsf{Prove}: \ x, w, \ \mathsf{Verify}: \ x$
• Protocol:	
Prover Commitment:	
Round 1:	Pick $\mathsf{random} \xleftarrow{\$} \mathbb{Z}_p$
	Compute $\mathsf{Com} = g^{\mathsf{random}}$
	Prover $\xrightarrow{\mathsf{Com}}$ Verifier
Verifier Challenge	
Round 2:	Pick $e \xleftarrow{\$} \mathbb{Z}_p$
	Verifier $\xrightarrow{e}$ Prover
Prover Respond:	
Round 3:	Compute $z = w \times e + \mathsf{random}$
	Prover $\xrightarrow{z}$ Verifier
Verification	$\mathsf{Verify}(x, \mathsf{Com}, e, z)$ accepts if and only if $g^z = h^e \times \mathsf{Com}$

Theorem 3.5.1. *[233] The Schnorr protocol is a Sigma-Protocol. Namely, it is perfectly complete, knowledge-extractable, and honest-verifier Zero-Knowledge proof system.*

It is worth mentioning that we can transform any sigma-protocol into full-fledged Zero-Knowledge proofs of knowledge using some standard techniques [113], at the cost of an additional round of interaction (plus a small additive cost in the communication). Furthermore, we may transfer any interactive sigma protocol to the non-interactive Zero-Knowledge proof system using the Fiat-Shamir Paradigm (See 3.8.4).

Sigma Protocol for DDH-Relation. As a second example for Sigma-protocol, we demonstrate the interactive proof system for relation R_{DDH}. As we will see in the second part of our research, we frequently rely on the non-interactive version of this proof to design a verifiable e-voting protocol.

FIGURE 3.3: Sigma Protocol for proof of knowledge of Paillier Plaintext

• **Setting:**	$\mathcal{G}^{\mathsf{RSA}} = (n, p, q, G, g)$,
	$\mathrm{R}_g = \{(x, w) : x = (n, g, \mathsf{ct}), w = (m, \mathsf{r}) \in \mathbb{Z}_n \times \mathbb{Z}_n^* : \mathsf{ct} = g^m \times \mathsf{r}^n\}$
• **Inputs:**	Prover: (x, w), Verifier: (x)
• **Protocol:**	
Prover Commitment:	
Step 1:	Pick $(a, b) \xleftarrow{\$} \mathbb{Z}_n \times \mathbb{Z}_n^*$
	Compute $\mathsf{Com} = g^a \times b^n$
	Prover $\xrightarrow{\mathsf{Com}}$ Verifier
Verifier Challenge	
Step 2:	Pick $e \xleftarrow{\$} \mathbb{Z}$
	Verifier $\xrightarrow{e}$ Prover
Prover Respond:	
Step 3:	Compute $z_1 = e \times m + a \mod n$,
	$c = (e \cdot m + a - z_1)n^{-1} \mod n$,
	$z_2 = (b \times \mathsf{r}^e) \cdot g^t \mod n^2$,
	$\mathcal{P} \xrightarrow{\pi = (z_1, z_2)}$ Verifier
Verification:	$\mathsf{Verify}(x, \mathsf{Com}, e, \pi)$ accepts if and only if $g^{z_1} \cdot z_2^n = \mathsf{ct}^e \cdot \mathsf{Com}$

Sigma Protocol for Proof of Knowledge of Paillier Plaintext. Since we use Paillier cryptosystem and its proof of knowledge in our protocol 7, as a third example we present the Sigma Protocol for proof of knowledge of Paillier plaintext.

3.6 Composing Zero-Knowledge Proof Systems

We now discuss Zero-Knowledge proof system composition, focusing on which properties of a proof system are kept throughout the composition, and which do not necessarily remain intact. By the *composition* of proof systems, we refer to the execution of many copies of the protocol, with the prescribed (honest) parties executing each copy independently of the others. For example, if a party is required to toss coins in a particular round, it will toss independent coins for each duplicate. We consider (polynomially) many Zero-Knowledge proof systems composed in parallel and sequential.

3.6.1 Sequential Composition

Sequential composition invokes a set of ZK proof systems multiple (polynomial) times, with each invocation following the termination of the previous one. The interesting result, in this case, is that the ZK proof system as defined originally (simulator without auxiliary input) is not closed under sequential composition [121]. Yet, the modified version (simulator with auxiliary inputs) retains its Zero-Knowledgeproperty after sequential repetition. [126]

3.6.2 Parallel Composition

Parallel composition invokes a set of ZK proof systems multiple (polynomial) times simultaneously and proceeds at the same pace. There exist two negative results regarding the parallel composition of Zero-Knowledge protocols. The first approach refutes the parallel conjecture of the parallel-composition conjecture by constructing a counterexample but does not explicitly mention the natural candidates. In contrast the second approach establishes that there is a class of Zero-Knowledge proof systems whose members cannot be proved Zero-Knowledge in parallel composition using a general paradigm (known by the name "black-box simulation") [119].

The parallel composition conjecture can also be refuted for probabilistic polynomial-time prover (with auxiliary inputs) and statistical Zero-Knowledge proof systems.

We briefly demonstrate an example that shows the parallel composition of two Zero-Knowledge, where the proof system is not always a Zero-Knowledge proof system.

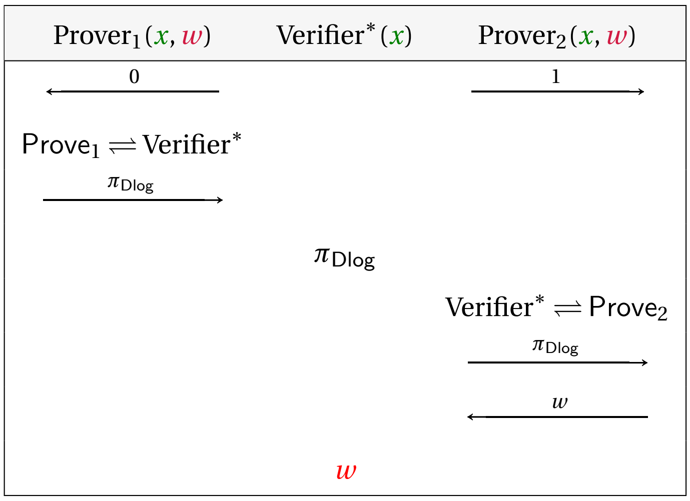
Example [102]. Consider the Zero-Knowledge system for discrete logarithm language in (3.1), $\Pi^{\mathsf{dLog}} = \langle \mathsf{Prove}, \mathsf{Verify} \rangle$. we construct a new Zero-Knowledge proof system as follows:

1. On input $(\mathcal{G}, g, h)$, Verifier* tries to guess w randomly. If Verifier* succeeds he sends 1; otherwise sends 0.

2. If Verifier* sent 1, then it proves the knowledge of w using the protocol Π^{dLog}. If the prover is convinced by Verifier*, the prover sends w to the verifier; otherwise, the prover sends reject and terminates the protocol.

3. If Verifier* sent 0 in step 1, the prover proves the knowledge of w using Π^{dLog}.

FIGURE 3.4: A parallel composition of two Zero-Knowledge Protocol



The fact that all known formulations of (computational) Zero-Knowledge are not closed under parallel composition motivates the introduction of weaker notions such as *witness indistinguishability*.

3.7 Witness Indistinguishable and Witness Hiding Proof System

Consider an interactive proof system with multiple witnesses for each statement. If the verifier cannot determine which witness the prover employs to generate the proof, we say the system has witness indistinguishability. Furthermore, the system is witness hiding; if the verifier cannot generate any new witnesses (he was unaware of before the protocol began) after interacting with the prover [102].

We formally define these two systems as follows where $\mathrm{R}_{\mathscr{L}}(x)$ denotes the set of witnesses for the statement x.

3.7.1 Witness Indistinguishability

Definition 23 (Witness Indistinguishable Proof System). *Consider an interactive proof system* $\Pi^{\mathsf{wi}} = \langle \mathsf{Prove}, \mathsf{Verify} \rangle$ *for an* ***NP****-language* $\mathscr{L}$ *with relation* $R_{\mathscr{L}}$*. We say it is witness-indistinguishable for* $R_{\mathscr{L}}$ *iffor every PPT distinguisher* $\mathcal{D}$ *and all* $z \in \{0,1\}^*$ *it holds:*

$$\Big| \Pr\Big[\mathcal{D}\big(x, z, \Big\{\mathcal{V}iew[\mathsf{Verify}(z)^{\mathcal{P}(\mathsf{w}_x^1)}](x)\Big\}_{x\in\mathscr{L}, z\in\{0,1\}^*}\big) = 1\Big] \\ -\Pr\Big[\mathcal{D}\big(x, z, \Big\{\mathcal{V}iew[\mathsf{Verify}(z)^{\mathcal{P}(\mathsf{w}_x^2)}](x)\Big\}_{x\in\mathscr{L}, z\in\{0,1\}^*}\big) = 1\Big] \Big| \\ < \mathsf{negl}(|x|)$$

A stronger notion for the WI proof system defines as follows:

Definition 24 (Strong Witness Indistinguishable[119]). *Interactive system* $\langle \mathsf{Prove}, \mathsf{Verify} \rangle$ *is strongly witness indistinguishable proof system for* $R_{\mathscr{L}}$ *if for every two probability ensemble:*

$$\{X_n^1, Y_n^1, Z_n^1\} \ , \ \{X_n^2, Y_n^2, Z_n^2\}$$

such that $\{X_n^b, Y_n^b, Z_n^b\}$ *ranges over* $(R_{\mathscr{L}} \times \{0,1\}^*) \cap (\{0,1\}^n \times \{0,1\}^* \times \{0,1\}^*)$ *,the following holds:*

If $\{X_n^1, Z_n^1\}$ *and* $\{X_n^2, Z_n^2\}$ *are computationally indistinguishable, then so are*

$$\Big\{\mathcal{V}iew[\mathsf{Verify}(Z_n^1)^{\mathsf{Prove}(\mathsf{Y}_n^1)}](X_n^1)\Big\}_{n\in\mathbb{N}} \ , \ \Big\{\mathcal{V}iew[\mathsf{Verify}(Z_n^2)^{\mathsf{Prove}(\mathsf{Y}_n^2)}](X_n^2)\Big\}_{n\in\mathbb{N}}$$

Additional Note. According to [119] although it is proved that any auxiliary-input Zero-Knowledge proof system, for an NP-language is strongly witness-indistinguishable, assuming that one-way permutations exist, witness indistinguishability does not imply strong witness indistinguishability.

3.7.2 Witness Hiding

A proof system for an NP-language has the witness-hiding property if the verifier after interacting with the prover cannot find a fresh witness for the statement [102, 87, 185, 149, 93].

It should be noted that the witness hiding property only makes sense if obtaining witnesses from scratch is impossible. As every language has instances where witness retrieval is straightforward, we need to consider witness retrieval for specially selected difficult instances. As a result, to capture the above pointers, we consider the concept of "Distribution of Hard Instances", which means for the language $\mathscr{L}$ correspond to the relation $\mathrm{R}_{\mathscr{L}}$, with the probability distribution

$$X = \left\{X_n : \mathscr{L} \cap \{0,1\}^n\right\},$$

the following holds true:

$$\Pr\left[F(X_n, z) \in \mathrm{R}_{\mathscr{L}}(X_n)\right] < \mathsf{negl}(n)$$

Where F is a probabilistic polynomial-time (witness-finding) algorithm. Hence considering this definition a Zero-Knowledge proof system has witness hiding property; the probability of finding the witness for a verifier remains negligible after interacting with the prover.

Additional Note. Although in general WI does not implies witness hiding property, we have the following result: Consider a WI proof system, $\langle \mathsf{Prove}, \mathsf{Verify} \rangle$ for relation R with a PPT prover algorithm. We define the new relation as follow:

$$\mathrm{R}_2 := \left\{((x_1, x_2), w) : |x_1| = |x_2|\,, \exists i : (x_i, w) \in \mathrm{R}_{\mathscr{L}}\right\}$$

Then Π^{zk} has witness hiding property for R_2.

Additional Note.

1. A WI-proof system is called witness-independent if the above ensembles are identically distributed.
2. If the prover can generate proof without considering the witness in any proof system, then the proof system is witness indistinguishable. This shows that the Witness Indistinguishability property is practically defined for the bounded prover that takes the witness as its private auxiliary input. However, for non-trivial language, a PPT prover cannot generate valid proof without having the witness.
3. Any Zero-Knowledge proof system also has the Witness Indistinguishability property, while the other Witness Indistinguishable property does not always imply the Zero-Knowledge property.
4. Although the WI proof system guarantees a weaker security level, it has many applications due to some of its properties. For example unlike the Zero-Knowledge proof systems, a WI property is preserved in parallel compositions.

3.8 Non-Interactive Zero Knowledge Proof Systems

As stated in [33], the zero-knowledge proof system's results are based on interactive protocols that guarantee the highest levels of real-world security in an adversarial context without making any trust assumptions. Hence, the approach is referred to as the plain model, and it provides the strongest real-world security guarantees in an adversarial context. However, the ZK-Proof system cannot be used in real-world protocols such as electronic voting or public key exchange because of its interactive nature. Indeed, due to the large number of parties involved, even a single interaction causes a high cost on these systems.

On the other hand, based on the impossibility result for the zero-knowledge proof system in the plain model with a single round of interaction for non-trivial languages [126], we know that we must either sacrifice some security property or replace it with a trusted assumption if we wish to avoid using the interactive requirement. Given that security is a highly desirable property, the first approach (giving up the security level) is not an option; thus, we constructed the non-interactive zero-knowledge proof system by introducing a trusted assumption captured in two different approaches: the Hidden-Bit model and the CRS-Model.

This section will present an overview of the background and the result of the non-interactive zero-knwoledge proof system, which we refer to as NIZK.

3.8.1 NIZK in RHB Model

In the hidden-bits model of the NIZK proof system [101, 118], the prover is initially given a sequence of bits that are hidden from the verifier. The prover then chooses an arbitrary subset of these bits to reveal to the verifier. Although the verifier never learns the unrevealed parts of the string, the prover cannot alter the values in the string it is given. Formally assume that the prover is given the string s of length n and sends to the verifier $\{s_i\}_{i\in I}$ where $I \subset \{1,2,\ldots,n\}$ is the index set.

Definition 25 (**Non-Interactive Zero-Knowledge Proof System**). *A pair of probabilistic algorithms $\Pi^{\mathsf{nizk}} = \langle \mathsf{Prove}, \mathsf{Verify}\rangle$ is called a non-interactive zero-knowledge proof system in RHB-model (random hidden bit) for language $\mathscr{L}$ if* Verify *is a polynomial-time algorithm and the following conditions hold true:*

- **Completeness property:** *For very $x \in \mathscr{L}$, the verifier always outputs 1 (accept the proof) after interacting with the prover:*

$$\forall x \in \mathscr{L} : \Pr\Big[\mathsf{Verify}(x,(I,s_I),\pi) = 1 \mid s \xleftarrow{\$} \{0,1\}^{\mathsf{poly}}, (\pi, I) \leftarrow \mathsf{Prove}(s,x,w)\Big] = 1$$

- **Soundness property:** *For any $x \notin \mathscr{L}$, and any potential adversary $\mathcal{A}$, the verifier output* 0 *(reject the proof) with overwhelming probability:*

$$\forall x^* \notin \mathscr{L} : \Pr\Big[\mathsf{Verify}(x^*,(I,s_I),\pi) = 1 \mid s \xleftarrow{\$} \{0,1\}^{\mathsf{poly}}, (\pi, I) \leftarrow \mathcal{A}(s,x)\Big] < \mathsf{negl}(\ell)$$

- **Zero-Knowledge property:** *For any $x \in \mathscr{L}$ a PPT* simulator *algorithm,* Sim *exists, such that the following two distributions are identical:*

$$\begin{gathered}\big\{(x,(I,s_I),\pi)\big| s \leftarrow \mathsf{crsGen}(1^{\ell}), \pi \leftarrow \mathsf{Prove}(s,x,w)\big\}\\ \approx \\ \big\{(x,(I,s_I),\pi)\big| s \xleftarrow{\$} \{0,1\}^{\mathsf{poly}}, \pi \leftarrow \mathsf{Sim}(s,x)\big\}\end{gathered} \tag{3.2}$$

Additional Note. It is worth noting that the hidden-bits model is not intended to be realistic; rather, it has intended to be conceptual. However, this model facilitates the existential and constructive path toward a realistic, concrete model, the CRS-Model. To begin, it establishes a simple abstraction for NIZK systems, which we know exist for NP-hard languages due to the following theorem:

Theorem 3.8.1. *[101] There exists a NIZK proof system in the hidden-bit model for any NP-language (unconditionally). Furthermore, the protocol is statistical zero-knowledge and statistically sound.*

3.8.2 NIZK in CRS Model

Ivan Damgård introduced the first approach to establishing a non-interactive Zero-Knowledge proof system that achieves a sufficient level of security in [89]. In this model, we assume the existence of an algorithm called CRS-generator that chooses a common-reference string in an honest way (CRS). The prover and verifier are both given access to a common string that serves as their input for generating and verifying the proof, respectively. Formally

Definition 26 (**Interactive Zero-Knowledge Proof System**). *A pair of probabilistic algorithms* $\Pi^{\mathsf{nizk}} = \langle \mathsf{Prove}, \mathsf{Verify} \rangle$ *is called a non-interactive zero-knowledge proof system in CRS-model (common random string) for language* $\mathscr{L}$ *if* Verify *is a polynomial-time algorithm and the following conditions hold true:*

- **CRS-Generator**: *A PPT algorithm* $\mathsf{crsGen}(1^\ell)$ *exists that on input the security parameter, generates a string* crs *:*

$$\mathsf{crs} \leftarrow \mathsf{crsGen}(1^\ell)$$

- **Completeness property:** *For very* $x \in \mathscr{L}$*, the verifier always outputs 1 (accept the proof) after interacting with the prover:*

$$\forall x \in \mathscr{L} : \Pr\Big[\mathsf{Verify}(x, \mathsf{crs}, \pi) = 1 \mid \mathsf{crs} \leftarrow \mathsf{crsGen}(1^\ell), \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\Big] = 1$$

- **Soundness property:** *For any* $x \notin \mathscr{L}$*, and any potential adversary* $\mathcal{A}$*, the verifier output* 0 *(reject the proof) with overwhelming probability:*

$$\forall x^* \notin \mathscr{L} : \Pr\Big[\mathsf{Verify}(x^*, \mathsf{crs}, \pi) = 1 \mid \mathsf{crs} \leftarrow \mathsf{crsGen}(1^\ell), \pi \leftarrow \mathcal{A}(\mathsf{crs}, x^*)\Big] < \mathsf{negl}(\ell)$$

- **Zero-Knowledge property:** *For any* $x \in \mathscr{L}$ *there exists a PPT* simulator *algorithm,* Sim*, such that the following two distributions are identical:*

$$\begin{aligned} &\Big\{(x, \mathsf{crs}, \pi) \Big| \mathsf{crs} \leftarrow \mathsf{crsGen}(1^\ell), \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\Big\}_\ell \\ &\approx \\ &\Big\{(x, \mathsf{crs}, \pi) \Big| \mathsf{crs} \leftarrow \mathsf{crsGen}(1^\ell), \pi \leftarrow \mathsf{Sim}(\mathsf{crs}, x)\Big\}_\ell \end{aligned} \tag{3.3}$$

Additional Note. According to some definitions, the simulator is composed of two algorithms, $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$, the first of which generates a simulated CRS, crs^*. This provides the simulator with additional power, as the simulator may generate some trapdoor that helps in the subsequent generation of a valid proof. In this case the equation 3.4 is replaced with:

$$\begin{aligned} &\{(x, \mathsf{crs}, \pi) \big| \mathsf{crs} \leftarrow \mathsf{crsGen}(1^\ell), \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\}_\ell \\ &\approx \\ &\{(x, \mathsf{crs}, \pi) \big| \mathsf{crs}^* \leftarrow \mathsf{Sim}_1(1^\ell), \pi \leftarrow \mathsf{Sim}_2(\mathsf{crs}, x)\}_\ell \end{aligned} \tag{3.4}$$

Composable Zero-Knowledge: The composable zero-knowledge property was first introduced in [139], strengthening the standard zero-knowledge definition in the following way. First, it requires that an adversary cannot distinguish a real CRS from a simulated CRS. Second, it requires that the adversary, even when access to the trapdoor, cannot distinguish real proofs on a simulated CRS from simulated proofs. This robust security property ensures that the same common reference string can be used for multiple proofs, which enhances the proof system's composability:

1. **Reference String Indistinguishability:** For all non-uniform polynomial-time adversary $\mathcal{A}$, the following holds true:

$$\Pr\Big[\mathcal{A}(\mathsf{crs}) = 1 \mid \mathsf{crs} \leftarrow \mathsf{crsGen}(1^{\ell})\Big] \approx \Pr\Big[\mathcal{A}(\mathsf{crs}^*) = 1 \mid (\mathsf{crs}^*, \tau) \leftarrow \mathsf{Sim}_1(1^{\ell})\Big]$$

2. **Simulation Indistinguishability:** For all non-uniform polynomial-time adversary $\mathcal{A}$ the following holds true:

$$\begin{aligned} &\Pr\left[\mathcal{A}(\pi) = 1 \wedge (x, w) \in \mathrm{R}_n \mid \begin{array}{r} (\mathsf{crs}^*, \tau) \leftarrow \mathsf{Sim}_1(1^{\ell}) \\ (x, w) \leftarrow \mathcal{A}(\mathsf{crs}^*, \tau) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w) \end{array}\right] \\ \approx &\Pr\left[\mathcal{A}(\pi) = 1 \wedge (x, w) \in \mathrm{R}_n \mid \begin{array}{r} (\mathsf{crs}^*, \tau) \leftarrow \mathsf{Sim}_1(1^{\ell}) \\ (x, w) \leftarrow \mathcal{A}(\mathsf{crs}^*, \tau) \\ \pi \leftarrow \mathsf{Sim}_2(\mathsf{crs}, x, \tau) \end{array}\right] \end{aligned} \tag{3.5}$$

Theorem 3.8.2. *[101]Assuming the existence of trapdoor permutations and any NIZK proof system in the hidden-bits model, we may construct a NIZK proof system in the common random string model.*

Adaptive versus non-Adaptive NIZK. We consider adaptive and non-adaptive versions of the soundness property. *Adaptive soundness* allows $x \notin \mathscr{L}$ to be adaptively chosen after the crs is fixed, whereas *non-adaptive soundness* requires the adversary to choose the statement before seeing the crs.

Non-Interactive sigma protocol in RO model for relation $\mathrm{R}_{\mathsf{OR-DDH}}$ is shown in Figure 8.1.

3.8.3 NIZK for NP-Language

Similarly to the ZK proof system, we can establish the existence of a NIZK in two steps for all NP-languages:

First, we build a NIZK for an **NP**-complete language, and then we use the Karp reduction to extend it to all NP-languages.

Feige, Lapidot and Shamir in [101], develop a NIZK proof system in the HBS model for Hamiltonian-Graph language and then convert it to NIZK in the CRS model using a technique known as the FLS technique. More recently, Groth, Ostrovsky, and Sahai [141] presented an efficient design for Circuit-SAT based on a bilinear group that obtained perfect zero-knowledge.

This, together with other constructs in the literature, leads to the following theorem:

Theorem 3.8.3 ([119]). *Assuming the existence of one-way permutations, each language in **NP** class has a non-interactive proof system that is adaptively Zero-Knowledge proof system. Furthermore, assuming the existence of families of trapdoor permutations, the* *prover strategy in such a proof system can be implemented by a probabilistic polynomial-time machine that gets an NP-witness as an auxiliary input.*

3.8.4 Fiat-Shamir Heuristic

The Fiat–Shamir heuristic, introduced in [104], is a technique that transforms an interactive proof system (such as sigma-protocol) into non-interactive zero-knowledge proofs with the help of some secure hash function(See 2.7.3).

In general, this approach replaces the verifier's challenge (See 3.5) value with the output of some random oracle that depends on the prover commitment values. This results in reducing the interaction rounds to a single round interaction between the prover and the verifier.

While the Fiat-Shamir approach provides an extremely efficient non-interactive ZK proof system, the security property of NIZK is dependent on the hash function used. As a result, we cannot always assume that the NIZK system is secure based on its security in the RO model [69]. We refer to [32], in which the authors demonstrate that some protocol instantiation in the random oracle model may be proven secure, whereas some hash functions are insufficient as a replacement for a random oracle. As such, this abstraction should be viewed as a heuristic indicator of security.

For more details on Fiat-Shamir Heuristic approach we refer to [70].

3.8.5 Designated Verifier Zero-Knowledge Proof Systems

For many Zero-Knowledge proof systems, only the verifier designated by some secret input (verification-key) must be convinced of the statement's validity. In contrast to the Zero-Knowledge proof system, the verifier does not need any extra inputs to run the verification algorithm. The formal definition for non-interactive designated verifier (NIDV) proofs was first introduced by Jakobsson *et al.* in [163] and have been used as confirmation and denial proofs for undeniable signature schemes.

Definition 27 (Designated Verifier NIZK[178]). *A designated verifier non-interactive Zero-Knowledge proof system is a tuple of PPT algorithms* $\Pi^{\mathsf{dv-nizk}} = \langle \mathsf{Setup}, \mathsf{Kgen}, \mathsf{Prove}, \mathsf{Verify} \rangle$ *such that:*

- **Set up algorithm** *outputs a common reference string,* crs *and the public parameters* pp*, which describe the language* $\mathscr{L}$*.*
- **Key generator** *takes as input the public parameters and returns a key pair of public key and the verification key,* vk*:*

$$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Kgen}(\mathsf{pp})$$

- **Prover** *generates the proof,* π *on inputs* $(x, w) \in R_{\mathscr{L}}$ *and* pk.
- **Verifier** *on input the public key,* pk*, the verification key,* vk*, statement* x *and the proof* π*, outputs reject or accept.*

which satisfies the completeness, Zero-Knowledge, and soundness properties.

Additional Note. Definition 27 shows that the verification algorithm is the primary distinction between a conventional and a designated verifier. In the latter system, the verification algorithm requires additional input, namely the verification key. In contrast, the former allows any public party to execute the verification algorithm using the system's public parameters. Many protocols, such as electronic voting schemes, use the designated-verifier proof system to demonstrate the validity of computations to the individual voter. We stress that the difference between the designated verifier and the ordinary Zero-Knowledge proof system comes from the verification key

As an example, we present the following example from [74].

Example: A prover wishes to prove to a verifier that he knows a value $w \in \mathbb{Z}_n$ such that $h = g^w$. Let $u \in \mathbb{J}_n$ be an arbitrary generator of $\mathbb{J}_n$. Let's define $R = u^n \mod n^2$ $\mathsf{key} = (\mathsf{pk}, \mathsf{vk}) = (E = R^e, e)$.

Prover steps:

$T' = (1+n)^t R^{\mathrm{r}} \mod n^2, X' = (1+n)^x E^{-\mathrm{r}} \mod n^2$ and then the verifier computes $D = T^e X \mod n^2$ and $D' = T'^e \cdot X' \mod n^2$ and then checks that D' is the form $(1+n)^d$ mod n^2. If so, computes d mod n from D' and checks that $D = g^d$ the verifier accepts if and only if, both checks succeeded.

3.8.5.1 Implicit Zero-Knowledge Proof Systems

Benhamoda *et al.*[42] introduced a new type of Zero-Knowledge proof, called implicit Zero-Knowledge arguments and stands between two existing notions, interactive Zero-Knowledge proofs and non-interactive Zero-Knowledge proofs.

The implicit Zero-Knowledge argument is an encapsulation mechanism that allows masking a message to retrieve if and only if the statement is true. Additionally, iZK maintains the same Zero-Knowledge properties as standard Zero-Knowledge arguments. The ability to unmask a message only leaks the validity of the statement and nothing more.

iZk can be used in two-party computations to force parties to follow the protocol. iZK ensures the confidentiality of the parties' inputs in a different method. However, it does not explicitly check that the opponent behaved honestly. Rather than that, it ensures that if this is not the case, it will be impossible for the other party to recover any further protocol messages.

Definition 28 ([42])**.** *The following polynomial-time algorithms define an* Π^{iZK}*:*

- $\mathsf{icrs} \leftarrow \mathsf{iSetup}(\mathsf{crs})$ *generates the (normal) common reference string which implicitly contains* crs*). The resulting CRS provides statistical soundness.*
- $(\mathsf{icrs}^*, \mathsf{i}\tau) \leftarrow \mathsf{iTSetup}(\mathsf{crs})$ *generates the (trapdoor) common reference string* icrs *together with a trapdoor* $\mathsf{i}\tau$*. The resulting CRS provides statistical Zero-Knowledge.*
- $(\mathsf{ipk}, \mathsf{isk}) \leftarrow \mathsf{iKG}(\mathsf{icrs}, x, \mathsf{iw})$ *generates a pair of keys, associated with statement* $x \in \mathscr{L}$ *and the witness* w*.*
- $(\mathsf{ipk}^*, \mathsf{itk}) \leftarrow \mathsf{iTKG}(\mathsf{i}\tau, x)$ *generates a public and trapdoor key pair, associated with* x*.*
- $(\mathsf{ct}, \mathsf{key}) \leftarrow \mathsf{iEnc}(\mathsf{icrs}, \mathsf{ipk}, x)$ *outputs a ciphertext of a value* key *(an ephemeral key), for* x*.*
- $\mathsf{key} \leftarrow \mathsf{iDec}(\mathsf{icrs}, \mathsf{isk}, \mathsf{ct})$ *decrypts the ciphertext, and outputs the ephemeral key,* key.
- $\mathsf{key} \leftarrow \mathsf{iTDec}(\mathsf{icrs}, \mathsf{itk}, \mathsf{ct})$ *decrypts the ciphertext and outputs the ephemeral key,* key.

Security Notion. Implicit ZK must have correctness, setup indistinguishability, soundness and Zero-Knowledge properties, which are defined similarly to other variants of proof systems. Additionally, it requires the Setup Indistinguishability that states that the two setup outputs should be indistinguishable.

$$\left\{\mathsf{icrs}|\mathsf{icrs} \leftarrow \mathsf{iSetup}(1^{\ell}, \mathsf{crs})\right\}_{\ell} \approx \left\{\mathsf{icrs}^{*}|(\mathsf{icrs}^{*}, \mathsf{i}\tau) \leftarrow \mathsf{iTSetup}(1^{\ell}, \mathsf{crs})\right\}_{\ell}$$

We will not provide the formal definition here; rather, we will refer to the original publication [42] for additional information.

3.8.6 Non-Algebraic Language; Rang-Proof and Proof of Shuffle

All of the ZK protocols we have discussed so far can be naturally expanded to prove a wide range of claims, such as arbitrary algebraic relations between values. As we will see in the second part, many of these proofs are used to demonstrate the well-formedness of some ciphertext or commitment value. However, some languages are not classified as algebraic languages. For example, we mention the range-proof, widely used in e-voting protocols.

A typical technique would be to commit to every single bit of the witness, then demonstrate that each commitment value commits to either 0 or 1, and finally demonstrate that the relation exists. This strategy, however, is relatively inefficient.

Several solutions have been proposed to address this issue, including garbled-circuit-based Zero-Knowledge proofs for statements expressed by boolean circuits [106, 165, 75] and Zero-Knowledge arguments with sub-linear communication based on generalised Pedersen commitments [138, 137].

Due to the extensive application of electronic voting protocols to range-proofs and proofs of shuffle, we refer readers to the following papers for additional information [61, 193, 202, 29, 28, 145, 109]

3.9 Non-Interactive Witness Indistinguishable Proof Systems

The Groth-Sahai NIWI-proof system is, indeed, a turning point in the field of zero-knowledge proof systems.

Since the advent of ZK proof systems, it has been shown that NIZK proofs exist for all NP-languages. However, this fact was proven existentially and not in a constructive way. Precisely, theorem 3.8.3 was proved in the following way. First, we develop a reduction from our language to an NP-complete language (e.g., 3-SAT or Graph colouring problem) for which proof has already had a NIZK proof system. Then we transform back the proof from the complete language to our language using the Karp reduction. Unfortunately, the system obtained in this way is an inefficient, and it is computationally too expensive for real-world protocols and applications.

With the emergence of elliptic curves and bilinear maps in modern cryptography, considerable effort has been spent developing ZK-proof systems over bilinear groups. This research's line has resulted in a fascinating and efficient NIZK proof system, beginning with Groth, Ostrovsky, and Sahai's seminal work [142] and continuing with Groth-Sahai proof techniques [143]. Groth and Sahai present a method based on a set of equations that identifies a broad class of languages for which an efficient pairing-based NIZK could be constructed with security based on the standard bilinear group assumption. The Groth-Sahai proof techniques, which was subsequently updated in [117, 48], have a significant impact on practical applications, and it is one of our primary tools for verifiable and secure computation.

This section formally defines a non-interactive witness indistinguishable proof system (NIWI for short) and briefly reviews the Groth-Sahai proof techniques. Then we point out definitions, notions and notations in this section are taken from [143].

3.9.1 NIWI; Formal Definitions

Notation We let R refer to efficiently computable ternary relation, which includes the member of the form (gk, x, w) where gk is considered the setup group a.k.a. the public parameter, x is the statement, and w is the witness. Compared to the binary relation, here we include the group setting as a part of the triple, which shows the system's flexibility. By $\mathscr{L}$, we refer to the NP-language consisting of the statements x for which witnesses w exist such that $(\mathsf{gk}, x, w) \in \mathrm{R}$. Groth-Sahai setting relates gk to be the description of a bilinear group which implies $\mathscr{L}$ should be corresponding to some bilinear group.

Definition 29 (NIWI-Proof System). *The non-interactive witness indistinguishable proof system,* $\Pi^{\mathsf{niwi}} = \langle \mathsf{Setup}, \mathsf{Kgen}, \mathsf{Prove}, \mathsf{Verify} \rangle$ *for the relation R is a tuple of four probabilistic polynomial-time algorithms, which fulfils the perfect completeness, perfect soundness and composable witness indistinguishability properties as detailed below:*

- ***Set Up*** *is a probabilistic algorithm that takes security parameters and generates a pair* $(\mathsf{gk}, \mathsf{sk})$. *We refer to the first component,* gk, *as a public parameter. The Groth-Sahai setting represents the description of a pairing group setup, and the second component* sk *as the secret parameter. It requires that both keys have a length polynomial in terms of the security parameter.*

$$(\mathsf{gk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^{\ell})$$

- ***Key Generation*** *is a probabilistic algorithm that outputs the CRS on input, the public parameter and the secret key:*

$$\mathsf{crs} \leftarrow \mathsf{Kgen}(\mathsf{gk}, \mathsf{sk})$$

- ***Prove*** *is a probabilistic algorithm that on inputs* gk, crs, x *and* w *first checks whether* $(\mathsf{gk}, x, w) \in R$ *and if so outputs a proof* π*:*

$$\pi \leftarrow \mathsf{Prove}(\mathsf{gk}, \mathsf{crs}, x, w)$$

- ***Verify*** *is a deterministic algorithm that takes* $(\mathsf{gk}, \mathsf{crs}, x, \pi)$ *and outputs* accept *if* π *is a valid proof, namely that* $x \in \mathscr{L}$*, or* reject *if that is not the case:*

$$\mathsf{Verify}(\mathsf{gk}, \mathsf{crs}, x, \pi) = \mathsf{accept} \;/\; \mathsf{reject}$$

Security Requirements: A NIWI proof system is required to have the following properties:

1. **Perfect Completeness:** The verifier always accepts the proofs generated by the prover for the valid statement.

$$\Pr\left[\mathsf{Verify}(\mathsf{gk}, \mathsf{crs}, x, \pi) = \mathsf{accept} \mid \begin{array}{r} (\mathsf{gk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^{\ell}) \\ (\mathsf{crs}) \leftarrow \mathsf{Kgen}(\mathsf{gk}, \mathsf{sk}) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{gk}, \mathsf{crs}, x, w) \end{array} \right] = 1 \qquad (3.6)$$

2. **Perfect Soundness:** For all adversaries $\mathcal{A}$ and $x \notin \mathscr{L}$ the following holds:

$$\Pr\left[\mathsf{Verify}(\mathsf{gk},\mathsf{crs},x^*,\pi) = \mathsf{accept} \mid \begin{array}{c}(\mathsf{gk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\ell)\\ (\mathsf{crs}) \leftarrow \mathsf{Kgen}(\mathsf{gk},\mathsf{sk})\\ (x^*,\pi) \leftarrow \mathcal{A}(\mathsf{gk},\mathsf{crs})\\ x^* \notin \mathscr{L}\end{array}\right] = 0 \tag{3.7}$$

   If we consider the PPT adversary, we call the proof system with the computational soundness or an argument 3.3.5.

3. **Perfect Culpable Soundness:** In the original paper [143] they consider the cases, that may require soundness against the adversary who generates a valid proof for $x^* \in \mathscr{L}_{\mathsf{guilt}}$ instead of $x^* \in \bar{\mathscr{L}}$, where $\mathscr{L}_{\mathsf{guilt}}$ may depend on gk and crs. Based on this modification, they provide an alternate definition called culpable soundness. Note that if we put $\mathscr{L}_{\mathsf{guilt}} := \bar{\mathscr{L}}$, we get the original soundness definition as above. Formally:

$$\Pr\left[\mathsf{Verify}(\mathsf{gk},\mathsf{crs},x^*,\pi) = \mathsf{accept} \mid \begin{array}{c}(\mathsf{gk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\ell)\\ (\mathsf{crs}) \leftarrow \mathsf{Kgen}(\mathsf{gk},\mathsf{sk})\\ (x^*,\pi) \leftarrow \mathcal{A}(\mathsf{gk},\mathsf{crs})\\ x^* \notin \mathscr{L}_{\mathsf{guilt}}\end{array}\right] = 0 \tag{3.8}$$

4. **Indistinguishability:** The standard definition of witness indistinguishability requires that proofs computed on different witnesses for the same instance are computationally indistinguishable. For composable witness indistinguishability we need to use the idea of a simulated CRS to generates a simulated common reference string that is indistinguishable from a real one. Hence we first define the CRS indistinguishability.

   i. **CRS Indistinguishability** requires that there exists a PPT simulator Sim such that the advantage of any PPT adversary is negligible in the following experiment:

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ind-crs}}_{\mathcal{A}}(\ell) = \Big|&\Pr\Big[\mathcal{A}(\mathsf{crs},\mathsf{gk}) = 0 \mid (\mathsf{gk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\ell), \mathsf{crs} \leftarrow \mathsf{Kgen}(\mathsf{gk},\mathsf{sk})\Big] - \\ &\Pr\Big[\mathcal{A}(\mathsf{crs}^*,\mathsf{gk}) = 1 \mid (\mathsf{gk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\ell), \mathsf{crs}^* \leftarrow \mathsf{Sim}(\mathsf{gk},\mathsf{sk})\Big]\Big| \\ &< \mathsf{negl}(\ell)\end{aligned}$$

   ii. **Witness Indistinguishability** states that, an adversary cannot distinguish between the proof for w_0 and the proof for w_1 more than a random guess on a simulated CRS. Formally the advantage of the adversary in the experiment $\mathsf{Exp}^{\mathsf{wi}}_{\mathcal{A}}(1^\ell)$ (See Figure 3.5) is negligible.

5. **Zero-Knowledge:** This property requires the adversary not to distinguish between a real and a simulated CRS. In addition, it involves the use of two simulators. One generates the crs along with a trapdoor,τ, and the other generates a proof using

FIGURE 3.5: Witness Indistinguishability Experiment

Challenger		***Adversary***
$(\mathsf{gk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\ell)$		
	$\mathcal{C}^{\mathsf{wi}} \xrightarrow{(gk,\mathsf{crs})} \mathcal{A}$	
		$(x, w_0, w_1) \leftarrow \mathcal{A}(\mathsf{gk},\mathsf{crs})$:
		$(\mathsf{gk}, x, w_0), (\mathsf{gk}, x, w_1) \in \mathrm{R}$
	$\mathcal{C}^{\mathsf{wi}} \xleftarrow{(x,w_0,w_1)} \mathcal{A}$	
$\beta \overset{\$}{\leftarrow} \{0,1\}$		
$\pi \leftarrow \mathsf{Prove}(\mathsf{gk},\mathsf{crs}, x, w_\beta)$		
	$\mathcal{C}^{\mathsf{wi}} \xrightarrow{\pi} \mathcal{A}$	
		$\beta' \leftarrow \mathcal{A}(\mathsf{gk},\mathsf{crs},\pi)$
Success probability:	$\mathsf{Succ}^{\mathsf{wi}}_{\mathcal{A}}(1^\ell) = \Pr[\,\beta = \beta'\,]$	

the simulated CRS, crs^* such that the adversary cannot distinguish between the real proof and the simulated proof.

$$\Big|\Pr\left[\begin{array}{l}(\mathsf{gk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\ell);\ \mathsf{crs} \leftarrow \mathsf{Sim}(\mathsf{gk},\mathsf{sk});\\ (x, w_0, w_1) \leftarrow \mathcal{A}(\mathsf{gk},\mathsf{crs}); \pi \leftarrow \mathsf{Prove}(\mathsf{gk},\mathsf{crs}, x, w_0)\end{array} : \mathcal{A}(\pi) = 1\ \wedge (\mathsf{gk}, x, w_0) \in \mathrm{R}\right] -$$
$$\Pr\left[\begin{array}{l}(\mathsf{gk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\ell);\ \mathsf{crs} \leftarrow \mathsf{Sim}(\mathsf{gk},\mathsf{sk});\\ (x, w_0, w_1) \leftarrow \mathcal{A}(\mathsf{gk},\mathsf{crs}); \pi \leftarrow \mathsf{Prove}(\mathsf{gk},\mathsf{crs}, x, w_1)\end{array} : \mathcal{A}(\pi) = 1\ \wedge (\mathsf{gk}, x, w_1) \in \mathrm{R}\right]\Big| = 0.$$

and,

$$\begin{aligned}&\Big|\Pr\Big[\,(\mathsf{gk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\ell)\mathsf{crs} \leftarrow \mathsf{Kgen}(\mathsf{gk},\mathsf{sk}) \mid \mathcal{A}(\mathsf{gk},\mathsf{crs}) = 1\Big]\\ &\Pr\Big[\,(\mathsf{gk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\ell);\ \mathsf{crs} \leftarrow \mathsf{Sim}(\mathsf{gk},\mathsf{sk}) \mid \mathcal{A}(\mathsf{gk},\mathsf{crs}) = 1\Big]\\ &< \mathsf{negl}(\ell).\end{aligned}$$

Moreover, for all non-uniform adversaries $\mathcal{A}$, it holds that:

$$\Big|\Pr\left[\begin{array}{l}(\mathsf{gk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\ell);\ \mathsf{crs} \leftarrow \mathsf{Sim}(\mathsf{gk},\mathsf{sk});\\ (x, w_0, w_1) \leftarrow \mathcal{A}(\mathsf{gk},\mathsf{crs}); \pi \leftarrow \mathsf{Prove}(\mathsf{gk},\mathsf{crs}, x, w_0)\end{array} \mid \mathcal{A}(\pi) = 1\ \wedge (\mathsf{gk}, x, w_0) \in \mathrm{R}\right] -$$
$$\Pr\left[\begin{array}{l}(\mathsf{gk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\ell);\ \mathsf{crs} \leftarrow \mathsf{Sim}(\mathsf{gk},\mathsf{sk});\\ (x, w_0, w_1) \leftarrow \mathcal{A}(\mathsf{gk},\mathsf{crs}); \pi \leftarrow \mathsf{Prove}(\mathsf{gk},\mathsf{crs}, x, w_1) :\end{array} \mid \mathcal{A}(\pi) = 1\ \wedge (\mathsf{gk}, x, w_1) \in \mathrm{R}\right]\Big| = 0.$$

3.10 Groth Sahai NIWI proof System

Groth-Sahai scheme Π^{GS} is a NIWI-proof system under a trusted setup (i.e., in the CRS model) for the satisfiability of four types of equations (Figure 3.7) over bilinear groups.

FIGURE 3.6: $\mathsf{Exp}^{\mathsf{sim-zk}}_{\mathcal{A}}(1^\ell)$

Challenger		***Adversary***
$(\mathsf{gk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\ell)$		
$(\mathsf{crs},\tau) \leftarrow \mathsf{Sim}_1(\mathsf{gk},\mathsf{sk})$		
	$\mathcal{C}^{\mathsf{zk}} \xrightarrow{(gk,\mathsf{crs})} \mathcal{A}$	
		$(x,w) \leftarrow \mathcal{A}(\mathsf{gk},\mathsf{crs},\tau)$
		$(\mathsf{gk},x,w) \in \mathrm{R}$
	$\mathcal{C} \xleftarrow{(x,w)} \mathcal{A}$	
$\beta \xleftarrow{\$} \{0,1\}$		
If $\beta = 1 : \big(\pi \leftarrow \mathsf{Prove}(\mathsf{gk},\mathsf{crs},x,w)\big)$		
If $\beta = 0 : \big(\pi \leftarrow \mathsf{Sim}_2(\mathsf{gk},\mathsf{crs},\tau)\big)$		
	$\mathcal{C}^{\mathsf{wi}} \xrightarrow{\pi} \mathcal{A}$	
		$\beta' \leftarrow \mathcal{A}(\mathsf{gk},\mathsf{crs},\pi)$
Success probability:	$\mathsf{Succ}^{\mathsf{zkp}}_{\mathcal{A}}(1^\ell) = \Pr\left[\,\beta = \beta'\,\right]$	

3.10.1 Groth-Sahai Technique; Overview

The GS proof system is developed using commitment schemes, group isomorphism that preserves group actions, and a bilinear map. First we give an overview of the technique.

Assume that isomorphisms ι_s and ρ_s exist between groups $\mathbb{A}_s$ and $\mathbb{B}_s$:

$$s \in \{1,2,T\} \,:\, \iota_s : \mathbb{A}_s \mapsto \mathbb{B}_s\,,\, \rho_s : \mathbb{B}_s \mapsto \mathbb{A}_s$$

These maps are designed in such a way that they preserve both the group actions and the bilinear map and have communicative property. Put simply; this means that as illustrated in Figure 3.8, there are two ways to get to point $a_T = \mathbf{e}(a_1,a_2) \in \mathbb{A}_T : a_1 \in \mathbb{A}_1, a_2 \in \mathbb{A}_2$: one through groups $\mathbb{A}_1$ and $\mathbb{A}_2$ and evaluating the bilinear map over (a_1,a_2) the red path, and the other through $\mathbb{B}_1,\mathbb{B}_2$ the green path.

Now in order to provide proof for $(x,w) \in \mathrm{R}^{\mathsf{Eq}}_{GS}$ where:

$$\mathsf{Eq} : \mathbf{e}(\mathcal{X},\beta)\cdot\mathbf{e}(\alpha,\mathcal{Y}) = t_T, \tag{3.9}$$

the prover first commits to the witness components $\mathsf{Com}_1(\mathcal{X}) \in \mathbb{B}_1, \mathsf{Com}_2(\mathcal{Y}) \in \mathbb{B}_2$, which technically means that we transfer the members of $\mathbb{A}_1,\mathbb{A}_2$ to $\mathbb{B}_1,\mathbb{B}_2$, respectively.

The prover must demonstrate that the committed values $(\mathsf{Com}_1,\mathsf{Com}_2)$ satisfy the equation in the second step. This part can be proven using the commutative property. Because the verifier only needs to perform some computations on the committed and constant values in groups $\mathbb{B}_1$, $\mathbb{B}_2$, and $\mathbb{B}_T$ to determine whether the target value b_T is the image of the target value $a_T \in \mathbb{A}_T$.

FIGURE 3.7: The Set of Equations over bilinear groups supported by Groth-Sahai NIWI Proof System:

- Setting:

$$\mathcal{G} = \left(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathbf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{G}_T\right) : |\mathbb{G}_1| = |\mathbb{G}_2| = p$$

- Variables:

$$\vec{\mathcal{X}} = (\mathcal{X}_1, \dots, \mathcal{X}_m) \in \mathbb{G}_1^m \,,\ \vec{\mathcal{Y}} = (\mathcal{Y}_1, \dots, \mathcal{Y}_n) \in \mathbb{G}_2^n,$$
$$\vec{x} = (x_1, \dots, x_{m'}) \in \mathbb{Z}_n^{m'} \,,\ \vec{y} = (y_1, \dots, y_{n'}) \in \mathbb{Z}_n^{n'}$$

- Constants:

$$\vec{\mathcal{A}} = (\mathcal{A}_i) \in \mathbb{G}_1 \,,\ \vec{\mathcal{B}} = (\mathcal{B}_i) \in \mathbb{G}_2 \,,$$
$$\Gamma = \{\gamma_{ij}\}_{i,j} \in \mathbb{Z}_n \,,\ \mathcal{T}_1 \in \mathbb{G}_T, t_T \in \mathbb{G}_T$$

- Set of equations: $\mathcal{GS}^{\mathsf{Eq}} = \{\mathsf{Eq}^{\mathsf{pp}}, \mathsf{Eq}^{\mathsf{ms}}, \mathsf{Eq}^{\mathsf{qe}}\}$,
- $\mathsf{Eq}^{\mathsf{pp}}$: Pairing product equation

$$\prod_{i=1}^{n} \mathbf{e}(\mathcal{A}_i, \mathcal{Y}_i) \cdot \prod_{i=1}^{m}\prod_{j=1}^{m} \mathbf{e}(\mathcal{X}_i, \mathcal{X}_j)^{\lambda_{ij}} \cdot \prod_{i=1}^{m} \mathbf{e}(\mathcal{X}_i, B_i) = t_T = \mathbf{e}(R, S)$$
$$(\vec{\mathcal{A}} \cdot \vec{\mathcal{Y}})(\vec{\mathcal{X}} \cdot \Gamma\vec{\mathcal{Y}})(\vec{\mathcal{X}} \cdot \vec{\mathcal{B}}) = t_T = \mathbf{e}(R, S)$$

- $\mathsf{Eq}^{\mathsf{ms}}$: Multi-scalar multiplication equation in $\mathbb{G}_1$:

$$\sum_{i=1}^{n'} y_i \mathcal{A}_i + \sum_{i=1}^{m}\sum_{j=1}^{n'} \gamma_{ij} y_j \mathcal{X}_i + \sum_{i=1}^{m} b_i \mathcal{X}_i = \mathcal{T}$$
$$(\vec{\mathcal{A}} \cdot \vec{y}) + (\vec{\mathcal{X}}.\Gamma\vec{\mathcal{Y}}) + (\vec{\mathcal{X}}.\vec{b}) = \mathcal{T}$$

- $\mathsf{Eq}^{\mathsf{qe}}$: Quadratic equation in $\mathbb{Z}_n$:

$$\sum_{i=1}^{n} a_i y_i) + \sum_{i=1}^{m'}\sum_{j=1}^{n'} \gamma_{ij} x_i y_j + \sum_{i=1}^{m'} x_i b_i = t \mod n$$
$$(\vec{a} \cdot \vec{y})(\vec{x} \cdot \Gamma\vec{y}) + \vec{x} \cdot \vec{b} = t \mod n$$

- $\Pi^{\mathsf{niwi-GS}} = \langle \mathsf{Prove}_{\mathbb{GS}}, \mathsf{Verify}_{\mathbb{GS}} \rangle$: Groth-Sahai NIWI Proof System for relation R_{GS}

$\mathrm{R}_{GS} = \{(x, w) :$

$x = \mathsf{Eq} \in \mathcal{GS}^{\mathsf{Eq}}$,

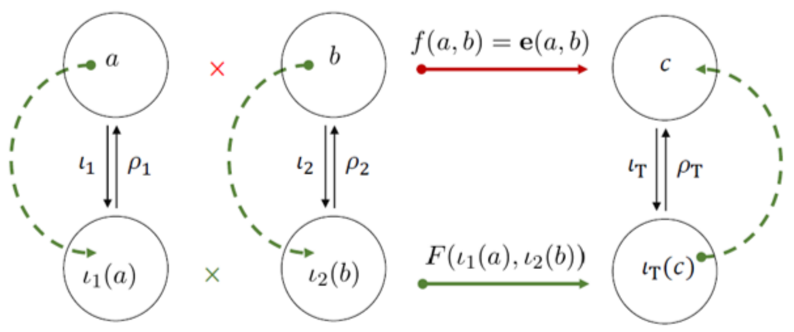
$w = (\{\alpha_i\} : \alpha_i \in (\mathbb{G}_1 \cup \mathbb{G}_2, \mathbb{G}_T)$

$\mathsf{Eq}[w] = \mathsf{Eq}[\{g_i\}] = \mathsf{True}$

$\}$

- Notation: By $\mathsf{Eq}[w] = \mathsf{True}$ we mean that w satisfies the equation. If we need to specify the set of solution (witness) for a specific equation, Eq, we present it by $\mathrm{R}_{GS}^{\mathsf{Eq}}$.

FIGURE 3.8: Commutative Diagram between $\mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_T$ and $\mathbb{B}_1, \mathbb{B}_2, \mathbb{B}_T$



$$F(\iota_1(a_1), \iota_2(a_2)) = \iota_T(f(a_1,a_2))$$
$$f(\rho_1(b_1), \rho_2(b_2)) = \rho_T(F(b_1,b_2))$$

3.10.2 Formal Description

We now move to a formal description of the Groth-Sahai NIWI proof system. We stress that, we explain the proof system for pairing product equation in the DLin setting, since we will use this instantiation in our research. For more details on other equations, we refer to the original paper. The content in this part is taken from [143].

We call an abelian group $(\mathbb{A}, +, 0)$ a $\mathcal{R}$-module for the finite commutative ring $(\mathcal{R}, +, ., 0, 1)$ if there exist a scalar multiplication, that maps $(r, x) \in \mathcal{R} \times \mathbb{A}$ to an element of $rx \in \mathbb{A}$, with the following properties:

$$(r+s)x = rx + sx, \quad r(x+y) = rx + ry, \quad r(sx) = (rs)x, \; 1x = x.$$

For example, we can mention a prime cyclic group $\mathbb{G}$, $|\mathbb{G}| = p$, which can be considered a $\mathbb{Z}_p$-module.

Now for the commutative ring $\mathcal{R}$ and $\mathcal{R}$-modules $\mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_T$ equipped with bilinear map $f : \mathbb{A}_1 \times \mathbb{A}_2 \mapsto \mathbb{A}_T$ a quadratic equations over variables $x_1, \ldots x_m \in \mathbb{A}_1$ and $y_1, \ldots, y_n \in \mathbb{A}_2$ has the following form:

$$\sum_{j=1}^{n} f(a_j, y_j) + \sum_{i=1}^{m} \sum_{j=1}^{n} \gamma_{ij} f(x_i, y_j) + \sum_{i=1}^{m} f(x_i, b_i) = t$$

where $a_1, \ldots a_n \in \mathbb{A}_1, b_1, \ldots b_m \in \mathbb{A}_2$ and $\gamma_{ij} \in \mathcal{R}$.

In order to avoid the heavy notation, we define

$$\vec{a} = (a_1 \ldots, a_n) \in \mathbb{A}_1^n, \; \vec{b} = (b_1, \ldots b_m) \in \mathbb{A}_2^m, \Gamma = [\gamma_{ij}]_{n \times m} \in \mathcal{R}^{n \times m}.$$

As a result, we will obtain

$$\vec{a} \cdot \vec{y} + \vec{x} \cdot \Gamma \cdot \vec{y} + \vec{x} \cdot \vec{b} = t,$$

where $\vec{x} \cdot \vec{y} = \sum_{i=1}^{n} f(x_i, y_i)$.

Commitment from Modules: Following the Groth-Sahai technique, to commit to the elements from $\mathcal{R}$-module $\mathbb{A}$, we first define homomorphisms ($\mathcal{R}$-linear)

$$\iota : \mathbb{A} \mapsto \mathbb{B} \text{ and } \rho : \mathbb{B} \mapsto \mathbb{A},$$

then we take $u_1, \dots, u_{\hat{m}} \xleftarrow{\$} \mathbb{B}$ and we let U be the space generated by elements $u_1, \dots, u_n$ *i.e.,* $U = \langle u_1, \dots, u_{\hat{m}} \rangle$. In fact, the public key for the commitment scheme will describe the $\mathcal{R}$-modulo $\mathbb{B}$ and these two homomorphisms. We require that operations in $\mathbb{B}$ and computation of the map ι are efficiently computable, but ρ is hard to compute.

Then to commit $x \in \mathbb{A}$, the algorithm picks $\hat{m}$ value $\mathsf{r}_1, \dots, \mathsf{r}_{\hat{m}} \xleftarrow{\$} \mathcal{R}$ and output the following value as the commitment:

$$\mathsf{Com} = \iota(x) + \sum_{i=1}^{\hat{m}} \mathsf{r}_i u_i$$

Notation. To simplify our notation to present the commitment to elements

$$x_1, \dots, x_m \in \mathbb{A},$$

we will write

$$\vec{c} = \iota_1(\vec{x}) + R\vec{u} \tag{3.10}$$

where

$$R \in Mat_{m\times\hat{m}}(\mathcal{R}) \ , \ \mathsf{Com}_i = \iota(x_i) + \sum_{j=1}^{\hat{m}} \mathsf{r}_{ij} u_j.$$

• **Commitment keys:** This commitment scheme has two types of commitment keys:

- **Binding key** defines $(\mathbb{B}, \iota, \rho, u_1, \dots, u_{\hat{m}})$ where $\rho(u_i) = 0$ and $\rho \circ \iota$ is nontrivial for all $i = 1 \dots, \hat{m}$.

$$\begin{aligned}\forall i : \rho(u_i) = 0 \implies \rho(\mathsf{Com}) &= \rho(\iota(x) + \sum_{i=1}^{\hat{m}} \mathsf{r}_i u_i) = \\ &= \rho(\iota(x)) + \sum_{i=1}^{\hat{m}} \mathsf{r}_i \underbrace{\rho(u_i)}_{0} \\ &= \rho(\iota(x))\end{aligned} \tag{3.11}$$

  Hence the non-trivial information inside the commitment, $\rho(\iota(x))$, make the commitment is perfectly binding to x.

- **Hiding key** defines $(\mathbb{B}, \iota, \rho, u_1, \dots, u_{\hat{m}})$ where $\iota(\mathbb{A}) \subseteq \langle u_1, \dots, u_{\hat{m}} \rangle$:

$$\begin{aligned}x \in \mathbb{A}\ \exists \alpha_i \in \mathcal{R} : \iota(x) = \sum_{i=1}^{\hat{m}} \alpha_i u_i \implies \mathsf{Com} &= \rho(\iota(x) + \sum_{i=1}^{\hat{m}} \mathsf{r}_i u_i) = \\ &= \sum_{i=1}^{\hat{m}} \alpha_i u_i + \sum_{i=1}^{\hat{m}} \mathsf{r}_i(u_i) \\ &= \sum_{i=1}^{\hat{m}} (\alpha_i + \mathsf{r}_i) u_i\end{aligned} \tag{3.12}$$

  therefore, Com perfectly hides the element x since $\mathsf{r}_1, \dots, \mathsf{r}_i$ are chosen at random from $\mathcal{R}$.

3.10.3 Groth-Sahai NIWI Proofs

We now outline how to prove the satisfiability of pairing product equations and in Section 3.11 we briefly present the Groth-Sahai proof system under the Subgroup Decision assumption and DLIN assumptions for pairing product equations.

FIGURE 3.9: Commutative Diagram for Groth-Sahai Proofs in DLin setting

$$\begin{tikzcd}
\mathbb{A}_1 = \mathbb{G} \arrow[d, shift right, "\iota_1"'] & \times & \mathbb{A}_2 = \mathbb{G} \arrow[d, shift right, "\iota_2"'] \arrow[rr, "f(x,y)"] & & \mathbb{A}_T = \mathbb{G}_T \arrow[d, shift right, "\iota_T"'] \\
\mathbb{B}_1 = \mathbb{G}^3 \arrow[u, shift right, "\rho_1"'] & \times & \mathbb{B}_2 = \mathbb{G}^3 \arrow[u, shift right, "\rho_2"'] & & \mathbb{B}_T = \mathbb{G}^9 \arrow[u, shift right, "\rho_T"'] \arrow[ll, "F"]
\end{tikzcd}$$

3.10.4 Set Up

In the setup algorithm of the Groth-Sahai Proof System, the CRS contains the commitment keys:

$$\begin{aligned}
\mathsf{Com}^1_{\mathsf{key}} &= (\iota_1, \rho_1, \mathbb{B}_1, U = \langle u_1, \ldots, u_{\hat{m}} \rangle), \\
\mathsf{Com}^2_{\mathsf{key}} &= (\iota_2, \rho_2, \mathbb{B}_2, V = \langle v_1, \ldots, v_{\hat{n}} \rangle), \\
\mathsf{Com}^T_{\mathsf{key}} &= (\iota_T, \rho_T, \mathbb{B}_T),
\end{aligned}$$

to commit to element in $\mathbb{A}_1$ ($\mathbb{A}_2$).

Considering the above commitment scheme, the CRS and gk define the following parameters:

$$\begin{aligned}
\mathsf{gk} &\mapsto (\mathcal{R}, \mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_T, f), \\
\mathsf{crs} &\mapsto \left(\mathsf{Com}^2_{\mathsf{key}}, \mathsf{Com}^T_{\mathsf{key}}, \mathsf{Com}^T_{\mathsf{key}}, H_1, \ldots H_k\right)
\end{aligned}$$

It is required that the maps are commutative as described in Figure 3.9 and except ρ_1, ρ_2, ρ_T all maps are efficiently computable. To avoid confusion, we present the action group in groups $\mathbb{B}_1, \mathbb{B}_2, \mathbb{B}_T$ with $\bullet$ which results to:

$$\vec{x} \in \mathbb{B}_1^n, \vec{y} \in \mathbb{B}_2^n \implies \vec{x} \bullet \vec{y} = \sum_{i=1}^{n} F(x_i, y_i)$$

The final part of the CRS is a set of matrices $H_1, \ldots, H_k \in Mat_{\hat{m} \times \hat{n}}(\mathcal{R})$ that all satisfy $\vec{u} \bullet H\vec{v} = 0$. The exact number k depends on the concrete setting.

Now we present two different settings:

Soundness setting: In this setting, we have the binding commitment keys:

$$\rho_1(\vec{u}) = 0, \rho_2(\vec{v}) = 0$$

and the maps $\rho_1 \circ \iota_1$, $\rho_2 \circ \iota_2$ and $\rho_T \circ \iota_T$ are non-trivial.

Witness Indistinguishability Setting. In this setting we have the hiding commitment keys:

$$\iota_1(\mathbb{A}_1) \subseteq \langle u_1, \dots, u_{\hat{m}} \rangle, \iota_2(\mathbb{A}_2) \subseteq \langle v_1, \dots, v_{\hat{n}} \rangle,$$

and also $H_1, \dots, H_k$ generate the $\mathcal{R}$-module of all matrices $H \in Mat_{\hat{m} \times \hat{n}}(\mathcal{R})$ such that:

$$\vec{u} \bullet H\vec{v} = 0.$$

Considering the above setting, the first step in of Groth-Sahaiproof techniques is to commit to all the variables $\vec{x}$ and $\vec{y}$, $\vec{c} = \iota_1(\vec{x}) + R\vec{u}, \vec{d} = \iota_2(\vec{y}) + S\vec{v}$ with $R \in Mat_{m \times \hat{m}}(\mathcal{R})$.

The second step is to show that these $\vec{c}$ and $\vec{d}$ are committed to the values that satisfy the equation.

Formal Description. Consider the relation R_{GS} with the equation 3.13 that has variables $\vec{x}$ and $\vec{y}$ from groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

$$\vec{a} \cdot \vec{y} + \vec{x} \cdot \Gamma \cdot \vec{y} + \vec{x} \cdot \vec{b} = t \tag{3.13}$$

Prover and the verifier perform the following computations:

- **Prover:** Choose random matrix $T \xleftarrow{\$} Mat_{\hat{n} \times \hat{m}}(\mathcal{R})$ and $\mathsf{r}_1, \dots, \mathsf{r}_\eta \xleftarrow{\$} \mathcal{R}$. Obtain:

$$\vec{\pi} := R^\top \iota_2(\vec{b}) + R^\top \Gamma \iota_2(\vec{y}) + R^\top \Gamma S \vec{v} - T^\top \vec{v} + \sum_{i=1}^{\eta} \mathsf{r}_i H_i \vec{v}$$
$$\vec{\theta} := S^\top \iota_1(\vec{a}) + S^\top \Gamma^\top \iota_1(\vec{x}) + T\vec{u}$$

  and return the proof $(\vec{\pi}, \vec{\theta})$

- **Verifier**: Return 1 if and only if:

$$\iota_1(\vec{a}) \bullet \vec{d} + \vec{c} \bullet \iota_2(\vec{b}) + \vec{c} \bullet \Gamma \vec{d} = \iota_T(t) + \vec{u} \bullet \vec{\pi} + \vec{\theta} \bullet \vec{v}$$

Completeness, soundness (in the soundness setting) and witness-indistinguishable in the (in WI setting) are proved in [143].

3.11 Instantiation Based on the DLin Assumption

this section presents an instantiation of the Groth-Sahai NIWI-proof system based on the Decisional Linear assumption.

Recall that DLin states that given

$$(g, A = g^\alpha, B = g^\beta, C = g^{r\alpha}, D = g^{s\beta}, Z) \in \mathbb{G}^6$$

for random α, β, r, s it is hard to tell whether $Z = g^{r+s}$ or is random. (See formal definition in 6).

We now describe the proof system.

- **Equation in DLin Setting.**

Pairing product equations:

$$\mathcal{R} = \mathbb{Z}_p, \mathbb{A}_1 = \mathbb{A}_2 = \mathbb{G}, \mathbb{A}_T = \mathbb{G}_T, f(x,y) = \mathbf{e}(x,y) \ : (\vec{\mathcal{A}} \cdot \vec{\mathcal{Y}})(\vec{\mathcal{Y}} \cdot \Gamma \vec{\mathcal{Y}}) = t_T$$

Multi-scalar multiplication in $\mathbb{G}$:

$$\mathcal{R} = \mathbb{Z}_p,\ \mathbb{A}_1 = \mathbb{Z}_p, \mathbb{A}_2 = \mathbb{G},\ \mathbb{A}_T = \mathbb{G},\ f(x,\mathcal{Y}) = x\mathcal{Y}:\ \vec{a} \cdot \vec{\mathcal{Y}} + \vec{x}\vec{\mathcal{B}} + \vec{x} \cdot \Gamma \vec{\mathcal{Y}} = \mathcal{T}$$

Quadratic equations:

$$\mathcal{R} = \mathbb{Z}_p, \mathbb{A}_1 = \mathbb{Z}_p, \mathbb{A}_2 = \mathbb{Z}_p, \mathbb{A}_T = \mathbb{Z}_p, f(x,y) = xy \ : \ \vec{x} \cdot \vec{b} + \vec{x} \cdot \Gamma \vec{x} = t \tag{3.14}$$

- **Commitment Keys.** We will now describe how to commit to elements in $\mathbb{Z}_p$ and $\mathbb{G}$.

  1. The commitments will belong to the $\mathbb{Z}_p$-module $\mathbb{B} = \mathbb{G}^3$ formed by entry-wise action.
  2. For two integers $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p$ we define

  $$\mathcal{U} = g^{\alpha}, \mathcal{O} = 1_{\mathbb{G}}, \mathcal{V} = g^{\beta} \in \mathbb{G}.$$

  3. The commitment key is of the form

  $$u_1 = (A = g^{\alpha}, \mathcal{O}, g), u_2 = (\mathcal{O}, B = g^{\beta}, g), \begin{cases} u_3 = (A^r, B^s, g^{r+s}); & \text{Binding Setting} \\ u_3 = (A^r, B^s, g^{r+s-1}); & \text{Hiding Setting} \end{cases}$$

  And the two maps:

  $$\begin{aligned} \iota : \mathbb{G} \mapsto \mathbb{G}^3\,; & \qquad \iota(Y) = (0,0,Y), \\ \rho : \mathbb{G}^3 \mapsto \mathbb{G}\,; & \qquad \rho(Z_1, Z_2, Z_3) = Z_3 \cdot Z_1^{-\frac{1}{\alpha}} \cdot Z_2^{\frac{1}{\beta}}. \end{aligned}$$

  To commit to some $Y \in \mathbb{G}$ we pick three random numbers $\mathsf{r}_1, \mathsf{r}_2, \mathsf{r}_3$ and obtain

  $$\mathsf{Com}(Y) = \iota(Y) = \sum_{i=1}^{3} \mathsf{r}_i u_i.$$

  If u_1, u_2, u_3 are linearly independent we obtain a perfectly hiding commitment scheme:

  – **Perfectly hiding:**

  $$\begin{aligned} &\mathsf{Com} : \ \mathbb{G} \mapsto \mathbb{G}_3 \\ &\mathsf{Com}(Y) = (0,0,Y) + \mathsf{r}_1(A, \mathcal{O}, g) + \mathsf{r}_2(\mathcal{O}, B, g) + \mathsf{r}_3(A^r, B^s, g^{r+s-1}) \\ &\qquad = (A^{\mathsf{r}_1 + r\mathsf{r}_3}, B^{\mathsf{r}_2 + \mathsf{r}_3 s}, g^{\mathsf{r}_1 + \mathsf{r}_2 + \mathsf{r}_3(r+s-1)} + Y) \end{aligned}$$

  – **Perfectly binding:** In case of binding key $\rho \circ \iota = \mathbb{I}$

  $$\begin{aligned} &\mathsf{Com} : \ \mathbb{G} \mapsto \mathbb{G}_3 \\ &\mathsf{Com}(Y) = (0,0,Y) + \mathsf{r}_1(A, \mathcal{O}, g) + \mathsf{r}_2(\mathcal{O}, B, g) + \mathsf{r}_3(A^r, B^s, g^{r+s}) \\ &\qquad = (A^{\mathsf{r}_1 + r\mathsf{r}_3}, B^{\mathsf{r}_2 + \mathsf{r}_3 s}, g^{\mathsf{r}_1 + \mathsf{r}_2 + \mathsf{r}_3(r+s)} + Y) \end{aligned}$$

Which is the encryption of (Z_1, Z_2, Z_3) respect to BBS Linear-Encryption (See 2.7.2.3) with key $\mathsf{pk} = (\mathcal{U}, \mathcal{V}, \mathcal{H} = g^{\alpha\beta}), \mathsf{sk} = (\alpha, \beta)$.

To commit to an exponent, an integer, $x \in \mathbb{Z}_p$, we define, $u = u_3 + (0, 0, g), \iota(x) = xu$ and $\rho(c_1, c_2, c_3) = \log_g(c_3 - \frac{1}{\alpha}c_1 - \frac{1}{\beta}c_2)$ and then the commitment is $\mathsf{Com}(x) = wu + \mathsf{r}_1 u_1 + \mathsf{r}_2 u_2$.

- ***Common Reference String:*** We consider $(\mathbb{A}_1 = \mathbb{A}_2 = \mathbb{A}_T = \mathbb{Z}_p)$ for exponents, $(\mathbb{A}_1, \mathbb{A}_2 = \mathbb{Z}_p, \mathbb{A}_T = \mathbb{G}_T)$ for group elements and $(\mathbb{B}_1 = \mathbb{B}_2 = \mathbb{G}^3, \mathbb{B}_T = \mathbb{G}_T^9)$ for the target groups. Considering the original bilinear map $\mathbf{e} : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_T$ defined by the group generator $\mathcal{G}(1^\ell)$, we define the following bilinear maps:

$$\tilde{F}\left(\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}\right) = \begin{pmatrix} f(x_1,y_1) & f(x_1,y_2) & f(x_1,y_3) \\ f(x_2,y_1) & f(x_2,y_2) & f(x_2,y_3) \\ f(x_3,y_1) & f(x_3,y_2) & f(x_3,y_3) \end{pmatrix} :$$

$$F = (x, y) = \frac{1}{2}\tilde{F}(x, y) + \frac{1}{2}\tilde{F}(y, x).$$

where:

$$\begin{array}{ll} \text{For exponents:} & x, y \in \mathbb{Z}_p : f(x, y) = x \cdot y \mod p \\ \text{For group elements:} & x, y \in \mathbb{Z}_p : f(x, y) = \mathbf{e}(x, y) \end{array}$$

We use the notation $\bullet$ and $\tilde{\bullet}$ for F and $\tilde{F}$ respectively, as the underlying bilinear maps.

- Maps for each equation

  1. Pairing Product Equation:

$$\iota_T(z) : \mathbb{Z}_p \mapsto \mathbb{G}^9, \quad \iota_T(z) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & z \end{pmatrix}$$

$$\rho_T : \mathbb{G}^9 \mapsto \mathbb{Z}_p, \qquad \rho_T(\begin{pmatrix} z_{11} & z_{12} & z_{13} \\ z_{21} & z_{22} & z_{23} \\ z_{31} & z_{32} & z_{33} \end{pmatrix}) = z_{33} z_{13}^{-\frac{1}{\alpha}} z_{23}^{-\frac{1}{\beta}} \left(z_{31} z_{11}^{-\frac{1}{\alpha}} z_{21}^{-\frac{1}{\beta}}\right)^{-\frac{1}{\alpha}} \left(z_{32} z_{12}^{-\frac{1}{\alpha}} z_{22}^{-\frac{1}{\beta}}\right)^{-\frac{1}{\beta}}$$

  H-matrices for F:

$$H_1 = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, H_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix}, H_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$$

  2. Multi Scalar Equation:

  $\iota_T^{\mathsf{ms}}(\mathcal{Z}) : \mathbb{Z}_p \mapsto \mathbb{G}^9, \iota_T^{\mathsf{ms}}(\mathcal{Z}) = \tilde{F}(\iota'(1), \iota(\mathcal{Z})) = \tilde{F}(u, (\mathcal{O}, \mathcal{O}, \mathcal{Z}))$
  $\rho_T^{\mathsf{ms}} : \mathbb{G}^9 \mapsto \mathbb{Z}_p, \quad \rho_T^{\mathsf{ms}} = \mathbf{e}^{-1}(\rho_T(z))$ where $\mathbf{e}^{-1}(\mathbf{e}(g, \mathcal{Z})) := \mathcal{Z}$.

  In the soundness setting $\rho_T^{\mathsf{ms}} \cdot \iota_T^{\mathsf{ms}} = \mathbb{I}_{\mathbb{G}}$

  3. Quadratic Equation in $\mathbb{Z}_p$

$$\tilde{\iota_T}^{\mathsf{q}}(z) : \mathbb{Z}_p \mapsto \mathbb{G}^9,\ \iota_T^{\mathsf{q}}(\mathcal{Z}) = \tilde{F}(\iota'(1), \iota'(z))$$
$$\iota_T^{\mathsf{q}}(z) : \mathbb{Z}_p \mapsto \mathbb{G}^9,\quad \iota_T^{\mathsf{z.q}}(\mathcal{Z}) = \tilde{F}(\iota'(1), \iota'(z))$$
$$\rho_T^{\mathsf{q}} : \mathbb{G}^9 \mapsto \mathbb{Z}_p,\qquad \rho_T^{\mathsf{z.q}} = \log_{\mathbf{e}(g,g)}(\rho_T(z)).$$

In the soundness setting $\rho_T^{\mathsf{q}} \cdot \iota_T^{\mathsf{q}} = \mathbb{I}_{\mathbb{Z}_p}$

- **NIWI proof**

  1. **Setup:** $\mathcal{G}(1^{\ell}) \mapsto \mathsf{gk} = (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g)$
  2. **Soundness String** On input gk return crs := (u_1, u_2, u_3) for random integers $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^*$ and $r, s \xleftarrow{\$} \mathbb{Z}_p$

     $$u_1 = (A = g^{\alpha}, \mathcal{O}, g), u_2 = (\mathcal{O}, B = g^{\beta}, g), u_3 = (A^r, B^s, g^{r+s-1}) = ru_1 + su_2$$

  3. **Witness Indistinguishability String:** On input gk return crs := (u_1, u_2, u_3) for random integers $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^*$ and $r, s \xleftarrow{\$} \mathbb{Z}_p$

     $$u_1 = (A = g^{\alpha}, \mathcal{O}, g), u_2 = (\mathcal{O}, B = g^{\beta}, g), u_3 = (A^r, B^s, g^{r+s}) = ru_1 + su_2 - (\mathcal{O}, \mathcal{O}, g)$$

- **Prover**: On input x = gk, crs, E and $w = \vec{x}, \vec{\mathcal{Y}}$, the algorithm takes the following steps:

  1. Commit to the integers $\vec{x} \in \mathbb{Z}_p^m$ and the group elements $\vec{\mathcal{Y}} \in \mathbb{G}^n$ by randomly choosing $R \xleftarrow{\$} Mat_{m\times 2}(\mathbb{Z}_p)$ and $S \xleftarrow{\$} Mat_{n\times 3}(\mathbb{Z}_p)$ returns $\vec{c}, \vec{d}$:

     $$\vec{c} = \iota'(\vec{x}) + R\vec{v}, \vec{d} = \iota(\vec{\mathcal{Y}}) + S\vec{u}$$

  2. For each pairing product equation of the form, $(\vec{\mathcal{A}} \cdot \vec{\mathcal{Y}})(\vec{\mathcal{Y}} \cdot \Gamma\vec{\mathcal{Y}}) = t_T$ generates a proof using the map F and random integers $\mathsf{r}_1, \mathsf{r}_2, \mathsf{r}_3 \xleftarrow{\$} \mathbb{Z}_p$:

     $$\vec{\Phi} := S^{\top}\iota(\vec{\mathcal{A}}) + S^{\top}(\Gamma + \Gamma^{\top})\iota(\vec{\mathcal{Y}}) + S^{\top}\Gamma S(\vec{u}) + \sum_{i=1}^{3} \mathsf{r}_i H_i \vec{u}$$

     for each linear equation $\vec{\mathcal{A}} \cdot \vec{\mathcal{Y}} = t_T$, by using map $\tilde{F}$ generates the proof:

     $$\vec{\pi} = \vec{0}, \vec{\theta} = S^{\top}\iota(\vec{\mathcal{A}})$$

  3. For each multi-scalar multiplication equation generates a proof using the map F and random integers $\mathsf{r}_1, \mathsf{r}_2, \mathsf{r}_3 \xleftarrow{\$} \mathbb{Z}_p$ to build R, and we let R' be R 3.10 with an appended 0-row. The proof is:

     $$\vec{\Phi} := (R')^{\top}\iota(\vec{\mathcal{B}}) + (R')^{\top}(\Gamma)\iota(\vec{\mathcal{Y}}) + S^{\top}\iota'(\vec{a}) + S^{\top}\Gamma^{\top}\iota'(\vec{x}) + (R')^{\top}\Gamma S\vec{u} + \sum_{i=1}^{3} \mathsf{r}_i H_i \vec{u}.$$

- **Verifier:** The verifier needs to check if the following equation does hold on input (gk, crs), $\vec{c}, \vec{d}$ and proof Φ for each equation as follows:

  1. For each pairing product equation:

     $$\iota(\vec{\mathcal{A}}) \bullet \vec{d} + \vec{d} \bullet \Gamma\vec{d} = \iota_T(t_T) + \vec{u} \bullet \vec{\Phi}.$$

2. For each linear equation $\mathcal{A}\cdot\vec{\mathcal{Y}} = t_T$:
$$\iota(\vec{\mathcal{A}})\tilde{\bullet}\vec{d} = \iota_T(t_T) + \iota_T(t_T) + \iota(\vec{\phi})\tilde{\bullet}\vec{u}.$$
3. For multi-scalar multiplication:
$$\iota'(\vec{a}) \bullet \vec{d} + \vec{c} \bullet \iota(\vec{\mathcal{B}}) + \vec{c} \bullet \Gamma\vec{d} = \iota_T^{\mathsf{ms}}(\mathcal{T}) + \vec{u} \bullet \vec{\Phi}.$$
4. For each linear equation $\vec{a}\cdot\vec{\mathcal{Y}} = \mathcal{T}$:
$$\iota(\vec{a})'\tilde{\bullet}\vec{d} = \iota_T^{\mathsf{ms}}(\mathcal{T}) + \iota'(\vec{\Phi})\tilde{\bullet}\vec{u}.$$
5. For each linear equation $\vec{x}\cdot\vec{\mathcal{B}} = \mathcal{T}$:
$$\iota(\vec{c})\tilde{\bullet}\iota(\mathcal{B}) = \iota_T^{\mathsf{ms}}(\mathcal{T}) + \vec{v}\tilde{\bullet}\iota(\Phi).$$
6. For each quadratic equation:
$$\vec{c} \bullet \iota'(\vec{b}) + \vec{c} \bullet \Gamma\vec{c} = \iota'_T(t) + \vec{v} \bullet \vec{\Phi}.$$
7. For each linear equation $\vec{x}\cdot\vec{b} = t$:
$$\vec{c}\tilde{\bullet}\iota'(\vec{b}) = \iota'_T(t) + \vec{v}\tilde{\bullet}\iota(\vec{\Phi}).$$

Theorem 3.11.1. *[143] The above protocol is a NIWI proof with perfect completeness, perfect soundness and composable witness-indistinguishability for satisfiability of a set of equations over a bilinear group where the DLin problem is hard.*

Additional Note. In the improvement of Groth-Sahai technique in [96] they replace replace some of the commitments with ElGamal encryptions, which reduces the prover's computation and for some types of equations, reduces the proof size in SXDH setting 8, the one that we will use in section 8.

3.12 OR Statements

[1]Some of our relations of Section 5.4 consist of a generalized form of disjunction (OR) of two predicates, let us say P_1 and P_2. Suppose that we have equivalent systems of equations for each of the two predicates that are a system of equations E_1 (resp. E_2) representing predicate P_1 (resp. P_2). Consider the following relation:
$$\begin{aligned}\mathrm{R}_{OR} =\big\{&(x, w) \mid x = (\mathsf{E}_1, \mathsf{E}_2), w = (\mathsf{idx}, w_1, w_2) : \mathsf{idx} \in \{1,2\} \wedge \\ &(\mathsf{E}_{\mathsf{idx}}, w_{\mathsf{idx}}) \in \mathrm{R}_E \ \wedge\ w_{\bar{\mathsf{idx}}} \in \mathbb{G}^3\big\},\end{aligned}$$
where $\bar{\mathsf{idx}}$ means $\{1,2\}/\{\mathsf{idx}\}$.

Notice that the relation is not exactly a disjunction of pairing product equations because we need to make sure that the statement that holds is the one selected by the index in the witness, so we cannot use the technique of Groth [139] and we will follow a different approach.

[1]This section is part of our publication in [242]

By hypothesis, $\mathcal{P}_{\mathbb{GS}}$ takes as input a system of equations E as a statement and a solution $(g_1,\dots,g_m)$ as a witness and provides a NIWI-proof of membership of $(\mathsf{E}, w) \in \mathrm{R}_E$. Therefore, to use $\mathrm{NIWI}_{\mathbb{GS}}$ to generate a NIWI-proof for relation R_{OR}, we need to define a third system of equation E_{OR} with the following properties:

1. $\mathsf{E}_{\mathtt{OR}} \approx \mathrm{R}_{OR}$. With this notation, we mean that there are two efficiently computable functions f and g such that:

$$\exists w = (\mathsf{idx}, w_1, w_2)\ \big(x = (\mathsf{E}_1, \mathsf{E}_2), w\big) \in \mathrm{R}_{OR} \Leftrightarrow \exists \tilde{w}\ \big(\mathsf{E}_{\mathsf{OR}} = f(x), \tilde{w}\big) \in \mathrm{R}_E.$$

$$\big(x, w\big) \in \mathrm{R}_{OR} \Rightarrow \big(f(x), g(x,w)\big) \in \mathrm{R}_{OR}.$$

   The latter properties guarantee that a proof for relation R_{OR} computed using $\mathrm{NIWI}_{\mathbb{GS}}$ satisfies completeness and soundness. For WI to hold, we need the following property.

2. The function f is efficiently invertible.

Now we show how to construct the system of equations $\mathsf{E}_{\mathtt{OR}}$ with the properties above. Consider two systems of pairing product equations E_1 and E_2 - same structure as in 3.7. For simplicity, we assume the equations are over two variables (the general case is straightforward).

$$\mathsf{E}_1 : \mathbf{e}(\mathcal{X}_1, a_1)\cdot \mathbf{e}(\mathcal{X}_2, a_2) = \tau_1\ , \mathsf{E}_2 : \mathbf{e}(\mathcal{Y}_1, b_1)\cdot \mathbf{e}(\mathcal{Y}_2, b_2) = \tau_2$$

We define the new system of equations E_{OR} with 4 new variables $\mathcal{Z}_{11}, \mathcal{Z}_{12}, \mathcal{Z}_{21}, \mathcal{Z}_{22}$ as follows:

$$\mathsf{E}_{\mathsf{OR}} : \begin{cases} \mathbf{e}(\mathcal{X}_1, a_1)\cdot \mathbf{e}(\mathcal{X}_2, a_2)\cdot \mathbf{e}(\mathcal{Z}_{11}, \mathcal{Z}_{12}) = \tau_1 \\ \mathbf{e}(\mathcal{Y}_1, b_1)\cdot \mathbf{e}(\mathcal{Y}_2, b_2)\cdot \mathbf{e}(\mathcal{Z}_{21}, \mathcal{Z}_{22}) = \tau_2 \\ \mathbf{e}(\mathcal{Z}_{11}, \mathcal{Z}_{22}) = 1 \\ \mathbf{e}(\mathcal{Z}_{11}, g)\cdot \mathbf{e}(\mathcal{Z}_{\mathsf{idx}}, g) = \mathbf{e}(g,g) \\ \mathbf{e}(\mathcal{Z}_{22}, g)\cdot \mathbf{e}(\mathcal{Z}_{\mathsf{idx}}, g) = \mathbf{e}(g^2,g) \end{cases}$$

Analysis of the equations: Consider

$$(\mathcal{Z}_{\mathsf{idx}} \hookleftarrow g_{\mathsf{idx}}, \mathcal{X}_1 \hookleftarrow g_1, \mathcal{X}_2 \hookleftarrow g_2, \mathcal{Y}_1 \hookleftarrow g_3, \mathcal{Y}_2 \hookleftarrow g_4, \mathcal{Z}_{11} \hookleftarrow g_{11}, \dots, \mathcal{Z}_{22} \hookleftarrow g_{22})$$

as a solution for E_{OR}. So, there exist values $\mathsf{idx}, z_{11}, z_{22} \in \mathbb{Z}_p$ such that

$$g_{\mathsf{idx}} = g^{\mathsf{idx}}, g_{11} = g^{z_{11}}, g_{22} = g^{z_{22}}$$

and for $t \in [k]$ there exist values α_t such that $\tau_t = \mathbf{e}(g, \alpha_t)$.

- $\mathbf{e}(\mathcal{Z}_{11}, g)\cdot \mathbf{e}(\mathcal{Z}_{\mathsf{idx}}, g) = \mathbf{e}(g,g) \Rightarrow \mathbf{e}(g^{z_{11}+\mathsf{idx}-1}, g) = 1$
  $\Rightarrow z_{11} = 1 - \mathsf{idx}$ and similarly $z_{22} = 2 - \mathsf{idx}$.
- $\mathbf{e}(\mathcal{Z}_{11}, \mathcal{Z}_{22}) = 1 \Rightarrow (z_{11} = 0 \lor z_{22} = 0)$
- $z_{11} = 0 \land z_{11} = 1 - \mathsf{idx} \Rightarrow \mathbf{e}(\mathcal{X}_1 \hookleftarrow g_1, a_1)\cdot \mathbf{e}(\mathcal{X}_2 \hookleftarrow g_2, a_2) = \tau_1$
  $\Rightarrow (\mathsf{E}_1[g_1, g_2] = \mathsf{True} \land \mathsf{idx} = 1)$
- Similarly, $z_{22} = 0 \land z_{22} = 2 - \mathsf{idx}$
  $\Rightarrow \mathbf{e}(\mathcal{Z}_{21}, \mathcal{Z}_{22}) = 1 \Rightarrow (\mathsf{E}_2[g_3, g_4] = \mathsf{True} \land \mathsf{idx} = 2)$

The above facts imply that:

$$\begin{aligned}&\mathsf{E}_{\mathsf{OR}}[(g_{\mathsf{idx}}, g_1, \ldots, g_4, g_{11}, \ldots, g_{22})] = \mathsf{True} \Rightarrow \\ &\quad \Big((\mathsf{E}_1[g_1, g_2, \alpha_1] = \mathsf{True} \wedge \mathsf{idx} = 1\Big) \vee \Big(\mathsf{E}_2[g_3, g_4, \alpha_2] = \mathsf{True} \wedge \mathsf{idx} = 2)\Big),\end{aligned}$$

as it was to show. It is also easy to see that the previous transformation is efficiently invertible.

For the other direction, suppose w.l.o.g that $w_1 = (g_1, g_2, \alpha_1)$ is a solution to $x = \mathsf{E}_1$ (the other case is symmetrical, and we omit it), namely $(x, w_1) \in \mathrm{R}$. Suppose also that $w_2 = (g_3, g_4, \alpha_2) \in \mathbb{G}^3$ is an arbitrary triple of elements of $\mathbb{G}$. Therefore $(1, w_1, w_2)$ is a witness to $(\mathsf{E}_1, \mathsf{E}_2)$ with respect to relation R_{OR}. Then, setting

$$(\mathcal{Z}_{\mathsf{idx}} \hookleftarrow g^1, \mathcal{X}_1 \hookleftarrow g_1, \mathcal{X}_2 \hookleftarrow g_2, \mathcal{Y}_1 \hookleftarrow g^0, \mathcal{Y}_2 \hookleftarrow g^0, \mathcal{Z}_{11} \hookleftarrow g^0, \mathcal{Z}_{12} \hookleftarrow g^1, \mathcal{Z}_{21} \hookleftarrow \alpha_2, \mathcal{Z}_{22} \hookleftarrow g^1),$$

we have that:

$$\mathsf{E}_{\mathsf{OR}}[(g_{\mathsf{idx}}, g_1, \ldots, g_4, g_{11}, \ldots, g_{22})] = \mathsf{True}.$$

(Notice that we implicitly defined a transformation g as needed.)

OR proof in the general case. If the number of pairing products (m) in each of the two equations is greater than 1, such as:

$$\mathsf{E}_1 : \begin{cases}\mathbf{e}(\mathcal{X}_1, a_1) \cdot \mathbf{e}(\mathcal{X}_2, a_2) = \tau_1 \\ \mathbf{e}(\mathcal{X}_1, a'_1) \cdot \mathbf{e}(\mathcal{X}_2, a'_2) = \tau'_1\end{cases}, \; \mathsf{E}_2 : \begin{cases}\mathbf{e}(\mathcal{Y}_1, b_1) \cdot \mathbf{e}(\mathcal{Y}_2, b_2) = \tau_2 \\ \mathbf{e}(\mathcal{Y}_1, b'_1) \cdot \mathbf{e}(\mathcal{Y}_2, a'_2) = \tau'_2\end{cases}$$

then E_{OR} can be defined as:

$$\mathsf{E}_{\mathsf{OR}} : \begin{cases}\mathbf{e}(\mathcal{X}_1, a_1) \cdot \mathbf{e}(\mathcal{X}_1, a_2) \cdot \mathbf{e}(\mathcal{Z}_{11}, \mathcal{Z}_{12}) = \tau_1 \\ \mathbf{e}(\mathcal{X}_1, a'_1) \cdot \mathbf{e}(\mathcal{X}_2, a'_2) \cdot \mathbf{e}(\mathcal{Z}_{11}, \mathcal{Z}_{13}) = \tau'_1 \\ \mathbf{e}(\mathcal{Y}_1, b_1) \cdot \mathbf{e}(\mathcal{Y}_2, b_2) \cdot \mathbf{e}(\mathcal{Z}_{21}, \mathcal{Z}_{22}) = \tau_2 \\ \mathbf{e}(\mathcal{Y}_1, b'_1) \cdot \mathbf{e}(\mathcal{Y}_2, b'_2) \cdot \mathbf{e}(\mathcal{Z}_{23}, \mathcal{Z}_{22}) = \tau'_2 \\ \mathbf{e}(\mathcal{Z}_{11}, \mathcal{Z}_{22}) = 1 \\ \mathbf{e}(\mathcal{Z}_{11}, g) \cdot \mathbf{e}(\mathcal{Z}_{\mathsf{idx}}, g) = \mathbf{e}(g, g) \\ \mathbf{e}(\mathcal{Z}_{22}, g) \cdot \mathbf{e}(\mathcal{Z}_{\mathsf{idx}}, g) = \mathbf{e}(g^2, g)\end{cases}$$

Chapter 4

Perfect Inner Product Encryption Schemes

"One of the basic rules of the universe is that nothing is PERFECT."
Stephen Hawking

The path toward developing a functional encryption scheme capable of handling all polynomial-time predicates began with an Identity-Based Encryption scheme, and it has a steppingstone, ***Inner Product Encryption*** scheme.
In this section, we formalize the concept of the Inner product encryption scheme as a particular case of the functional encryption scheme. We begin by reviewing the definition and various security concepts in the context of a functional encryption scheme, and then we present our perfectly correct inner product encryption scheme.

Contents

4.1 Functional Encryption Scheme

Functional encryption is a new encryption paradigm that emerged from a series of refinements of the traditional encryption concept, namely, the "all-or-none" approach toward "fine-grained access" to information. In other words, it enables the data owner to control how much of the data each user can access in a decentralized fashion. The concept of Functional encryption (FE for short) was first proposed by Sahai and Waters [229] and formalized independently by Boneh, Sahai and Waters [55] and O'Neil [214], and later, new perspectives on security definitions for functional encryption, presented in [14, 133].

4.1.1 Introduction

Informally in an FE scheme, the holder of the master secret key (MSK) can issue tokens associated with some function f. Later this token, tok_f, allows a user to learn the function of the original message, $f(m)$ rather than the message m itself. Furthermore, the security notion of FE guarantee that only $f(m)$ is learnt about m and nothing else.

As with classical encryption, functional encryption is classified into two broad categories: *private settings*, also known as symmetric FE, which use a single master key MSK, and *public-key settings*, which use a pair of master keys $(\mathsf{MPK}, \mathsf{MSK})$, the public master key and the secret master key.

In public-key FE, anyone can encrypt the message using the master public key, while only the owner of the secret token can retrieve an evaluation of the original message. In contrast, there is no master public key in the private setting, and the encryption algorithm also requires some secret input to encrypt the message [23, 24]. Therefore, we stress that our research mainly focuses on the public-key setting for the functional encryption scheme.

Prior to the formal introduction of the functional encryption scheme, there had been cryptosystems that may be considered special cases of Functional encryption schemes.

It is straightforward to see that, Identity-Based Encryption [53, 81, 234], Hidden Vector Encryption, public-key encryption with keyword search [59, 6], attribute-based encryption [18, 229, 47, 134], and predicate encryption [56, 189, 173, 213], can be elegantly expressed as a functional encryption scheme with particular functionalities.

Regardless of the examples given above, achieving generic functional encryption that provides an adequate level of security for any functionality is a complex and difficult research subject. Some existing constructions are based on strong and impractical assumptions, while others restrict the security notion.

For instance, the FE proposed in [114, 116] and [62, 115] is based on multi-linear maps and indistinguishability obfuscation, respectively. Additionally, as an example of restriction security notion, we can mention FE schemes in [130, 133, 228], which put an upper bound on the number of tokens that the adversary can query. As a result of the above argument, we can conclude that investigating functional encryption schemes with specific functionality would be more promising.

Outline. In Section 4.2 we give a formal introduction of a functional encryption system and its security notion. Following that, in Section 4.3 we present a brief overview of some sub-classes of FE. In Section 4.4, we formally introduce the inner product encryption scheme. In Section 4.5 we present a technical overview of our research process toward building a perfect correct IPE. This section explains our study methodology by highlighting the challenge involved with perfect correctness. Following the technical overview, we present our perfect internal product encryption scheme in Section 4.6. Finally, this chapter ends with a security proof of our system based on the standard assumptions in Section 4.7. We would like to emphasize that the research presented in this chapter was previously published in (PKC-2020) [242][1].

4.2 Functional Encryption; Formal Definition

For $\ell \in \mathbb{N}$, let $\mathcal{F} = \{\mathcal{F}_\ell\}_\ell$, $\mathcal{K} = \{\mathcal{K}_\ell\}_\ell, \mathcal{M} = \{\mathcal{M}_\ell\}_\ell$ and $\mathcal{R} = \{\mathcal{R}_\ell\}_\ell$ denote ensembles of finite sets with index of security parameter $\ell \in \mathbb{N}$. The functionality ensemble $\mathcal{F} = \{\mathcal{F}_\ell\}_\ell$ is defined over $(\mathcal{K}_\ell, \mathcal{M}_\ell)$ which is a collection of functions f described as a (deterministic) Turing machine:

$$f \in \mathcal{F}_\ell\,,\; f : \mathcal{K}_\ell \times \mathcal{M}_\ell \mapsto \mathcal{R}_\ell.$$

The ensemble $\mathcal{K} = \{\mathcal{K}_\ell\}$ is called the ***key space***, and $\mathcal{M}_\ell$ is called the ***message space***. We define the functional encryption scheme concerning to the functionality $\mathcal{F}$ as follows:

Definition 30 (**Functional Encryption Scheme**). *A functional encryption scheme (FE) for the class of functionality $\mathcal{F}$ is a tuple of PPT algorithms* $\Pi^{\mathsf{FE}} = \langle \mathsf{Setup}, \mathsf{TokGen}, \mathsf{Enc}, \mathsf{Dec} \rangle$, *detailed as below:*

- ***Set up*** *is a probabilistic algorithm that takes security parameters and generates a pair of master public key and master secret key:*

$$(\mathsf{MPC}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\ell).$$

- ***Token generator*** *is a probabilistic algorithm that on input master keys and the function $f \in \mathcal{F}_\ell$, output the token* tok_f*:*

$$\mathsf{tok}_f \leftarrow \mathsf{TokGen}(\mathsf{MSK}, f).$$

- ***Encryption*** *is a probabilistic algorithm that encrypts message $m \in \mathcal{M}_\ell$ with respect to the master public key:*

$$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{MPK}, m).$$

- ***Decryption*** *is a deterministic algorithm that takes a ciphertext,* ct*, function $f \in \mathcal{F}_\ell$ and its secret token,* tok_f*, as input and outputs a value from the range set or a symbol $\perp$ as a sign of the failure:*

$$\mathsf{Dec}(\mathsf{ct}, \mathsf{tok}_f) \mapsto y \in \mathcal{R}_\ell \textit{ Or } \perp.$$

Security Properties. A functional encryption scheme requires to have the following properties:

- ***Correctness:*** The output of the decryption algorithm is evaluation of the original message, m, for all function $f \in \mathcal{F}_\ell$, except with negligible probability over the randomness of set up , token-generation and encryption algorithms:

$$\Pr\left[\mathsf{Dec}\big(\mathsf{tok}_f, f, \mathsf{ct}\big) = F(f, m) = f(m) \mid \begin{array}{c} (\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\ell) \\ \mathsf{ct} = \mathsf{Enc}(\mathsf{MPK}, m) \\ \mathsf{tok}_f \leftarrow \mathsf{TokGen}(\mathsf{MSK}, f) \end{array} \right] \geq 1 - \mathsf{negl}(\ell)$$

[1]The International Conference on Practice and Theory of Public-Key Cryptography (PKC)

- ***Security:*** Intuitively, FE's security captures that given the encrypted message m and the token for function f, the only information the adversary learned is evaluation $f(m)$. In addition to the security requirement in traditional encryption, functional encryption needs to be secure in the presence of *key holder collusion*, which means that malicious users should not be able to combine their secret keys to obtain unauthorized information [14].

  Moreover, when it comes to security in the context of a functional encryption scheme, we need to consider the "security guarantee" and "the adversarial model", and the level of privacy regarding to the functionality and its token.

  In the following, we present the formal definition and framework for security notion in the context of functional encryption schemes in the various models.

4.2.1 Security Notions in the Context of FE

Two approaches capture the security notion for functional encryption scheme; *simulation-based security*, (denoted by SIM-based) and *indistinguishability-based security*, a.k.a., game-based security, (denoted by IND-based), both of which can be considered with the flavor of adaptive versus non-adaptive, one versus many and fully versus selectively secure [55, 133, 214].

4.2.1.1 Simulation-Based Security

Simulation-based security requires that an efficient algorithm, simulates the adversary's view. This simulator, only has access to the token-generator oracle and some function evaluation on the corresponding message, m. It does not, however, receive the message m as input. As a result, if the PPT adversary cannot distinguish between its real view and the simulation view provided by the simulation, it implies that the adversary cannot learn anything other than message evaluation because the simulator does not know the message m.

Simulation-based security notion also has different flavours; here, we give a formal definition for the many-message simulation-based security against an adaptive adversary, which is the strongest notion of security.

Consider two experiments in Figure 4.1; the left is the real game between the adversary, and the challenger and the right is between the adversary, and the simulator.

FIGURE 4.1: Many-message Simulation-based experiments against adaptive adversary

$\mathsf{Exp}^{\mathsf{real}}_{\mathcal{A}}(1^{\ell})$	$\mathsf{Exp}^{\mathsf{idea}}_{\mathcal{A}}(1^{\ell})$
1: $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^{\ell})$	1: $\mathsf{MPK} \leftarrow \mathsf{Sim}(1^{\ell})$
2: $(\{m_i\}_{i\in[l]}, \mathsf{st}) \leftarrow \mathcal{A}_1^{\rightleftharpoons \mathsf{TokGen}(\mathsf{MSK},.)}(\mathsf{MPK})$	2: $(\{m_i\}_{i\in[l]}, \mathsf{st}) \leftarrow \mathcal{A}_1^{\rightleftharpoons \mathsf{Sim}(.)}(\mathsf{MPK})$
3: $\mathsf{ct}_i \leftarrow \mathsf{FE.Enc}(\mathsf{MPK}, m)$ for all $i \in [l]$	3: $\mathsf{ct}_i \leftarrow \mathsf{Sim}^{\rightleftharpoons U_{m_i}}(1^{\ell}, 1^{\lvert m\rvert})$ for all $i \in [l]$
4: $\alpha \leftarrow \mathcal{A}_2^{\rightleftharpoons \mathsf{TokGen}(\mathsf{MSK},.)}(\mathsf{MPK}, \{\mathsf{ct}_i\}_{i\in[l]}, \mathsf{st})$	4: $\alpha \leftarrow \mathcal{A}_2^{\rightleftharpoons \mathsf{Sim}(.)}(\mathsf{MPK}, \{\mathsf{ct}_i\}_{i\in[l]}, \mathsf{st})$
5: Output $(\{m_i\}_{i\in[l]}, \alpha)$	5: Output $(\{m_i\}_{i\in[l]}, \alpha)$

We distinguish between these experiments and we have the following definition:

Definition 31. *[**Many-message Simulation-based experiments against adaptive adversary** [14]] Consider the functional encryption scheme Π^{FE} for a family of functionality $\mathcal{F}$. Let $\mathcal{U}_x(.)$ denote the universal oracle that:*

$$f \in \mathcal{F} : \mathcal{U}_m(f) \mapsto f(m).$$

Consider the two experiments in Figure 4.1 for a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a stateful PPT simulator Sim where $l = \mathsf{poly}(\ell)$ is a positive integer. The functional encryption scheme Π^{FE} is then said to be simulation-secure for many messages against adaptive adversaries if there exists an admissible stateful PPT simulator Sim such that for every PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the following two distributions are computationally indistinguishable:

$$\left\{\mathsf{Exp}^{\mathsf{FE-real}}_{\mathcal{A}}(1^{\ell})\right\}_{\ell\in\mathbb{N}} \approx \left\{\mathsf{Exp}^{\mathsf{FE-ideal}}_{\mathcal{A}}(1^{\ell})\right\}_{\ell\in\mathbb{N}}$$

Additional Note. Other versions for specifying simulation-based security for FE exist in the literature; In some of them, the simulator is given Oracle access to $\mathcal{A}_2$, allowing it to "rewind" the adversary and adaptively reconstruct the view. Furthermore, several definitions of the simulator rely on a "trapdoor" information obtained via faking the setup parameters. We refer to [55, 133, 214] for more detail.

Negative Result. Although the simulation-based secure functional encryption scheme guarantees a high level of security, the well-studied research has been shown that some functionality cannot have simulation-based security even with the weak variants:

Theorem 4.2.1. *There exist a circuit family $\mathscr{C}$ for which there is no 1-message simulation-based-secure functional encryption scheme against non-adaptive adversary. [14]*

It is worth mentioning that their simulation-based secure FE is achievable in the random oracle model; we refer to [55, 71, 159] for more on the subject.

Negative results in simulation-based secure FE led to establishing a more realistic notion for the security of a functional encryption scheme, namely IND-based security.

4.2.1.2 IND-Based Security

In IND-base security a.k.a. game-based security, the security's notion is captured through an interactive game between challenger $\mathcal{C}^{\mathsf{fe}}$ and a PPT adversary $\mathcal{A}$. The adversary attempts to guess some secret random bit chosen by the challenger, by inquiring about the secret token tok_{f_i} for the function f_i.

Definition 32. *A functional scheme* Π^{FE} *is* IND-CCA-*secure against adaptive adversary if for all PPT adversary, the advantage of* $\mathcal{A}$ *in the experiment,* $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ind-cpa}}(1^{\ell})$ *(Figure 4.2) is negligible.*

Variants of Security. Considering the definition 32, we can classify functional encryption scheme into the following categories:

- **CCA versus CPA:** IND-CPA secure, functional encryption scheme is identical to the IND-CCA except that the adversary does not access the decryption oracle in CPA-security. Precisely to adopt the experiment in Figure 4.2, we need to delete the decryption query in both phases 1 and 2.

  Additional Note. Agrawal *et al.* [11] proposed the first adaptively secure (IND-CPA) schemes under the learning with errors (LWE), DDH (See 5), and DCR (See 3) assumptions; In 2017, Zhang [264] and Benhamouda [41] proposed IND-CCA functional encryption schemes against active adversaries.

- **1 versus Many:** In definition 31 the adversary can query many, namely $l \in \mathbb{N}$ token where $l = \mathsf{poly}(\ell)$ for some polynomial poly. The weaker security notion is 1-message simulation-based security in which the adversary is allowed to output only one message, namely, $l = 1$.

- **Adaptive versus non-Adaptive:** If the adversary cannot query after receiving the ciphertext (skip step 4 in experiment 4.1) then the experiment captures the simulation-based security against non-adaptive adversaries.

- **Fully versus Selective** A weaker notion of IND-based security, so-called *selective security*, does not allow the adversary to choose m_0 and m_1 adaptively after inquiring about some token; instead, the adversary must choose the challenge messages before obtaining any token or seeing the master public key. As a result, the query phase 1 in experiment 4.2 is removed under this security model, and the challenge phase occurs either after or before the Setup phase.

  Additional Note. Although the IND-secure functional encryption scheme scheme in selective mode is weaker than the functional encryption scheme against an adaptive adversary, a transformation is proposed in [16] for converting a selectively secure FE to a fully secure FE.

- **Function hiding:** Intuitively, functional hiding means that a secret token tok_f does not reveal any more information on the function f except what is implicitly leaked by $f(m)$. We can highlight anonymous-IBE as a fascinating and practical example of functional hiding property.

4.2.1.3 Multi-Input FE

A more general notion of the functional encryption scheme is a multi-input functional encryption scheme, which can be considered in two categories:

FIGURE 4.2: IND-CCA Security Game: $\mathsf{Exp}^{\mathsf{fe}}_{\mathcal{A}}(1^{\ell})$

- **Setup Phase.** The challenger $\mathcal{C}^{\mathsf{fe}}$ generates the pair $(\mathsf{MSK}, \mathsf{MPK})$ by invoking the setup algorithm on input (1^{λ}). Then $\mathcal{C}^{\mathsf{fe}}$ sends MPK to $\mathcal{A}$.

$$\mathcal{C}^{\mathsf{fe}} \xrightarrow{\mathsf{MPK}} \mathcal{A}$$

- **Token Query Phase 1.** $\mathcal{A}$ adaptively query token and decryption are as follows:
  - Token key query:

$$\begin{aligned} &\xleftarrow{f_i} \mathcal{A}(1^{\ell}, \mathsf{pk}) \\ \mathsf{tok}_{f_i} \leftarrow \mathsf{TokGen}(\mathsf{MSK}, .) & \\ &\mathcal{C}^{\mathsf{fe}} \xrightarrow{\mathsf{tok}_{f_i}} \end{aligned}$$

  - Decryption query, $\mathcal{A}$ chooses a ciphertext ct and $f \in \mathcal{F}$, then the challenger computes $\mathsf{tok}_f \leftarrow \mathsf{TokGen}(\mathsf{MSK}, f)$, computes $\mathsf{z} = \mathsf{Dec}(\mathsf{ct}, \mathsf{tok}_f)$ and sends back z to $\mathcal{A}$.

$$\begin{aligned} &\xleftarrow{\mathsf{ct}, f_j} \mathcal{A}(1^{\ell}, \mathsf{pk}) \\ \mathsf{tok}_f \leftarrow \mathsf{TokGen}(\mathsf{MSK}, f_j) & \\ \mathsf{z} \leftarrow \mathsf{Dec}(\mathsf{tok}_{f_j}, \mathsf{ct}) & \\ &\mathcal{C}^{\mathsf{fe}} \xrightarrow{\mathsf{z}} \end{aligned}$$

- **Challenge Phase.** $\mathcal{A}$ sends to the challenger two messages $m_0, m_1 \in \mathcal{M}$ of the same length such that $f(m_0) = f(m_1)$ for all queried functions f the adversary has queried in the query phase 1.

$$\mathcal{C}^{\mathsf{fe}} \xleftarrow{(m_0, m_1)} \mathcal{A}$$

- **Challenge Respond.** $\mathcal{C}^{\mathsf{fe}}$ flips a coin to generate random bit β and send $\mathsf{ct}^* = \mathsf{Enc}(\mathsf{MPK}, m_{\beta})$.

$$\mathcal{C}^{\mathsf{fe}} \xrightarrow{\mathsf{ct}^*} \mathcal{A}$$

- **Query Phase 2.** Query Phase 2: same as Query Phase 1, except that the challenger responds to token-query for function f_i is $f_i(m_0) = f_i(m_1)$
- **Output Phase.** Adversary outputs a bit β^*.

$$\mathcal{C}^{\mathsf{fe}} \xleftarrow{\beta^*} \mathcal{A}$$

- **Winning Condition, Success and Advantage.** Adversary wins the game if $\beta = \beta^*$ and the following condition is met. It is required that if $m_0 \neq m_1$, $f_i(m_0) = f_i(m_1)$ for all the function f_i queried in both query phases 1 and 2. If the winning condition is satisfied, the output of the game is 1 or 0 otherwise. The following are the definitions for the success probability and the advantage of the adversary:

$$\mathsf{Succ}^{\mathsf{ind-cpa}}_{\mathsf{FE},\mathcal{A}}(\ell) = \Pr[\beta = \beta^*], \ \mathsf{Adv}^{\mathsf{ind-cpa}}_{\mathsf{FE},\mathcal{A}}(\ell) = |\Pr[\beta = \beta^*] - \frac{1}{2}|,$$

- **Multi Input FE** is a more general notion of FE, introduced by Goldwasser in [131]. In a MI-FE scheme, given ciphertexts:

  $$\mathsf{ct}_1 = \mathsf{Enc}(m_1), \mathsf{ct}_2 = \mathsf{Enc}(m_2), \ldots, \mathsf{ct}_k = \mathsf{Enc}(m_k)$$

  the owner of the secret token tok_f for function f, can run the decryption algorithm to evaluate

  $$\mathsf{Dec}(\mathsf{ct}_1, \ldots, \mathsf{ct}_k, \mathsf{tok}_f) \mapsto f(m_1, \ldots, m_k)$$

  MI-FE is incredibly beneficial in a wide variety of real-world applications, such as cloud-based calculations and electronic voting protocols. Unfortunately however, there are relatively few operational MI-FE schemes. For example, the MI-FE schemes proposed in [131, 22, 15, 62] are unsuitable for real-world applications since all rely on quite unrealistic assumptions, such as indistinguishability obfuscation or multi-linear maps.

  An efficient MI-FE in the symmetric setting is proposed in [24], which enables the computation of the sum of the components of an encrypted vector; besides, it can be used to embed FE into the problems of *order revealing encryption*and *differentially private databases*. Other MI-FE can be found in [60, 5]. In fact, [5] provided a MI-FE scheme for a non-trivial functionality that is secure against unbounded collusions and is based on standard cryptographic assumptions with polynomial security loss.

- **The Multi-Client FE** is defined in [169], and it can be considered as a particular case of MI-FE. In the multi-client functional encryption scheme, the encryption algorithm, instead of taking the message m as input, takes m_i, which is part of m along with index i for each user and a time-based tag:

  $$\big(\mathsf{ct}_1 = \mathsf{Enc}(1, m_1, l), \mathsf{ct}_2 = \mathsf{Enc}(2, m_2, l), \ldots, \mathsf{ct}_k = \mathsf{Enc}(k, m_k, l)\big).$$

  In fact, a single message m is broken into a vector $(m_1, \ldots m_k)$. Later the owner of the token tok_f, for function f can compute $f(m_1, \ldots, m_k)$. The security notion in MI-FE guarantees that a combination of ciphertexts generated for various labels does not generate a valid global ciphertext, and the adversary gains no knowledge as a result [3].

4.3 Sub Classes of Functional Encryption Scheme

Many encryption concepts and constructions can be viewed as special cases of Functional Encryption. We will now briefly review some practical variations of the functional encryption scheme for a particular functionality. It is worth noting that several of the following encryption schemes had been developed prior to establishing the functional encryption notion.

4.3.1 Identity-Based Encryption Scheme

The Identity-Based Encryption scheme can be regarded as the most well-studied functional encryption scheme and it was introduced by Adi Shamir in 1984 [234]. In 2005, Sahai and Waters [229] proposed a generalization of Shamir's approach, namely the *Fuzzy*

Identity-Based Encryption. In fact, proved in [58] IBE is the first known cryptosystem that cannot be realized from public key encryption.

In the IBE scheme, the plaintext is described as a pair $(\mathtt{id}, m)$ of an identity (such as email address) and the message (a.k.a., payload message), and the functionality is similar to the identity map if the ciphertext and the token contains the same identity; otherwise, it outputs $\perp$. Figure 4.3 illustrates a generic schema for an IBE scheme.

FIGURE 4.3: Identity-Based Encryption scheme

- Message Space $= (\mathcal{I} \times \mathcal{M})$,
- Keyspace $= \mathcal{I}$
- $\mathcal{F}_{\mathsf{IBE}} = \{f : (\mathcal{I} \times \mathcal{M}) \to \mathcal{M} \cup \{\perp\}\}$;
- $f(id, (\mathtt{id}, m)) = \begin{cases} m \text{ If } id = \mathtt{id} \\ \perp \text{ If } id \neq \mathtt{id} \end{cases}$
- $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{IBE.Setup}(1^{\ell})$
- $\mathsf{ct} \leftarrow \mathsf{IBE.Enc}(\mathsf{MPK}, (id, m))$
- $\mathsf{tok}_{\mathtt{id}} \leftarrow \mathsf{IBE.TokGen}(\mathsf{MSK}, \mathtt{id})$
- $\mathsf{IBE.Dec}(\mathsf{tok}_{\mathtt{id}}, \mathsf{ct}) = \begin{cases} m \text{ If } id = \mathtt{id} \\ \perp \text{ If } id \neq \mathtt{id} \end{cases}$

Additional Note. Technically, IBE serves the same function as the public key. The difference in this paradigm is that instead of encrypting the data with each user's public key, the information holder can encrypt the data with the identity, resulting in a more efficient cryptosystem. The owner of the token $\mathsf{Tok}_{\mathtt{id}}$ can later recover the message if and only if the identity inside the token matches the identity inside the ciphertext.

Although numerous schemes have been proposed in the literature, the first practical implementations were presented in 2001 by Boneh and Franklin [53], Water [257] and Cocks [81]. In [257], Water proposed a completely secure Identity-Based Encryption and Hierarchical Identity-Based Encryption (HIBE) system. Using an agile methodology called "Dual System Encryption", he demonstrated the scheme's security under the well-established decisional Bilinear Diffie-Hellman and linear decisional assumptions. We distinguish anonymous-IBE among all others, which has a function hiding property; namely, neither the token nor the ciphertext reveals the user's identity [262, 153].

4.3.2 Predicate Encryption Scheme

The predicate encryption scheme [56, 173] resembles the IBE; however, it supports more extensive functionality than IBE. Precisely, in PE, a plaintext contains the index and the payload message, (ind, m), same as IBE, while the functionality is defined in terms of a polynomial-time predicate $P : (\mathcal{K} \cup \{\varepsilon\}) \times \mathcal{I} \mapsto \{0, 1\}$.

Additional Note. Predicate encryption scheme has two flavors; public index and private index. PE does not provide any security over the index. In fact, using the empty token, $\mathsf{tok}_{\varepsilon}$, the decryption algorithm returns both the plaintext index and the payload message length.

4.3.3 Attribute-Based Encryption Scheme

Attribute-Based Encryption scheme (ABE for short) is introduced by Sahai and Water [229] as a generalization of IBE and PE in which access to encrypted data is restricted to users

FIGURE 4.4: Predicate Encryption scheme

- Index Space $= \mathscr{I}$
- Message Space $= (\mathscr{I} \times \mathcal{M})$,
- Keyspace $= \mathcal{K} \cup \{\varepsilon\}$
- $f(k, (\mathsf{ind}, m)) = \begin{cases} m \text{ If } P(k, \mathsf{ind}) = \mathsf{True} \\ \perp \text{ If } P(k, \mathsf{ind}) = \mathsf{False} \end{cases}$

- $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{PE.Setup}(1^{\ell})$
- $\mathsf{ct} \leftarrow \mathsf{PE.Enc}(\mathsf{MPK}, (\mathsf{ind}, m))$
- $\mathsf{tok}_k \leftarrow \mathsf{PE.TokGen}(\mathsf{MSK}, k \in \mathcal{K})$
- $\mathsf{PE.Dec}(\mathsf{tok}_k, \mathsf{ct}) = \begin{cases} m \text{ If } P(k, \mathsf{ind}) = \mathsf{True} \\ \perp \text{ If } P(k, \mathsf{ind}) = \mathsf{False} \end{cases}$

with specific attributes and policies. This concept was later refined by Goyal, Pandey, Sahai, and Waters [134] into two distinct formulations: *Key Policy* and *Ciphertext Policy*.

An ABE is defined over the functionality $\mathcal{F}$, which includes polynomial-size boolean formulas, a.k.a., *policy*, in terms of an n variables boolean $\vec{z} = (z_1, \ldots, z_n)$. When we link the policy to the keyspace, we call it the Key-Policy Attribute-Based. The alternative approach is the Ciphertext-Policy Attribute-Based, which encrypts the policy in the encryption algorithm and associates the attribute $\vec{z}$ with the key.

FIGURE 4.5: Attribute-Based Encryption scheme

- Message Space $= (\{0,1\}^n \times \mathcal{M})$,
- Policy Set $= \left\{\phi_n : \{0,1\}^n \to \{0,1\}\right\}$
- Attribute Set $= \{\alpha \in \{0,1\}^n\}$
- Functionality: $\mathcal{F}_{\mathsf{AB}} = \left\{f_{\phi} : \{0,1\}^n \to \{0,1\}\right\}$;
- $f_{\phi}(\alpha, m) = \begin{cases} m \text{ If } \phi(\alpha) = \mathsf{True} \\ \perp \text{ If } \phi(\alpha) = \mathsf{False} \end{cases}$

- Key Policy : $\mathsf{tok}_{\phi_n}, \mathsf{ct} = \mathsf{Enc}(\alpha, m)$ $\qquad \mathsf{PE.Dec}(\mathsf{tok}_{\phi}, \mathsf{ct}[\alpha, m]) = \begin{cases} m \text{ If } \phi(\alpha) = \mathsf{True} \\ \perp \text{ If } \phi_n(\alpha) = \mathsf{False} \end{cases}$
- Ciphertext Policy : $\mathsf{tok}_{\alpha}, \mathsf{ct} = \mathsf{Enc}(\phi_n, m)$ $\qquad \mathsf{PE.Dec}(\mathsf{tok}_{\alpha}, \mathsf{ct}[\phi_n, m]) = \begin{cases} m \text{ If } \phi_n(\alpha) = \mathsf{True} \\ \perp \text{ If } \phi(\alpha) = \mathsf{False} \end{cases}$

ABE has a number of features described in the literature, including threshold-policy access control [190, 195], key-policy access control [134, 132, 186], ciphertext-policy access control [47, 135, 189, 256, 263], non-monotonic access control [19], hierarchical access control [255], and revocable access control [155]. In [199] the authors collect a detailed survey on the Attribute-base encryption scheme; we refer to this survey for more on the subject.

4.3.4 Hidden Vector Encryption Scheme

Hidden Vector Encryption (HVE for short) was proposed in [56] By Boneh and Water as a subclass of *Searchable Encryption* systems. Prior to HVE, searchable encryption schemes were restricted to simple equality tests, whereas HVEallows for conjunctive and range searches.

FIGURE 4.6: Hidden Vector Encryption scheme

- Index Space $= \{0,1\}^*$
- Keyspace $= \{0,1,*\}^n$
- Functionality: $\mathcal{F}_{\mathsf{HV}} = \{f_{\vec{v}} : (\{0,1\}^*)^n \to \{0,1\}\}$
- $f_{\vec{v}}((w_1, \dots w_n), m) = \begin{cases} v_i = w_i \vee v_i = * : m \\ \text{otherwise: } \perp \end{cases}$

- $\mathsf{ct} \leftarrow \mathsf{HV.Enc}(\mathsf{MPK}, (\vec{w}, m))$
- $\mathsf{tok}_{\vec{v}} \leftarrow \mathsf{PE.TokGen}(\mathsf{MSK}, \vec{v})$
- $\mathsf{HV.Dec}(\mathsf{tok}_{\vec{v}}, \mathsf{ct}) = \begin{cases} m \text{ If } \phi(k, \mathsf{ind}) = \mathsf{True} \\ \perp \text{ If } \phi(\mathsf{ind}) = \mathsf{False} \end{cases}$

4.4 Inner Product Encryption Scheme

Inner product encryption IPE is a notable special case of functional encryption [56, 173, 189, 217, 212, 7, 231, 194] which has been subjected to extensive studies in the last decade. In comparison to previous schemes, which were restricted to conjunctive searches. For example Katz, Sahai, and Waters [173] proposed a system based on the Inner product (dot product) of two vectors over $\mathbb{Z}_n$ for some composite number n that allows for more complex evaluations of disjunctions, polynomials, and CNF/DNF formulae. Subsequently IPE was extended by Okamoto and Takashima [212] and Lewko *et al.* in [189] over the field $\mathbb{F}_p$. Moreover a post-quantum secure IPE scheme is proposed in [91].

Simply put, in the IPE scheme, the message associated with a pair $(m, \vec{x})$, with m being the *payload message* and vector $\vec{x} \in \mathbb{Z}_p^n$ the *attribute*, and the token is associated with a vector $\vec{v} \in \mathbb{Z}_p^n$.

The functionality is $F(\vec{v}, (m, \vec{x})) = f_{\vec{v}}(\vec{x}, m)$ which returns m if $\langle \vec{x}, \vec{v} \rangle = 0 \mod \mathbb{Z}_p$ (i.e. the two vectors are orthogonal) or $\perp$ otherwise, Figure 4.7.

In fact, IPE is a generalization of Identity-Based Encryption [234, 53, 81] and Anonymous Identity-Based Encryption [59, 6, 1].

FIGURE 4.7: Inner Product Encryption scheme

- Vector Space: $\Sigma_n = \{\vec{v} = (v_1, \dots, v_n) : v_i \in \mathbb{F}\}$
- Message Space: $(\Sigma_n \times \mathcal{M})$,
- Key Space: Σ_n
- $\mathcal{F}_{\mathsf{IP}} = \{f : \Sigma_n \times \mathcal{M} \to \mathcal{M} \cup \{\perp\}\}$;
- $f(\vec{v}, (\vec{x}, m)) = \begin{cases} m; \text{ If } \langle \vec{v}, \vec{x} \rangle = 0 \\ \perp; \text{ If } \langle \vec{v}, \vec{x} \rangle \neq 0 \end{cases}$

- $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{IP.Setup}(1^\ell, n)$
- $\mathsf{ct} \leftarrow \mathsf{IP.Enc}(\mathsf{MPK}, (\vec{x}, m))$
- $\mathsf{Tok}_{\vec{v}} \leftarrow \mathsf{IP.TokGen}(\mathsf{MSK}, \vec{v})$
- $\mathsf{IP.Dec}(\mathsf{Tok}_{\vec{v}}, \mathsf{ct}) = \begin{cases} m; \text{ If } \langle \vec{v}, \vec{x} \rangle = 0 \\ \perp; \text{ If } \langle \vec{v}, \vec{x} \rangle \neq 0 \end{cases}$

There are numerous applications for IPE, such as privacy-preserving statistical analysis, where statistical analysis includes sensitive information and conjunctive/disjunctive normal-form formulas. More concretely, we can mention anonymous Identity-Based encryption, Hidden-Vector encryption, predicate encryption schemes, computation of weighted averages and sums for statistical analysis (including sensitive information) on encrypted data, and support for polynomial evaluation over encrypted data as early applications [173], furthermore, recent applications include the development of bounded collusion FE for all circuits [11], the development of trace and revoke systems [13], and the construction of non-zero inner product encryption schemes [171].

4.4.1 IPE; Variants

There are three types of functional encryption schemes referred to as Inner Product encryption schemes; however, they differ in technical features. In all three, the keyspace and the message space are associated with vector $\vec{v}$ and vector $\vec{x}$, respectively:

1. **Predicate only IPE:** In this configuration, the message space contains only the vector $\vec{x}$ and no payload message m. The decryption algorithm outputs a boolean value depending on the inner product of $\vec{v}$ and $\vec{x}$:

$$\mathcal{F}_{\mathsf{predicate-only-IP}} = \left\{f_{\vec{v}} : \Sigma_n \to \{0,1\} : \vec{v} \in \Sigma_n\right\},$$

where

$$f_{\vec{v}}(\vec{x}) = \begin{cases} \mathsf{True} \ \text{ If } \langle \vec{x}, \vec{v} \rangle = 0 \\ \mathsf{False} \ \text{ If } \langle \vec{x}, \vec{v} \rangle \neq 0 \end{cases}$$

2. **Predicate IPE:** The message space includes vector $\vec{x}$ and a payload message $m \in \mathcal{M}$. As a result, the decryption algorithm returns m if two vectors $\vec{v}$ and $\vec{x}$ are orthogonal, or an error symbol if their inner product is not zero.

$$\mathcal{F}_{\mathsf{predicate-IP}} = \left\{f_{\vec{v}} : \Sigma_n \times \mathcal{M} \to \mathcal{M} \cup \{\perp\} : \vec{v} \in \Sigma_n\right\}$$

where

$$f_{\vec{v}}(m, \vec{x}) = \begin{cases} m \ \text{ If } \langle \vec{x}, \vec{v} \rangle = 0 \\ \perp \ \text{ If } \langle \vec{x}, \vec{v} \rangle \neq 0 \end{cases}$$

3. **IPE:** Unlike the first two schemes, IPE does not play a role as a predicate, and the functionality is the inner product of two vectors $\vec{x}$ and $\vec{v}$. It means that the output of the decryption algorithm would be $\langle \vec{x}, \vec{v} \rangle$.

$$\mathcal{F}_{\mathsf{IP}} = \left\{f_{\vec{v}} : \Sigma_n \to \mathbb{Z} : \vec{v} \in \Sigma_n\right\},$$

where

$$f_{\vec{v}}(\vec{x}) = \langle \vec{x}, \vec{v} \rangle$$

Multi-Input IPE. A more general notion of the IPE is the multi-input IPE [2, 3, 4] which, instead of a single vector, they consider m vectors and define the functionality as follows:

$$\mathcal{F}_{\mathsf{IP}} = \left\{f_{\vec{v}_1,\ldots,\vec{v}_m} : \Sigma_n^m \to \mathbb{Z} : \vec{v}_i \in \Sigma_n\right\},$$

where

$$f_{(\vec{v}_1,\ldots,\vec{v}_m)}(\vec{x}_1,\ldots,\vec{x}_m) = \sum_{i=0}^{m} \langle \vec{x}_i, \vec{v}_i \rangle$$

Additional Note. In our research, we study the Predicate IPE. The result is also applicable for the predicate-only-IPE; however, we abuse the notation and refer to Predicate IPE as IPE.

4.5 Technical Overview

Before describing our IPE scheme, we provide a detailed justification for the the modifications required to attain perfect correctness property in IPE. But first, we state that the ultimate purpose of our research was to develop a verifiable Inner product encryption scheme. To this end, we were looking for an IPE scheme that was compatible with Badrinarayanan's transformation from the start.

To instantiate the transformation of Badrinarayanan *et al.* we need to build an IPE scheme with perfect correctness. Our starting point to construct a perfectly correct IPE scheme is the IPE scheme of Park [217] which only enjoys statistical correctness.

The reason for choosing this IPE is that it is conceptually simple, and its security is based on standard assumptions over bilinear groups. However, to make the Park's scheme compatible with the Badrinarayanan *et al.*'s transformation, we need to solve several technical challenges, in particular:

i. The master public key needs to be verifiable.

ii. The scheme has to satisfy perfect correctness.

This requires substantial modification of all main algorithms: setup, token generation, encryption, and decryption.

4.5.1 Verification Algorithms

A VIPE scheme requires public verification algorithms that can verify the outputs of the setup, encryption, and token generation algorithms, in particular check whether these algorithms were run honestly. In more detail, if any string (master public key, ciphertext or token) passes the corresponding verification algorithm, it means it was a proper output of the corresponding algorithm (setup, encryption, or token generation). Each party who runs the setup, encryption or token generation algorithm needs to provide a proof that it executed the algorithm honestly without revealing harmful information about the secret parameters or the randomness used in the algorithm.

Usually, non-interactive Zero-Knowledge (NIZK) proofs are used in this context. Unfortunately, NIZK proofs cannot be used for verifiable FE as they rely on a trusted CRS (Common Reference String) or random oracles, and we aim at *perfect verifiability* which has to hold despite any collusion and computing power. The transform of Badrinarayanan *et al.* solves the issue cleverly employing NIWI-proofs.

Following the transform of [23], our VIPE consists of four instances of an IPE scheme. In the VIPE's encryption algorithm, we first run the IPE's encryption algorithm four times to generate four ciphertexts and then we prove that all these four ciphertexts are the encryption of the same message or that some other trapdoor predicate is satisfied (the latter is needed for message indistinguishability and will be detailed later).

For the sake of argument, let us assume the VIPE scheme consists only of two (instead of four) parallel perfectly correct IPE scheme instantiations IP and $\hat{\mathsf{IP}}$.

The master public key of the Park's scheme [217] contains a component $\Lambda = \mathbf{e}(g, g')$ in which g is public, but g' needs to be kept secret. An honestly computed ciphertext CT in IP includes $\mathsf{ct}_1 := g^{-s}$ and $\mathsf{ct}_7 := \Lambda^{-s} \cdot m$, among its components (we ignore the other components). We first prove that CT (resp. $\hat{\mathsf{CT}}$ in $\hat{\mathsf{IP}}$) is well-formed. Then we need to prove that the two ciphertexts are both encryptions of the same message M (i.e., $m = \hat{m} = M$). We reduce the problem to proving that the following property holds:

$$\frac{\mathsf{ct}_7}{\hat{\mathsf{ct}}_7} = \frac{\mathbf{e}(g, g')^{-s} \cdot m}{\mathbf{e}(\hat{g}, \hat{g}')^{-\hat{s}} \cdot \hat{m}} = \frac{\mathbf{e}(\hat{\mathsf{ct}}_1, \hat{g}')}{\mathbf{e}(\mathsf{ct}_1, g')} = \frac{\mathbf{e}(\hat{g}^{\hat{s}}, \hat{g}')}{\mathbf{e}(g^s, g')}.$$

However, since g' and $\hat{g}'$ are not public, the party who runs the encryption algorithm would be unable to prove this property.

We solve this issue in the following way. We add to the master public key of IP two elements g_1, g_2 (and $\hat{g}_1, \hat{g}_2$ for $\hat{\mathsf{IP}}$), satisfying

$$\Lambda = \mathbf{e}(g, g') = \mathbf{e}(g_1, g_2),$$
$$\hat{\Lambda} = \mathbf{e}(\hat{g}, \hat{g}') = \mathbf{e}(\hat{g}_1, \hat{g}_2).$$

Then, we add the following equations for the new secret variables $\mathcal{X}_3 := g_1^s, \hat{\mathcal{X}}_3 := \hat{g}_1^{\hat{s}}$:

$$\mathsf{ct}_7^{-1} \cdot \hat{\mathsf{ct}}_7 = \mathbf{e}(\mathcal{X}_3, g_2) \cdot \mathbf{e}(\hat{\mathcal{X}}_3, \hat{g}_2)^{-1},$$
$$\mathbf{e}(g, \mathcal{X}_3) = \mathbf{e}(\mathsf{ct}_1, g_1),$$
$$\mathbf{e}(\hat{g}, \hat{\mathcal{X}}_3) = \mathbf{e}(\hat{\mathsf{ct}}_1, \hat{g}_1).$$

It is easy to see that these equations are satisfied if and only if $m = \hat{m}$, which the encryptor can prover.

Having modified Park's scheme, we thus have to prove that the modified scheme is IND-secure. This is done in Section 4.7, in which we reduce the IND-Security of the scheme to the Decision Linear assumption.

4.5.2 Achieving Perfect Correctness

The underlying IPE scheme must have perfect correctness for Badrinarayanan *et al.*'s transform to work. Unfortunately, to our knowledge, all IPE schemes[2] known in the literature have a negligible probability of error which makes cheating possible and so not directly usable to construct verifiable functional encryption and functional commitments for the IPE functionality.

Suppose the IPE scheme had a negligible probability of decryption error rather than perfect correctness. In that case, dishonest parties might collude with each other so that the verification algorithms would accept invalid results. Contrast this with the functional commitments. In the functional commitment, the committer is the same party who generates the ciphertext (the commitment) and the token (the decommitment) and thus might profit from a negligible space of decryption error to prove false assertions on its committed value.

[2]Recall that we refer to the IPE functionality of Katz, Sahai and Waters [173].

In more detail, in most pairing-based IPE schemes the encryption and decryption algorithms work as follows:

$$\begin{aligned}&\mathsf{Enc}(\mathsf{MPK}, \overrightarrow{x}, m) \to \mathsf{CT},\\ &\mathsf{Dec}(\mathsf{Tok}_{\overrightarrow{v}}, \mathsf{CT}) \to m_* = m \cdot (\mathsf{random})^{\langle \vec{x}, \vec{v}\rangle}.\end{aligned}$$

in which random is some random value that depends on the randomness used by the token generator and encryption algorithms. Thus, even in the case of honest parties, there is a negligible probability that $\mathsf{random} = 1$ and so, even if $\langle \vec{x}, \vec{v}\rangle \neq 0$, the decryption algorithm may output a valid message m instead of $\perp$.

In the case of dishonest parties, two parties (the encryptor and the token generator) may collude with each other to create randomness such that random equals 1.

In this case, the parties would prove that they followed the protocol correctly, and invalid results would pass the verification algorithms. A similar problem also appears in the context of MPC in the head [161], where the soundness of the ZK protocol built from MPC, strongly relies on the perfect correctness of the underlying MPC. To cope with statistical correctness in MPC in the head, a coin tossing protocol can be employed, while in a completely non-interactive scenario like ours this is more challenging. Hence, to obtain a VIPE scheme it is crucial to construct an IPE scheme satisfying perfect correctness.

Recall that the decryption algorithm in the IPE scheme of Park [217] works as follows:

$$\mathsf{Dec}\big(\mathsf{Tok}_{\overrightarrow{v}}, \mathsf{CT} = \mathsf{Enc}(\overrightarrow{x}, m)\big) \mapsto m^* = m \cdot \mathbf{e}(g, h)^{(\lambda_1 s_3 + \lambda_2 s_4)\langle \vec{x}, \vec{v}\rangle}$$

in which λ_1, λ_2 are random values used in the token generation algorithm and s_3, s_4 are random values used in the encryption algorithm. To decide whether to accept the decryption's output or not, the first attempt would be the following. Generate two ciphertexts $\mathsf{ct}, \mathsf{ct}'$ with two independent random values $\{s_i\}, \{s_i'\}$, decrypt both ct and ct′ to get m_1^* and m_2^* and if $m_1^* = m_2^*$ accept the result, or output $\perp$ otherwise. In more detail:

$$\begin{aligned}m_1^* &= m \cdot \mathbf{e}(h, g)^{(\lambda_1 s_3 + s_4 \lambda_2)\langle \vec{x}, \vec{v}\rangle},\\ m_2^* &= m \cdot \mathbf{e}(h, g)^{(\lambda_1 s_3' + s_4' \lambda_2)\langle \vec{x}, \vec{v}\rangle}\end{aligned} \tag{4.1}$$

However, in case $\langle \vec{x}, \vec{v}\rangle \neq 0$ there is non-zero probability for which:

$$\lambda_1 s_3 + s_4 \lambda_2 = \lambda_1 s_3' + \lambda_2 s_4' \neq 0 \Rightarrow m_1^* = m_2^* \neq m$$

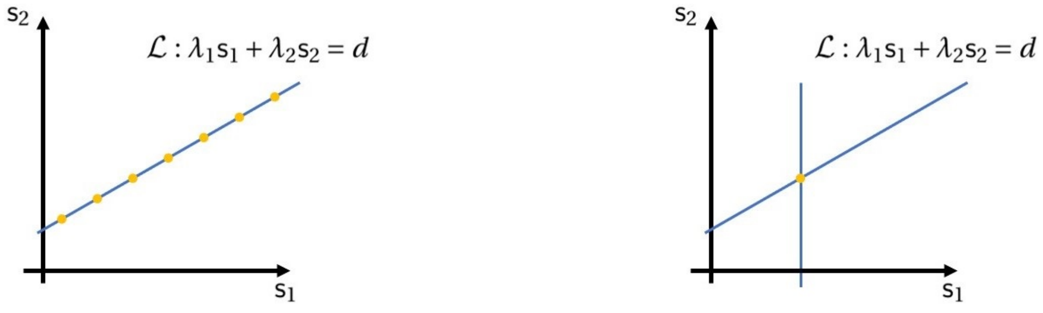
As seen in Figure 4.8 (the left diagram), if we interpret the $\mathcal{L} : \lambda_1 s_3 + s_4 \lambda_2$ as a line, each pair of green dots on this line would result in $M = m_2^*$.

To avoid this issue, we choose the random values in such a way that the above equality can never occur. To do so, in the encryption algorithm we choose non-zero random values $s_1, \ldots, s_4$ and $s_1', \ldots, s_4'$ such that $s_3 \neq s_3'$, and $s_4 = s_4'$. In this case, we have:

$$\begin{aligned}&\lambda_1 s_3 + s_4 \lambda_2 = \lambda_1 s_3' + \lambda_2 s_4 \Rightarrow\\ &\qquad \lambda_1 (s_3 - s_3') = 0 \Rightarrow (\lambda_1 = 0) \vee (s_3 = s_3')\end{aligned} \tag{4.2}$$

The right diagram in Figure 4.8 shows that in fact the intersection of line $\mathcal{L}$ with line $x = s_3$ has a unique point which is the only point the $M = m_2^*$ would occur.

FIGURE 4.8: The left diagram shows the points that result to incorrect output in decryption algorithm and the right diagram shows the single point that result to correct output in decryption algorithm.


s_2
$\mathcal{L} : \lambda_1 s_1 + \lambda_2 s_2 = d$
s_1
s_2
$\mathcal{L} : \lambda_1 s_1 + \lambda_2 s_2 = d$
s_1

Based on the way λ_1, s_3, s_3' have been chosen, neither $(\lambda_1 = 0)$ nor $(s_3 = s_3')$ may happen; hence the decryption algorithm outputs m if and only if $\langle \vec{x}, \vec{v} \rangle = 0$. The resulting IPE scheme satisfies perfect correctness as wished, and we prove that it is still selectively indistinguishability-secure under the DLin Assumption.

When constructing a VIPE scheme from such an IPE scheme, these additional constraints in the encryption and token generation procedures will correspond to more constraints in the proofs of correct encryption and token generation.

Furthermore, an additional challenge we will have to address is that some of the proofs in the Badrinarayanan*et al.* transform are for relations that consist of a generalized form of disjunction. Thus standard techniques to implement disjunctions for GS proofs cannot be directly applied, see Section 3.12.

4.5.3 IPE; Formal Definition

For security parameter $\ell \in \mathbb{N}$ and the filed $\mathbb{F}$, we present a set of vectors of length n defined over $\mathbb{F}$ by

$$\Sigma_n = \left\{ \vec{v} = (v_1, \dots, v_n) : v_i \in \mathbb{F} \right\}.$$

Considering a message space $\mathcal{M}$, we define the functionality $\mathcal{F}_{\mathsf{IP}}$ as follows:

$$\mathcal{F}_{\mathsf{PE}} = \left\{ f_{\vec{v}} : \Sigma_n \times \mathcal{M} \to \mathcal{M} \cup \{\perp\} : \vec{v} \in \Sigma_n \right\},$$

where

$$f_{\vec{v}}(\vec{x}, m) = \begin{cases} m & \text{If } \langle \vec{x}, \vec{v} \rangle = 0 \\ \perp & \text{If } \langle \vec{x}, \vec{v} \rangle \neq 0 \end{cases}.$$

Both $\mathcal{M}$ and the field size can depend on the security parameter (not necessarily polynomial of it) but for simplicity, we will skip this detail. IPE can be seen as a functional encryption scheme for the functionality of $\mathcal{F}_{\mathsf{IP}}$. More concretely, an IPE scheme is defined as follows.

Definition 33 (**Inner Product Encryption Scheme**). *An IPE scheme for a message space $\mathcal{M}$ and a family of vectors $\Sigma = \{\Sigma_n\}_{n \in \mathbb{N}}$ over $\mathbb{F}$ is a tuple of four polynomial-time algorithms:*

$$\Pi^{\mathsf{IP}} = \langle \mathsf{IP.SetUp}, \mathsf{IP.TokGen}, \mathsf{IP.Enc}, \mathsf{IP.Dec} \rangle$$

with the following syntax and satisfying the correctness property below.

- ***Set Up*** *is a probabilistic algorithm that takes the security parameter, ℓ, and integer $n \in \mathbb{N}$ as inputs and returns a pair of keys* $(\mathsf{MPK}, \mathsf{MSK})$. *We refer to the first component,* MPK*, as a master public key, which defines a message space $\mathcal{M}$, and the second one,* MSK*, as the master secret key. It requires that both keys have a length polynomial in terms of the security parameter:*

$$(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{IP.SetUp}(1^\lambda, n)$$

We implicitly consider the public key as the input to all algorithms even if it is not stated explicitly.

- ***Token Generation*** *is a probabilistic algorithm that on input master secret key and vector $\vec{v} \in \Sigma_n$ generates the token for $f_{\vec{v}}$:*

$$\mathsf{Tok}_{\vec{v}} \leftarrow \mathsf{IP.TokGen}(\mathsf{MSK}, \vec{v})$$

Notice that here $f_{\vec{v}}$ and $\vec{v}$ are considered identical to avoid heavy notation.

- ***Encryption*** *is a probabilistic algorithm that takes as inputs the master public key, a message $m \in \mathcal{M}$ and vector $\vec{x} \in \Sigma_n$ to generate a ciphertext:*

$$\mathsf{ct} \leftarrow \mathsf{IP.Enc}(\mathsf{MPK}, \vec{x}, m)$$

- ***Decryption*** *is a deterministic algorithm that on inputs token and a ciphertext outputs $m' \in \mathcal{M} \cup \{\perp\}$.*

$$\mathsf{IP.Dec}(\mathsf{MPK}, \mathsf{Tok}_{\vec{v}}, \mathsf{ct}) \mapsto m' \in \mathcal{M} \cup \{\perp\}.$$

- ***Perfect correctness:*** Π^{IP} *is perfectly correct if for all $\lambda, n \in \mathbb{N}$, all $\vec{x}, \vec{v} \in \Sigma_n$ and all $m \in \mathcal{M}$ the following holds:*

$$\Pr\left[\mathsf{IP.Dec}(\mathsf{MPK}, \mathsf{Tok}_{\vec{v}}, \mathsf{ct}) = f_{\vec{v}}(\vec{x}, m) \mid \begin{array}{l} (\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{IP.SetUp}(1^\ell, n), \\ \mathsf{Tok}_{\vec{v}} \leftarrow \mathsf{IP.TokGen}(\mathsf{MPK}, \mathsf{MSK}, \vec{v}), \\ \mathsf{ct} \leftarrow \mathsf{IP.Enc}(\mathsf{MPK}, \vec{x}, m) \end{array}\right] = 1$$

Additional Note. Contrary to the other two types, correctness in a predicate-IPE product encryption scheme is not easily achievable.

In predicate only IPE, there is no payload message m, so the output of the decryption algorithm would be equal to

$$1 = \mathsf{random}^{\langle \vec{x}, \vec{v} \rangle} = \mathsf{random}^0$$

if two vectors are orthogonal and would be some random number if they are not. Further, in the case of IPE, the correctness property guarantee that the output of the decryption algorithm is indeed the inner product of the two vectors. In IPE (the third one) 4.4.1, the correctness property ensures that the output of the decryption algorithm, indeed is equal to the inner product of the two vectors. The other two, is not actually the same. In the big picture, in most IPE schemes, the encryption and decryption algorithms work as follows:

$$\mathsf{Enc}(\mathsf{MPK}, \overrightarrow{x}, m) \to \mathsf{ct}, \;\; \mathsf{Dec}(\mathsf{Tok}_{\overrightarrow{v}}, \mathsf{ct}) \to m^* = m \odot (\mathsf{random})^{\langle \vec{x}, \vec{v} \rangle},$$

in which random is some random value in from the underlying group $\langle \mathbb{G}, \odot \rangle$ that depends on the randomness used by the token generator and encryption algorithms.

Thus, even in case of honest parties, there is a negligible probability that $\mathtt{r} = 1$ and so, even if $\langle \vec{x}, \vec{v} \rangle \neq 0$, the decryption algorithm may output a valid message m instead of $\perp$.

4.5.4 Security Notion for IPE

Extensive research has been done into developing the Inner Product Encryption scheme, which has resulted in a variety of IPE in terms of security. For example, Abdalla *et al.* [7], present an IPE with standard assumptions that has selectively security and later Agrawal *et al.* [12, 11] present an IPE scheme that achieves adaptive SIM-based security. Moreover, in [73] Castagnos *et al.*presents an adaptive IND-secure IPE schemes, which allow for the evaluation of unbounded inner products modulo a prime p. Finally, the first IND-CCA IPE schemes based on the DDH, DCR, and any of the MDDH assumptions is presented by Benhamouda et al in [41].

To model the security of the IPE, we adopt the indistinguishability-based (IND) notion of security [56], in particular selective security [57]. However, Boneh, Sahai, and Waters [55] showed deficiencies of this notion *in general* and impossibility results for the more general notion of simulation-based security; see also [71, 55, 159, 90] for general techniques to overcome the known impossibility results in different settings. Nonetheless, no practical attacks are known for natural schemes to our knowledge. Selective security is sufficient for CCA-security [57] and our application of verifiable polynomial commitments of Section 5.2.

The selectively indistinguishability-based notion of security for an IPE scheme over the vector space Σ and message space $\mathcal{M}$ are formalized using the game $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ind-ip}}(1^{\ell}, n)$ in Fig. 4.9, between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}^{\mathsf{ipe}}$ (defined in the game) parameterized by security parameter ℓ and dimension n. The advantage of $\mathcal{A}$ in this game is:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IP}}(\lambda, n) = \left| \Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ind-ip}}(1^{\ell}, n) = 1 \right] - \frac{1}{2} \right|.$$

Definition 34. *The inner product encryption scheme is selectively indistinguishable secure* (IND*-secure*) *if for all $n > 0$, the advantage* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{IP}}(\lambda, n)$ *for all PPTadversaries $\mathcal{A}$ is a negligible function in terms of ℓ.*

4.6 Perfectly Correct IPE

This section presents our perfectly correct IPE, the key ingredient for building verifiable inner-product encryption (see chapter 5).

Let $(p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathsf{GroupGen}(1^{\ell})$ be a bilinear group generator, and $n \in \mathbb{N}$ be the vector length. We construct a perfectly correct IPE scheme

$$\mathsf{IP} = \langle \mathsf{IP.SetUp}, \mathsf{IP.Enc}, \mathsf{IP.TokGen}, \mathsf{IP.Dec} \rangle$$

for the set $\mathbb{Z}_p^n$ of vectors of length n over $\mathbb{Z}_p$ and for message space $\mathcal{M} = \mathbb{G}_T$.

IPE Construction

- **Selective Challenge Phase.** $\mathcal{A}(1^\ell, n) \longrightarrow \vec{x}_0, \vec{x}_1 \in \Sigma_n$. Then $\mathcal{A}$ sends these two vectors to the challenger.
- **Setup Phase.** The challenger $\mathcal{C}^{\mathsf{ip}}$ generates the pair $(\mathsf{MSK}, \mathsf{MPK})$ by invoking the setup algorithm on input $(1^\ell, n)$. Then $\mathcal{C}^{\mathsf{ip}}$ sends MPK to $\mathcal{A}$.
- **Query Phase 1.** $\mathcal{A}$ asks for the token for a vector $\vec{v}_i \in \Sigma_n$.
- **Challenge Phase.** $\mathcal{A}$ sends the challenger two messages $m_0, m_1 \in \mathcal{M}$ of the same length.
- **Challenge Phase.** $\mathcal{C}^{\mathsf{ip}}$ flips a coin to generate random bit β and send
$$\mathsf{ct} = \mathsf{Enc}(\mathsf{MPK}, \vec{x_\beta}, m_\beta)$$
to the adversary.
- **Query Phase 2.** Query Phase 2: same as Query Phase 1.
- **Output Phase.** $\mathcal{A}$ outputs a bit β'.
- **Winning Condition.** $\mathcal{A}$ wins the game if $\beta' = \beta$ and if
$$m_0 \neq m_1\ ,\ \langle \vec{x}_0, \vec{v}_i \rangle, \langle \vec{x}_1, \vec{v}_i \rangle \neq 0,$$
for all the vectors $\vec{v}_i$ queried in both query phase 1 and 2, or
$$m_0 = m_1 : \langle \vec{v}_i, \vec{x}_0 \rangle = 0 \iff \langle \vec{v}_i, \vec{x}_1 \rangle = 0.$$
If the winning condition is satisfied, the game's output is 1 or 0 otherwise.

FIGURE 4.9: IND-CPA Security Game: $\mathsf{Exp}^{\mathsf{ip,n}}_{\mathcal{A}}(1^\ell)$

- **Set Up Algorithm:** $\mathsf{IP.SetUp}(1^\ell, n) \longrightarrow (\mathsf{MSK}, \mathsf{MPK})$

  For security parameter ℓ, $i \in [n]$ and $b \in [2]$, compute what follows:

  1. Run $\mathsf{GroupGen}(1^\ell)$ (cf. Section 2.6) to generate a tuple $\mathsf{pp} = \langle p, \mathbb{G}, \mathbb{G}_T, e \rangle$ as public parameter.
  2. Pick $g, g' \xleftarrow{\$} \mathbb{G}$.
  3. Pick $\delta_1, \theta_1, \delta_2, \theta_2, w_{1,i}, t_{1,i}, f_{b,i}, h_{b,i}, k \xleftarrow{\$} \mathbb{Z}_p^*$.
  4. Pick $\Omega \xleftarrow{\$} \mathbb{Z}_p$ and compute $\{w_{2,i}, t_{2,i}\}_{i \in [n]}$ such that:
  $$\Omega = \delta_1 w_{2,i} - \delta_2 w_{1,i} = \theta_1 t_{2,i} - \theta_2 t_{1,i}.$$
  5. For $i \in [n], b \in [2]$ set:
  $$\begin{array}{lllll} W_{b,i} = g^{w_{b,i}}, & F_{b,i} = g^{f_{b,i}}, & K_1 = g^k, & U_b = g^{\delta_b}, & h = g^{\Omega} \quad , \\ T_{b,i} = g^{t_{b,i}}, & H_{b,i} = g^{h_{b,i}}, & K_2 = g'^{\frac{1}{k}}, & V_b = g^{\theta_b}, & \Lambda = \mathbf{e}(g, g'). \end{array}$$

6. Set:

$$\begin{aligned}\mathsf{MPK} =& \big(g, h, \{W_{b,i}, F_{b,i}, T_{b,i}, H_{b,i}, U_b, V_b\}_{b\in[2], i\in[n]}, K_1, K_2, \Lambda\big)\\ &\in \mathbb{G}^{8n+8} \times \mathbb{G}_T,\\ \mathsf{MSK} =& \big(\{w_{b,i}, f_{b,i}, t_{b,i}, h_{b,i}, \delta_b, \theta_b\}_{b\in[2], i\in[n]}, g'\big)\\ &\in \mathbb{Z}_p^{8n+4} \times \mathbb{G}.\end{aligned}$$

7. Return $(\mathsf{MPK}, \mathsf{MSK})$.

- **Encryption Algorithm:** $\mathsf{IP.Enc}(\mathsf{MPK}, \vec{x}, m) \longrightarrow \mathsf{CT}$

1. For $\vec{x} = (x_1, \dots, x_n) \in \mathbb{Z}_p^n$ and a message $m \in \mathbb{G}_T$, pick random elements:

$$s_1, \dots s_4, s'_1, \dots, s'_3 \xleftarrow{\$} \mathbb{Z}_p^* : s_3 \neq s'_3$$

and compute what follows:

$$\begin{aligned}&\mathsf{ct}_1 = g^{s_2},\ \mathsf{ct}_2 = h^{s_1},\\ &\begin{Bmatrix}\mathsf{ct}_{3,i} = W_{1,i}^{s_1} \cdot F_{1,i}^{s_2} \cdot U_1^{x_i s_3} & , & \mathsf{ct}_{4,i} = W_{2,i}^{s_1} \cdot F_{2,i}^{s_2} \cdot U_2^{x_i s_3}\\ \mathsf{ct}_{5,i} = T_{1,i}^{s_1} \cdot H_{1,i}^{s_2} \cdot V_1^{x_i s_4} & , & \mathsf{ct}_{6,i} = T_{2,i}^{s_1} \cdot H_{2,i}^{s_2} \cdot V_2^{x_i s_4}\end{Bmatrix}_{i\in[n]},\\ &\mathsf{ct}_7 = \mathbf{e}(g^{s_3}, g^{s_4}),\ \mathsf{ct}_8 = \Lambda^{-s_2} \cdot m.\\ &\mathsf{ct}'_1 = g^{s'_2},\ \mathsf{ct}'_2 = h^{s'_1},\\ &\begin{Bmatrix}\mathsf{ct}'_{3,i} = W_{1,i}^{s'_1} \cdot F_{1,i}^{s'_2} \cdot U_1^{x_i s'_3} & , & \mathsf{ct}'_{4,i} = W_{2,i}^{s'_1} \cdot F_{2,i}^{s'_2} \cdot U_2^{x_i s'_3}\\ \mathsf{ct}'_{5,i} = T_{1,i}^{s'_1} \cdot H_{1,i}^{s'_2} \cdot V_1^{x_i s_4} & , & \mathsf{ct}'_{6,i} = T_{2,i}^{s'_1} \cdot H_{2,i}^{s'_2} \cdot V_2^{x_i s_4}\end{Bmatrix}_{i\in[n]},\\ &\mathsf{ct}'_7 = \mathbf{e}(g^{s'_3}, g^{s_4}),\ \mathsf{ct}'_8 = \Lambda^{-s'_2} \cdot m.\end{aligned}$$

2. Set:

$$\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \begin{Bmatrix}\mathsf{ct}_{3,i} & , & \mathsf{ct}_{4,i}\\ \mathsf{ct}_{5,i} & , & \mathsf{ct}_{6,i}\end{Bmatrix}, \mathsf{ct}_7, \mathsf{ct}_8),$$

$$\mathsf{ct}' = (\mathsf{ct}'_1, \mathsf{ct}'_2, \begin{Bmatrix}\mathsf{ct}'_{3,i} & , & \mathsf{ct}'_{4,i}\\ \mathsf{ct}'_{5,i} & , & \mathsf{ct}'_{6,i}\end{Bmatrix}, \mathsf{ct}'_7, \mathsf{ct}'_8).$$

3. Output $\mathsf{CT} = (\mathsf{ct}, \mathsf{ct}')$.

- **Token generation Algorithm:** $\mathsf{IP.TokGen}(\mathsf{MSK}, \vec{v}) \longrightarrow \mathsf{Tok}_{\vec{v}}$:

1. Pick $\ell_1, \ell_2 \xleftarrow{\$} \mathbb{Z}_p^*$.

2. For $i \in [n]$ pick $\{r_i\}, \{\Phi_i\} \xleftarrow{\$} \mathbb{Z}_p^*$.

3. Set $\mathsf{Tok}_{\vec{v}} = (K_A, K_B, \begin{Bmatrix} K_{3,i} & , & K_{4,i} \\ K_{5,i} & , & K_{6,i} \end{Bmatrix}_{i\in[n]})$ as follows and return $\mathsf{Tok}_{\vec{v}}$.

$$
\begin{aligned}
K_A &= g' \cdot \prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} \cdot K_{4,i}^{-f_{2,i}} \cdot K_{5,i}^{-h_{1,i}} \cdot K_{6,i}^{-h_{2,i}}, \\
K_B &= \prod_{i=1}^{n} g^{-(r_i+\Phi_i)}, \\
K_{3,i} &= g^{-\delta_2 r_i} \cdot g^{\lambda_1 v_i w_{2,i}}, \\
K_{4,i} &= g^{\delta_1 r_i} \cdot g^{-\lambda_1 v_i w_{1,i}}, \\
K_{5,i} &= g^{-\theta_2 \Phi_i} \cdot g^{\lambda_2 v_i t_{2,i}}, \\
K_{6,i} &= g^{\theta 1 \Phi_i} \cdot g^{-\lambda_2 v_i t_{1,i}}.
\end{aligned}
$$

- **Decryption Algorithm:** $\mathsf{IP.Dec}(\mathsf{CT}, \mathsf{Tok}_{\vec{v}})$, Let $\mathsf{CT} = (\mathsf{ct}, \mathsf{ct}')$ such that:

$$
\begin{aligned}
\mathsf{ct} &= (\mathsf{ct}_1, \mathsf{ct}_2, \{\mathsf{ct}_{3,i}, \mathsf{ct}_{4,i}, \mathsf{ct}_{5,i}, \mathsf{ct}_{6,i}\}, \mathsf{ct}_7, \mathsf{ct}_8), \\
\mathsf{ct}' &= (\mathsf{ct}'_1, \mathsf{ct}'_2, \{\mathsf{ct}'_{3,i}, \mathsf{ct}'_{4,i}, \mathsf{ct}'_{5,i}, \mathsf{ct}'_{6,i}\}, \mathsf{ct}_7, \mathsf{ct}_8).
\end{aligned}
$$

1. If $\mathsf{ct}_7 = \mathsf{ct}'_7$ output $\perp$ and stop, otherwise go to the next step.
2. Compute:

$$
\begin{aligned}
\Upsilon =& \mathsf{ct}_8 \cdot \mathbf{e}(\mathsf{ct}_1, K_A) \cdot \mathbf{e}(\mathsf{ct}_2, K_B) \cdot \\
& \prod_{i=1}^{n} \mathbf{e}(\mathsf{ct}_{3,i}, K_{3,i}) \cdot \mathbf{e}(\mathsf{ct}_{4,i}, K_{4,i}) \cdot \mathbf{e}(\mathsf{ct}_{5,i}, K_{5,i}) \cdot \mathbf{e}(\mathsf{ct}_{6,i}, K_{6,i}). \\
\Upsilon' =& \mathsf{ct}'_8 \cdot \mathbf{e}(\mathsf{ct}'_1, K_A) \cdot \mathbf{e}(\mathsf{ct}'_2, K_B) \cdot \\
& \prod_{i=1}^{n} \mathbf{e}(\mathsf{ct}'_{3,i}, K_{3,i}) \cdot \mathbf{e}(\mathsf{ct}'_{4,i}, K_{4,i}) \cdot \mathbf{e}(\mathsf{ct}'_{5,i}, K_{5,i}) \cdot \mathbf{e}(\mathsf{ct}'_{6,i}, K_{6,i}).
\end{aligned}
$$

3. If $\Upsilon = \Upsilon'$ output Υ otherwise output $\perp$.

4.6.1 Perfect Correctness Property

We now show the inner product scheme, is perfectly correct. It means that an honestly generated ciphertext decrypts correctly with probability 1.

Theorem 1. *The inner product scheme Π^{IP}, defined in 4.6 is perfectly correct.*

Proof. Since $F_{1,i}^{-s_2} \cdot \mathsf{ct}_{3,i} = W_{1,i}^{s_1} \cdot U_1^{s_3 x_i}$, we get

$$
\begin{aligned}
\mathbf{e}(F_{1,i}^{-s_2} \cdot \mathsf{ct}_{3,i}, K_{3,i}) &= \mathbf{e}(g,g)^{s_1\lambda_1 v_i w_{1,i} w_{2,i} - s_3 x_i \delta_1 \delta_2} \cdot \mathbf{e}(g,g)^{-s_1 r_i \delta_2 w_{1,i} + s_3 \lambda_1 v_i \delta_1 w_{2,i}} \\
\mathbf{e}(F_{2,i}^{-s_2} \cdot \mathsf{ct}_{4,i}, K_{4,i}) &= \mathbf{e}(g,g)^{-s_1\lambda_1 v_i w_{1,i} w_{2,i} + s_3 x_i \delta_1 \delta_2} \cdot \mathbf{e}(g,g)^{s_1 r_i \delta_1 w_{2,i} - s_3 \lambda_1 v_i \delta_2 w_{1,i}}
\end{aligned}
$$

We then obtain:

$$
\begin{aligned}
\mathbf{e}(F_{1,i}^{-s_2} \cdot \mathsf{ct}_{3,i}, K_{3,i}) \cdot \mathbf{e}(F_{2,i}^{-s_2} \cdot \mathsf{ct}_{4,i}, K_{4,i}) &= \Big(\mathbf{e}(g^{s_1}, g^{r_i}) \cdot \mathbf{e}(g^{x_i s_3}, g^{\lambda_1 v_i})\Big)^{\delta_1 w_{2,i} - \delta_2 w_{1,i}} \\
&= \mathbf{e}(h^{s_1}, g^{r_i}) \cdot \mathbf{e}(h^{s_3 \lambda_1}, g^{x_i v_i}) \\
&= \mathbf{e}(\mathsf{ct}_2, g^{r_i}) \cdot \mathbf{e}(h^{\lambda_1 s_3}, g^{x_i v_i})
\end{aligned}
$$

The same computation gives us

$$\mathbf{e}(H_{1,i}^{-s_2}\cdot \mathsf{ct}_{5,i}, K_{5,i})\cdot \mathbf{e}(H_{2,i}^{-s_2}\cdot \mathsf{ct}_{6,i}, K_{6,i}) = \mathbf{e}(\mathsf{ct}_2, g^{\Phi_i})\cdot \mathbf{e}(h^{\lambda_2 s_4}, g^{x_i v_i})$$

As a conclusion we have the following:

$$\begin{aligned}\mathbf{e}(\mathsf{ct}_1, K_A)\cdot \prod_{i=1}^{n} \mathbf{e}(\mathsf{ct}_{3,i}, K_{3,i})\cdot \mathbf{e}(\mathsf{ct}_{4,i}, K_{4,i})\cdot \mathbf{e}(\mathsf{ct}_{5,i}, K_{5,i})\cdot \mathbf{e}(\mathsf{ct}_{6,i}, K_{6,i}) =\\ = \lambda^{s_2}\cdot \prod_{i=1}^{n} \mathbf{e}(F_{1,i}^{-s_2}, K_{3,i})\mathbf{e}(F_{1,i}^{-s_2}, K_{4,i})\cdot \mathbf{e}(H_{1,i}^{-s_2}, K_{5,i})\cdot \mathbf{e}(H_{1,i}^{-s_2}, K_{6,i}) =\\ = \lambda^{s_2}\cdot \mathbf{e}(\mathsf{ct}_2, K_B^{-1})\cdot \mathbf{e}(h,g)^{(\lambda_1 s_3+\lambda_2 s_4)\langle \vec{x},\vec{v}\rangle}\end{aligned}$$

Plugging this into the decryption algorithm we obtain:

$$\Upsilon = m\cdot \mathbf{e}(h,g)^{(\lambda_1 s_3+\lambda_2 s_4)\langle \vec{x},\vec{v}\rangle},$$
$$\Upsilon' = m\cdot \mathbf{e}(h,g)^{(\lambda_1 s_3'+s_4\lambda_2)\langle \vec{x},\vec{v}\rangle}$$

First note that it cannot happen that $\mathsf{ct}_7 \neq \mathsf{ct}_7'$ for honestly generated ciphertexts. Clearly:

$$\langle \vec{x},\vec{v}\rangle = 0 \Rightarrow (\Upsilon = \Upsilon' = m).$$

All we need to check is thus that if $\langle \vec{x},\vec{v}\rangle \neq 0$, we get output $\perp$. We could only get a wrong output if $\Upsilon = \Upsilon'$, but this is impossible since it implies (using $\lambda_1 \neq 0, s_3 \neq s_3'$):

$$\begin{aligned}\mathbf{e}(h,g)^{(\lambda_1 s_3-\lambda_1 s_3')\langle \vec{x},\vec{v}\rangle} = 1_{\mathbb{G}_T} &\Rightarrow \lambda_1(s_3-s_3')\langle \vec{x},\vec{v}\rangle \equiv_p 0\\ &\Rightarrow \langle \vec{x},\vec{v}\rangle \equiv_p 0\,.\end{aligned} \tag{4.3}$$

□

4.7 Security Proof

This section proves our IPE scheme is IND-secure under the standard computational assumptions, DLin 6 and DBDH 7.

Theorem 4.7.1. *The IPE scheme* IP *of Construction 4.6 is IND-secure if the DBDH and DLin assumptions hold relative to* $\mathsf{GroupGen}$.

To prove the theorem, we define a series of hybrid experiments $\mathsf{H}_0,\ldots,\mathsf{H}_{12}$ in which H_0 corresponds to the real experiment with challenge bit $b=0$ and H_{12} corresponds to the real experiment with challenge bit $b=1$. We show that they are computationally indistinguishable.

Hybrid H_0: this hybrid is identical to the real game with challenge bit $b=0$. Precisely, the ciphertext is computed for message m_0 and vector $\overrightarrow{x}$ as follows:

$$\begin{aligned}\mathsf{ct} =&(g^{s_2}, h^{s_1}, \{W_{b,i}^{s_1}\cdot F_{b,i}^{s_2}\cdot U_b^{x_i s_3}, T_{b,i}^{s_1}\cdot H_{b,i}^{s_2}\cdot V_b^{x_i s_4}\}_{b\in[2],i\in[n]}, \mathbf{e}(g^{s_3}, g^{s_4}),\\ &\Lambda^{-s_2}\cdot m_0)\\ \mathsf{ct}' =&(g^{s_2'}, h^{s_1'}, \{W_{b,i}^{s_1'}\cdot F_{b,i}^{s_2'}\cdot U_b^{x_i s_3'}, T_{b,i}^{s_1'}\cdot H_{b,i}^{s_2'}\cdot V_b^{x_i s_4}\}_{b\in[2],i\in[n]}, \mathbf{e}(g^{s_3'}, g^{s_4}),\\ &\Lambda^{-s_2'}\cdot m_0)\end{aligned}$$

Hybrid H_1**:** this hybrid is identical to the previous hybrid, except that instead of

$$\mathbf{e}(g,g)^{s_3 s_4}, \mathbf{e}(g,g)^{s_3' s_4}$$

the ciphertext contains two random elements $R_1, R_1' \xleftarrow{\$} \mathbb{G}_T$. Precisely, the ciphertext is computed as follows:

$$\begin{aligned}\mathsf{ct} =& (g^{s_2}, h^{s_1}\{W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{x_i s_3}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{x_i s_4}\}_{b\in[2],i\in[n]}, \boxed{R_1}, \\ & \Lambda^{-s_2} \cdot m_0), \\ \mathsf{ct}' =& (g^{s_2'}, h^{s_1'}, \{W_{b,i}^{s_1'} \cdot F_{b,i}^{s_2'} \cdot U_b^{x_i s_3'}, T_{b,i}^{s_1'} \cdot H_{b,i}^{s_2'} \cdot V_b^{x_i s_4}\}_{b\in[2],i\in[n]}, \boxed{R_1'}, \\ & \Lambda^{-s_2'} \cdot m_0)\end{aligned}$$

Hybrid H_2**:** this hybrid is identical to the previous hybrid, except that instead of

$$\Lambda^{-s_2} \cdot m_0, \Lambda^{-s_2'} \cdot m_0$$

the ciphertext contains two random elements $R, R' \xleftarrow{\$} \mathbb{G}_T$. Precisely, the ciphertext is computed as follows:

$$\begin{aligned}\mathsf{ct} &= (g^{s_2}, h^{s_1}, \{W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{x_i s_3}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{x_i s_4}\}_{b\in[2],i\in[n]}, R_1, \boxed{R}), \\ \mathsf{ct}' &= (g^{s_2'}, h^{s_1'}, \{W_{b,i}^{s_1'} \cdot F_{b,i}^{s_2'} \cdot U_b^{x_i s_3'}, T_{b,i}^{s_1'} \cdot H_{b,i}^{s_2'} \cdot V_b^{x_i s_4}\}_{b\in[2],i\in[n]},, R_1', \boxed{R'})\end{aligned}$$

Hybrid H_3**:** this hybrid is identical to the previous hybrid, except that instead of

$$T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{x_i s_4}, T_{b,i}^{s_1'} \cdot H_{b,i}^{s_2'} \cdot V_b^{x_i s_4}$$

the ciphertext contains

$$T_{b,i}^{s_1} \cdot H_{b,i}^{s_2}, T_{b,i}^{s_1'} \cdot H_{b,i}^{s_2'}.$$

Precisely, the ciphertext is computed as follows:

$$\begin{aligned}\mathsf{ct} &= (g^{s_2}, h^{s_1}, \{W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{x_i s_3}, \boxed{T_{b,i}^{s_1} \cdot H_{b,i}^{s_2}}\}_{b\in[2],i\in[n]}, R_1, R), \\ \mathsf{ct}' &= (g^{s_2'}, h^{s_1'}, \{W_{b,i}^{s_1'} \cdot F_{b,i}^{s_2'} \cdot U_b^{x_i s_3'}, \boxed{T_{b,i}^{s_1'} \cdot H_{b,i}^{s_2'}}\}_{b\in[2],i\in[n]}, R_1', R')\end{aligned}$$

Hybrid H_4**:** this hybrid is identical to the previous hybrid, except that instead of

$$T_{b,i}^{s_1} \cdot H_{b,i}^{s_2}, T_{b,i}^{s_1'} \cdot H_{b,i}^{s_2'}$$

the ciphertext contains

$$T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{y_i s_4}, T_{b,i}^{s_1'} \cdot H_{b,i}^{s_2'} \cdot V_b^{y_i s_4}.$$

Precisely, the ciphertext is computed as follows:

$$\mathsf{ct} = (g^{s_2}, h^{s_1}, \{, W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{x_i s_3}, \boxed{T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{y_i s_4}}\}_{b\in[2],i\in[n]}, R_1, R),$$

$$\mathsf{ct}' = (g^{s_2'}, h^{s_1'}, \{W_{b,i}^{s_1'} \cdot F_{b,i}^{s_2'} \cdot U_b^{x_i s_3'},\ T_{b,i}^{s_1'} \cdot H_{b,i}^{s_2'} \cdot V_b^{y_i s_4}\}_{b\in[2], i\in[n]}, R_1', R')$$

Hybrid H_5**:** $\mathsf{CT}_6 = (\mathsf{ct}, \mathsf{ct}')$, This hybrid is identical to the previous hybrid, except that the power of V_b in ct is s_4 and its power in ct′ is s_4'. Precisely, the ciphertext is computed as follows:

$$\mathsf{ct} = (g^{s_2}, h^{s_1}, \{, W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{x_i s_3}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{y_i s_4}\}_{b\in[2], i\in[n]}, R_1, R),$$
$$\mathsf{ct}' = (g^{s_2'}, h^{s_1'}, \{W_{b,i}^{s_1'} \cdot F_{b,i}^{s_2'} \cdot U_b^{x_i s_3'},\ T_{b,i}^{s_1'} \cdot H_{b,i}^{s_2'} \cdot V_b^{y_i s_4'}\}_{b\in[2], i\in[n]}, R_1', R')$$

Hybrid H_6**:** this hybrid is identical to the previous hybrid, except that $s_3 = s_3'$. Precisely

$$\mathsf{ct} = (g^{s_2}, h^{s_1}, \{W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{x_i s_3}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{y_i s_4}\}_{b\in[2], i\in[n]}, R_1, R),$$
$$\mathsf{ct}' = (g^{s_2'}, h^{s_1'}, \{ W_{b,i}^{s_1'} \cdot F_{b,i}^{s_2'} \cdot U_b^{x_i s_3}, T_{b,i}^{s_1'} \cdot H_{b,i}^{s_2'} \cdot V_b^{y_i s_4'}\}_{b\in[2], i\in[n]}, R_1', R')$$

Hybrid H_7**:** This hybrid is identical to the previous hybrid, except we replace s_3 with 0.

$$\mathsf{ct} = (g^{s_2}, h^{s_1}, \{ W_{b,i}^{s_1} \cdot F_{b,i}^{s_2}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{y_i s_4}\}_{b\in[2], i\in[n]}, R_1, R),$$
$$\mathsf{ct}' = (g^{s_2'}, h^{s_1'}, \{ W_{b,i}^{s_1'} \cdot F_{b,i}^{s_2'}, T_{b,i}^{s_1'} \cdot H_{b,i}^{s_2'} \cdot V_b^{y_i s_4'}\}_{b\in[2], i\in[n]}, R_1', R')$$

Hybrid H_8**:** This hybrid is identical to the previous hybrid, except that instead of

$$W_{b,i}^{s_1} \cdot F_{b,i}^{s_2}, W_{b,i}^{s_1'} \cdot F_{b,i}^{s_2'}$$

we set

$$W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{y_i s_3}, W_{b,i}^{s_1'} \cdot F_{b,i}^{s_2'} \cdot U_b^{y_i s_3}.$$

Precisely

$$\mathsf{ct} = (g^{s_2}, h^{s_1}, \{ W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{y_i s_3}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{y_i s_4}\}_{b\in[2], i\in[n]}, R_1, R),$$
$$\mathsf{ct}' = (g^{s_2'}, h^{s_1'}, \{ W_{b,i}^{s_1'} \cdot F_{b,i}^{s_2'} \cdot U_b^{y_i s_3}, T_{b,i}^{s_1'} \cdot H_{b,i}^{s_2'} \cdot V_b^{y_i s_4'}\}_{b\in[2], i\in[n]}, R_1', R')$$

Hybrid H_9**:** this hybrid is identical to the previous hybrid, except that instead of

$$W_{b,i}^{s_1} \cdot F_{b,i}^{s_2}, W_{b,i}^{s_1'} \cdot F_{b,i}^{s_2'}$$

we set

$$W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{y_i s_3}, W_{b,i}^{s_1'} \cdot F_{b,i}^{s_2'} \cdot U_b^{y_i s_3'}.$$

Precisely

$$\mathsf{ct} = (g^{s_2}, h^{s_1}, \{W_{b,i}^{s_1} \cdot F_{b,i}^{s_2} \cdot U_b^{y_i s_3}, T_{b,i}^{s_1} \cdot H_{b,i}^{s_2} \cdot V_b^{y_i s_4}\}_{b\in[2], i\in[n]}, R_1, R),$$
$$\mathsf{ct}' = (g^{s_2'}, h^{s_1'}, \{ W_{b,i}^{s_1'} \cdot F_{b,i}^{s_2'} \cdot U_b^{y_i s_3'}, T_{b,i}^{s_1'} \cdot H_{b,i}^{s_2'} \cdot V_b^{y_i s_4'}\}_{b\in[2], i\in[n]}, R_1', R')$$

Hybrid H_{10}**:** this hybrid is identical to the previous hybrid, except that instead of

$$W_{b,i}^{s_1}\cdot F_{b,i}^{s_2}, W_{b,i}^{s_1'}\cdot F_{b,i}^{s_2'},$$

we set

$$W_{b,i}^{s_1}\cdot F_{b,i}^{s_2}\cdot U_b^{y_i s_3}, W_{b,i}^{s_1'}\cdot F_{b,i}^{s_2'}\cdot U_b^{y_i s_3'}.$$

Precisely

$$\begin{aligned}\mathsf{ct} &= (g^{s_2}, h^{s_1}, \{W_{b,i}^{s_1}\cdot F_{b,i}^{s_2}\cdot U_b^{y_i s_3}, T_{b,i}^{s_1}\cdot H_{b,i}^{s_2}\cdot V_b^{y_i s_4}\}_{b\in[2],i\in[n]}, R_1, R),\\ \mathsf{ct}' &= (g^{s_2'}, h^{s_1'}, \{W_{b,i}^{s_1'}\cdot F_{b,i}^{s_2'}\cdot U_b^{y_i s_3'}, \boxed{T_{b,i}^{s_1'}\cdot H_{b,i}^{s_2'}\cdot V_b^{y_i s_4}}\}_{b\in[2],i\in[n]}, R_1', R')\end{aligned}$$

Hybrid H_{11}**:** this hybrid is identical to the previous hybrid, except that instead of choosing

$$R, R' \xleftarrow{\$} \mathbb{G}_T,$$

we set

$$R = \Lambda^{-s_2}\cdot m_1, R' = \Lambda^{-s_2'}\cdot m_1.$$

Precisely, the ciphertext is computed as follows:

$$\begin{aligned}\mathsf{ct} &= (g^{s_2}, h^{s_1}, \{W_{b,i}^{s_1}\cdot F_{b,i}^{s_2}\cdot U_b^{y_i s_3}, T_{b,i}^{s_1}\cdot H_{b,i}^{s_2}\cdot V_b^{y_i s_4}\}_{b\in[2],i\in[n]}, R_1, \boxed{\Lambda^{-s_2}\cdot m_1}),\\ \mathsf{ct}' &= (g^{s_2'}, h^{s_1'}, \{W_{b,i}^{s_1'}\cdot F_{b,i}^{s_2'}\cdot U_b^{y_i s_3'}, T_{b,i}^{s_1'}\cdot H_{b,i}^{s_2'}\cdot V_b^{y_i s_4}\}_{b\in[2],i\in[n]}, R_1', \boxed{\Lambda^{-s_2'}\cdot m_1})\end{aligned}$$

Hybrid H_{12}**:** this hybrid is identical to the previous hybrid, except that instead of R_1, R_1', we set

$$\mathbf{e}(g^{s_3}, g^{s_4}), \mathbf{e}(g^{s_3'}, g^{s_4}),$$

which is identical to the real game with challenge bit $b = 1$, particularly for message m_1 and vector $\overrightarrow{y}$. Precisely, the ciphertext is computed as follows:

$$\begin{aligned}\mathsf{ct} =&(g^{s_2}, h^{s_1}, \{W_{b,i}^{s_1}\cdot F_{b,i}^{s_2}\cdot U_b^{y_i s_3}, T_{b,i}^{s_1}\cdot H_{b,i}^{s_2}\cdot V_b^{y_i s_4}\}_{b\in[2],i\in[n]}, \boxed{\mathbf{e}(g^{s_3}, g^{s_4})},\\ &\Lambda^{-s_2}\cdot m_1),\\ \mathsf{ct}' =&(g^{s_2'}, h^{s_1'}, \{W_{b,i}^{s_1'}\cdot F_{b,i}^{s_2'}\cdot U_b^{y_i s_3'}, T_{b,i}^{s_1'}\cdot H_{b,i}^{s_2'}\cdot V_b^{y_i s_4}\}_{b\in[2],i\in[n]}, \boxed{\mathbf{e}(g^{s_3'}, g^{s_4})},\\ &\Lambda^{-s_2'}\cdot m_1)\end{aligned}$$

Proposition 2. *If the DLin assumption holds relative to* GroupGen, *then* H_0 *is computationally indistinguishable from* H_1.

Proof. [3]

Let us assume there exists a PPT adversary $\mathcal{A}$ which distinguishes between H_0 and H_1 with non-negligible advantage. We describe a simulator $\mathcal{B}$ which uses $\mathcal{A}$, on input

$$(g, A = g^{\alpha}, B = g^{\beta}, C = g^{\tau}, D = g^{\alpha\eta}, Z) \in \mathbb{G}^6,$$

and would output 1 if $Z = g^{\beta(\eta+\tau)}$ and 0 if Z is a random element in $\mathbb{G}$.

[3]The proof is inspired by the paper [217]

$\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

Set Up phase. The adversary $\mathcal{A}$ sends to the simulator, $\mathcal{B}$, two vectors $\overrightarrow{x}, \overrightarrow{y} \in \mathbb{Z}_p^n$. The simulator picks

$$g' \xleftarrow{\$} \mathbb{G}, \tilde{\Omega}, k, \tilde{\delta}_b, \theta_b, \{w_{1,i}, \tilde{t}_{1,i}, f_{b,i}, h_{b,i}\}_{i\in[n], b\in[2]} \xleftarrow{\$} \mathbb{Z}_p,$$

compute $\{w_{2,i}, \tilde{t}_{2,i}\}_{i\in[n]}$ such that for each i:

$$\tilde{\Omega} = \tilde{\delta}_1 w_{2,i} - \tilde{\delta}_2 w_{1,i} = \theta_1 \tilde{t}_{2,i} - \theta_2 \tilde{t}_{1,i}.$$

Compute the master public key components as follows and returns it:

$$\begin{aligned}
&\{W_{b,i} = g^{w_{b,i}}, F_{b,i} = g^{f_{b,i}}\}_{b\in[2], i\in[n]}, \{U_b = A^{\tilde{\delta}_b}\}_{b\in[2]},\\
&\{T_{b,i} = A^{\tilde{t}_{b,i}}, H_{b,i} = g^{h_{b,i}}\}_{b\in[2], i\in[n]}, \{V_b = g^{\theta_b}\}_{b\in[2]},\\
&h = A^{\tilde{\Omega}}, \Lambda = \mathbf{e}(g, g'), K_1 = g^k,\ K_2 = g'^{\frac{1}{k}}
\end{aligned} \tag{4.4}$$

By doing so, $\mathcal{B}$ implicitly sets:

$$\delta_b = \alpha\tilde{\delta}_b, t_{b,i} = \alpha\tilde{t}_{b,i} \text{ for } b \in [2], i \in [n] \text{ and } \Omega = \alpha\tilde{\Omega}$$

which shows that each element of the master public key is independently and uniformly distributed in $\mathbb{Z}_p$. Also, notice that for each $i \in [n]$, we have:

$$\begin{aligned}
\delta_1 w_{2,i} - \delta_2 w_{1,i} &= \alpha\tilde{\delta}_1 w_{2,i} - \alpha\tilde{\delta}_2 w_{1,i}\\
&= \theta_1 \alpha\tilde{t}_{2,i} - \theta_2 \alpha\tilde{t}_{1,i}\\
&= \theta_1 t_{2,i} - \theta_2 t_{1,i}\\
&= \alpha\tilde{\Omega}\\
&= \Omega.
\end{aligned} \tag{4.5}$$

Hence the output has the same structure as the output of the real setup algorithm.

Token query phase. Note that all the secret parameters except

$$\{\delta_b, t_{b,i}\}_{b\in[2], i\in[n]}, \Omega$$

are known by $\mathcal{B}$. When $\mathcal{A}$ asks for a query for a vector $\overrightarrow{v}$, $\mathcal{B}$ picks:

$$\lambda_1, \tilde{\lambda}_2, \{\tilde{r}_i, \Phi_i\}_{i\in[n]} \xleftarrow{\$} \mathbb{Z}_p^\star.$$

. When generating $\mathsf{Tok}_{\overrightarrow{v}}$, the simulator implicitly sets:

$$\lambda_2 = \alpha\tilde{\lambda}_2, r_i = \alpha\tilde{r}_i$$

which are independently and uniformly distributed in $\mathbb{Z}_p^\star$. Token elements are set as follows:

$$\begin{aligned}
K_{3,i} &= A^{-\tilde{\delta}_2 r_i} \cdot g^{\lambda_1 v_i w_{2,i} x_i} = \text{(by the above settings)} = g^{-\delta_2 r_i} \cdot g^{v_i w_{2,i} \lambda_1}.\\
K_{5,i} &= g^{-\theta_2 \phi_i} \cdot A^{\lambda_2 v_i \tilde{t}_{2,i} x_i} = \text{(by the above settings)} = g^{-\theta_2 \phi_i} \cdot g^{\lambda_2 v_i t_{2,i} x_i}.\\
\text{Similarly, } K_{4,i} &= A^{\tilde{\delta}_1 r_i} \cdot g^{-\lambda_1 v_i w_{1,i} x_i},\\
K_{6,i} &= g^{\theta_1 r_i} \cdot A^{-\lambda_2 v_i \tilde{t}_{1,i} x_i}.\\
K_B &= \prod_{i=1}^{n} A^{-r_i} g^{-\Phi_i} = \prod_{i=1}^{n} g^{-(\alpha \tilde{r}_i + \Phi_i)} = \prod_{i=1}^{n} g^{-(r_i + \Phi_i)}.
\end{aligned}$$

$\mathcal{B}$ knows $\{f_{b,i}, h_{b,i}\}_{b \in [2], i \in [n]}$, hence it can compute K_A.

Generating the challenge ciphertext. $\mathcal{A}$ sends message m_0 to $\mathcal{B}$. To generate a challenge ciphertext, $\mathcal{B}$ picks

$$s_1, s_2, s_1', s_2', \tilde{s}_3, \tilde{s}_4, \tilde{s}_3' \xleftarrow{\$} \mathbb{Z}_p^\star$$

such that:

$$\tilde{s}_3 \neq \tilde{s}_3'.$$

In fact, $\mathcal{B}$ implicitly sets:

$$s_3 = \eta \tilde{s}_3, s_4 = \beta \tilde{s}_4$$

and computes the ciphertext as follows:

$$\begin{aligned}
&\mathsf{ct}_1 = g^{s_2}, && \mathsf{ct}_1' = g^{s_2'},\\
&\mathsf{ct}_2 = h^{s_1}, && \mathsf{ct}_2' = h^{s_1'},\\
&\mathsf{ct}_{3,i} = W_{1,i}^{s_1} \cdot F_{1,i}^{s_2} \cdot D^{\tilde{\delta}_1 \tilde{s}_3 x_i}, && \mathsf{ct}_{3,i}' = W_{1,i}^{s_1'} \cdot F_{1,i}^{s_2'} \cdot D^{\tilde{\delta}_1 x_i \tilde{s}_3'},\\
&\mathsf{ct}_{4,i} = W_{2,i}^{s_1} \cdot F_{2,i}^{s_2} \cdot D^{\tilde{\delta}_2 \tilde{s}_3 x_i}, && \mathsf{ct}_{4,i}' = W_{2,i}^{s_1'} \cdot F_{2,i}^{s_2'} \cdot D^{\tilde{\delta}_2 x_i \tilde{s}_3'},\\
&\mathsf{ct}_{5,i} = T_{1,i}^{s_1} \cdot H_{1,i}^{s_2} \cdot B^{\theta_1 \tilde{s}_4 x_i}, && \mathsf{ct}_{5,i}' = T_{1,i}^{s_1'} \cdot H_{1,i}^{s_2'} \cdot B^{\theta_1 \tilde{s}_4 x_i},\\
&\mathsf{ct}_{6,i} = T_{2,i}^{s_1} \cdot H_{2,i}^{s_2} \cdot B^{\theta_2 \tilde{s}_4 x_i}, && \mathsf{ct}_{6,i}' = T_{2,i}^{s_1'} \cdot H_{2,i}^{s_2'} \cdot B^{\theta_2 \tilde{s}_4 x_i},\\
&\mathsf{ct}_7 = (\tfrac{\mathbf{e}(Z,g)}{\mathbf{e}(B,C)})^{\tilde{s}_3 \tilde{s}_4}, && \mathsf{ct}_7' = (\tfrac{\mathbf{e}(Z,g)}{\mathbf{e}(B,C)})^{\tilde{s}_3' \tilde{s}_4},\\
&\mathsf{ct}_8 = \mathbf{e}(g, g')^{-s_2} \cdot m_0 && , \mathsf{ct}_8' = \mathbf{e}(g, g')^{-s_2'} \cdot m_0.
\end{aligned}$$

Since

$$\begin{aligned}
D^{\tilde{\delta}_b x_i \tilde{s}_3} &= g^{\alpha \tilde{\delta}_b \eta \tilde{s}_3 x_i} = U_b^{x_i s_3}\\
B^{\theta_b \tilde{s}_4 x_i} &= V_1^{\beta \tilde{s}_4 x_i} = V_b^{s_4 x_i}
\end{aligned}$$

for each $i \in [n]$ the values $\mathsf{ct}_{3,i}, \mathsf{ct}_{3,i}', \ldots, \mathsf{ct}_{6,i}, \mathsf{ct}_{6,i}'$ are computed properly.

Analyzing the game. Let us analyze the two events, $Z = g^{\beta(\tau+\eta)}$ and $Z \xleftarrow{\$} \mathbb{G}$:

1. $Z = g^{\beta(\tau+\eta)}$:

$$\begin{aligned}
\Rightarrow \frac{\mathbf{e}(Z,g)}{\mathbf{e}(B,C)} &= \frac{\mathbf{e}(g^{\beta(\tau+\eta)},g)}{\mathbf{e}(g^{\beta},g^{\tau})} \\
&= \frac{\mathbf{e}(g^{\beta},g^{\tau})\cdot\mathbf{e}(g^{\beta},g^{\eta})}{\mathbf{e}(g^{\beta},g^{\tau})} \\
&= \mathbf{e}(g^{\eta},g^{\beta}) \\
\Rightarrow \mathsf{ct}_7 &= (\frac{\mathbf{e}(Z,g)}{\mathbf{e}(B,C)})^{\tilde{s}_3\tilde{s}_4} \\
&= \mathbf{e}(g^{\eta\tilde{s}_3},g^{\beta\tilde{s}_4}) \\
&= \mathbf{e}(g^{s_3},g^{s_4}), \\
\mathsf{ct}'_7 &= \mathbf{e}(g^{s'_3},g^{s_4}) \qquad \text{(Same computation as above)} \\
&\Rightarrow \mathcal{A} \text{ interacting with } \mathsf{H}_0.
\end{aligned}$$

2. $Z \xleftarrow{\$} \mathbb{G}$, since A is a random element then $\mathsf{ct}_7, \mathsf{ct}'_7$ are also random elements in $\mathbb{G}_T$ which implies that adversary $\mathcal{A}$ interacts with H_1.

□

Proposition 3. *If the DBDH assumption holds relative to* GroupGen*, then* H_1 *is computationally indistinguishable from* H_2.

Proposition 4. *If the DLin assumption holds relative to* GroupGen*, then* H_2 *is computationally indistinguishable from* H_3.

Proposition 5. *If the DLin assumption holds relative to* GroupGen*, then* H_3 *is computationally indistinguishable from* H_4.

Propositions 3,7,8 are proved in the Appendix 9.6.

Proposition 6. *If the DLin assumption holds relative to* GroupGen*, then* H_4 *is computationally indistinguishable from* H_5.

Proof. The simulator takes as input

$$(g, A = g^{\alpha}, B = g^{\beta}, C = g^{\tau}, D = g^{\alpha\eta}, Z \stackrel{?}{=} g^{\beta(\eta+\tau)})$$

and by interacting with the adversary $\mathcal{A}$, distinguish between the two cases $Z = g^{\beta(\eta+\tau)}$ and $Z \xleftarrow{\$} \mathbb{G}$, a random element of the group.

SetUp and token query phase. $\mathcal{B}$ runs as in the SetUp phase and token query phase in proposition 8.

Generating the challenge ciphertext. $\mathcal{B}$ chooses random elements

$$\tilde{s}_1, \tilde{s}_2, \tilde{s}_3, \tilde{s}_4, \tilde{s}'_1, \tilde{s}'_2, \tilde{s}'_3, k \xleftarrow{\$} \mathbb{Z}_p^*$$

and computes the challenge ciphertext as follows:

$$
\begin{aligned}
\mathsf{ct}_1 &= C \cdot g^{\tilde{s}_2} = g^{\tau+\tilde{s}_2} \\
&\Rightarrow s_2 = \tau + \tilde{s}_2, \\
\mathsf{ct}'_1 &= C^k \cdot g^{\tilde{s}'_2} = g^{k\tau+\tilde{s}'_2} \\
&\Rightarrow s'_2 = k\tau + \tilde{s}_2 \\
\mathsf{ct}_2 &= D^{\tilde{\Omega}} \cdot A^{\tilde{\Omega}\tilde{s}_1} = (g^{\alpha\tilde{\Omega}})^{(\eta+\tilde{s}_1)} = h^{\eta+\tilde{s}_1} \\
&\Rightarrow s_1 = \eta + \tilde{s}_1 \\
\mathsf{ct}'_2 &= D^{k\tilde{\Omega}} \cdot A^{\tilde{\Omega}\tilde{s}'_1} = (g^{\alpha\tilde{\Omega}})^{(k\eta+\tilde{s}'_1)} = h^{k\eta+\tilde{s}'_1} \\
&\Rightarrow s'_1 = k\eta + \tilde{s}'_1 \\
\mathsf{ct}_{3,i} &= W_{1,i}^{\tilde{s}_1} \cdot F_{1,i}^{\tilde{s}_2} \cdot U_1^{\tilde{s}_3 x_i} \cdot D^{\tilde{w}_{1,i}} \cdot C^{f_{1,i}} \\
&= W_{1,i}^{\tilde{s}_1} \cdot F_{1,i}^{\tilde{s}_2+\tau} \cdot U_1^{\tilde{s}_3 x_i} \cdot g^{\eta\alpha\tilde{w}_{1,i}} \cdot F_{1,i}^{\tau} \\
&= W_{1,i}^{\tilde{s}_1} \cdot F_{1,i}^{\tilde{s}_2+\tau} \cdot U_1^{\tilde{s}_3 x_i} \cdot g^{\eta(w_{1,i}-\beta\delta_1 x_i)} \\
&= W_{1,i}^{\tilde{s}_1+\eta} \cdot F_{1,i}^{\tilde{s}_2+\tau} \cdot U_1^{(\tilde{s}_3-\eta\beta)x_i} \\
&\Rightarrow s_3 = -\eta\beta + \tilde{s}_3 \\
\mathsf{ct}_{4,i} &= W_{2,i}^{\tilde{s}_1} \cdot F_{2,i}^{\tilde{s}_2} \cdot U_2^{\tilde{s}_3 x_i} \cdot D^{\tilde{w}_{2,i}} \cdot C^{f_{2,i}}, (\text{ similar computation as } \mathsf{ct}_{3,i}) \\
\mathsf{ct}'_{3,i} &= W_{1,i}^{\tilde{s}'_1} \cdot F_{1,i}^{\tilde{s}'_2} \cdot U_1^{\tilde{s}'_3 x_i} \cdot D^{k\tilde{w}_{1,i}} \cdot C^{kf_{1,i}} \\
&= W_{1,i}^{\tilde{s}'_1} \cdot F_{1,i}^{\tilde{s}'_2} \cdot U_1^{\tilde{s}'_3 x_i} \cdot g^{k\eta\alpha\tilde{w}_{1,i}} \cdot F_{1,i}^{k\tau} \\
&= W_{1,i}^{\tilde{s}'_1} \cdot F_{1,i}^{\tilde{s}'_2+k\tau} \cdot U_1^{\tilde{s}'_3 x_i} \cdot g^{k\eta(w_{1,i}-\beta\delta_1 x_i)} \\
&= W_{1,i}^{\tilde{s}'_1+k\eta} \cdot F_{1,i}^{\tilde{s}'_2+k\tau} \cdot U_1^{(\tilde{s}'_3-k\eta\beta)x_i} \\
&\Rightarrow s'_3 = -k\eta\beta + \tilde{s}'_3 \\
\mathsf{ct}'_{4,i} &= W_{2,i}^{\tilde{s}'_1} \cdot F_{2,i}^{\tilde{s}'_2} \cdot U_2^{\tilde{s}'_3 x_i} \cdot D^{k\tilde{w}_{2,i}} \cdot C^{kf_{2,i}}, (\text{ similar computation as } \mathsf{ct}'_{3,i}) \\
\mathsf{ct}_{5,i} &= T_{1,i}^{\tilde{s}_1} \cdot D^{\tilde{t}_{1,i}} \cdot H_{1,i}^{\tilde{s}_2} \cdot C^{\tilde{h}_{1,i}} \cdot Z^{\theta_1 y_i} \cdot g^{\tilde{s}_4\theta_1 y_i} \\
\mathsf{ct}'_{5,i} &= T_{1,i}^{\tilde{s}'_1} \cdot D^{k\tilde{t}_{1,i}} \cdot H_{1,i}^{\tilde{s}'_2} \cdot C^{k\tilde{h}_{1,i}} \cdot Z^{k\theta_1 y_i} \cdot g^{\tilde{s}_4\theta_1 y_i} \\
\mathsf{ct}_{6,i} &= T_{2,i}^{\tilde{s}_1} \cdot D^{\tilde{t}_{2,i}} \cdot H_{2,i}^{\tilde{s}_2} \cdot C^{\tilde{h}_{2,i}} \cdot Z^{\theta_2 y_i} \cdot g^{\tilde{s}_4\theta_2 y_i} \\
\mathsf{ct}'_{6,i} &= T_{2,i}^{\tilde{s}'_1} \cdot D^{k\tilde{t}_{2,i}} \cdot H_{2,i}^{\tilde{s}'_2} \cdot C^{k\tilde{h}_{2,i}} \cdot Z^{k\theta_2 y_i} \cdot g^{\tilde{s}_4\theta_2 y_i}
\end{aligned}
$$

Analysis of the game. First, notice that:

$$
\begin{aligned}
D^{\tilde{t}_{1,i}} &= g^{\eta\alpha\tilde{t}_{1,i}} = g^{\eta(t_{1,i}-\beta\theta_1 y_i)} \\
&= T_{1,i}^{\eta} \cdot g^{-\beta\eta\theta_1 y_i}, \\
D^{k\tilde{t}_{1,i}} &= T_{1,i}^{k\eta} \cdot g^{-k\beta\eta\theta_1 y_i} \\
C^{\tilde{h}_{1,i}} &= g^{\tau(h_{1,i}-\beta\theta_1 y_i)} \\
&= H_{1,i}^{\tau} \cdot g^{-\beta\tau\theta_1 y_i}, \\
C^{k\tilde{h}_{1,i}} &= H_{1,i}^{k\tau} \cdot g^{-k\beta\tau\theta_1 y_i}
\end{aligned}
$$

Therefore:

$$\begin{aligned} \mathsf{ct}_{5,i} &= T_{1,i}^{\tilde{s}_1}\cdot D^{\tilde{t}_{1,i}}\cdot H_{1,i}^{\tilde{s}_2}\cdot C^{\tilde{h}_{1,i}}\cdot (Z\cdot g^{\tilde{s}_4})^{\theta_1 y_i} \\ &= T_{1,i}^{\eta+\tilde{s}_1}\cdot H_{1,i}^{\tau+\tilde{s}_2}\cdot (g^{-\beta(\tau+\eta)}\cdot Z\cdot g^{\tilde{s}_4})^{\theta_1 y_i} \\ &= T_{1,i}^{s_1}\cdot H_{1,i}^{s_2}\cdot (g^{(-\beta(\tau+\eta)}\cdot Z\cdot g^{\tilde{s}_4})^{\theta_1 y_i} \\ \mathsf{ct}'_{5,i} &= T_{1,i}^{s'_1}\cdot H_{1,i}^{s'_2}\cdot (g^{(-k\beta(\tau+\eta)}\cdot Z^k\cdot g^{\tilde{s}_4})^{\theta_1 y_i} \end{aligned}$$

Now we consider the following two cases:

1. $Z = g^{\beta(\eta+\tau)}$:

$$\begin{aligned} g^{-\beta(\tau+\eta)}\cdot Z\cdot g^{\tilde{s}_4} = g^{\tilde{s}_4} \quad &\Rightarrow \quad \mathsf{ct}_{5,i} = T_{1,i}^{s_1}\cdot H_{1,i}^{s_2}\cdot U_1^{s_4 y_i} \\ g^{-k\beta(\tau+\eta)}\cdot Z^k\cdot g^{\tilde{s}_4} = g^{\tilde{s}_4} \quad &\Rightarrow \quad \mathsf{ct}_{5,i} = T_{1,i}^{s'_1}\cdot H_{1,i}^{s'_2}\cdot U_1^{s_4 y_i} \\ &\Rightarrow \text{ The adversary interacts with hybrid } \mathsf{H}_4 \end{aligned}$$

2. $Z = g^r$:

$$\begin{aligned} g^{-\beta(\tau+\eta)}\cdot Z\cdot g^{\tilde{s}_4} = g^{r+\tilde{s}_4} &\Rightarrow \mathsf{ct}_{5,i} = T_{1,i}^{s_1}\cdot H_{1,i}^{s_2}\cdot U_1^{s_4 y_i} \\ g^{-k\beta(\tau+\eta)}\cdot Z^k\cdot g^{\tilde{s}_4} = g^{kr+\tilde{s}_4} &\Rightarrow \mathsf{ct}_{5,i} = T_{1,i}^{s'_1}\cdot H_{1,i}^{s'_2}\cdot U_1^{s'_4 y_i} \\ &\Rightarrow \text{ The adversary interacts with hybrid } \mathsf{H}_5. \end{aligned}$$

$$\begin{cases} g^{-\beta(\tau+\eta)}\cdot Z\cdot g^{\tilde{s}_4} = g^{r+\tilde{s}_4} \Rightarrow \mathsf{ct}_{5,i} = T_{1,i}^{s_1}\cdot H_{1,i}^{s_2}\cdot U_1^{s_4 y_i} \\ g^{(-k\beta(\tau+\eta)}\cdot Z^k\cdot g^{\tilde{s}_4} = g^{kr+\tilde{s}_4} \Rightarrow \mathsf{ct}_{5,i} = T_{1,i}^{s'_1}\cdot H_{1,i}^{s'_2}\cdot U_1^{s'_4 y_i} \end{cases}$$
$\Rightarrow$ The adversary interacts with hybrid H_5.

□

The proofs of indistinguishability for the other hybrids are the same as we have proved, Table 4.10:

FIGURE 4.10: The indistinguishability of two games in the left side is proven similar to the right side.

$\mathsf{H}_5 - \mathsf{H}_6$	Same as $\mathsf{H}_4 - \mathsf{H}_5$	$\mathsf{H}_9 - \mathsf{H}_{10}$	Same as $\mathsf{H}_4 - \mathsf{H}_5$
$\mathsf{H}_6 - \mathsf{H}_7$	Same as $\mathsf{H}_3 - \mathsf{H}_4$	$\mathsf{H}_{10} - \mathsf{H}_{11}$	Same as $\mathsf{H}_1 - \mathsf{H}_2$
$\mathsf{H}_7 - \mathsf{H}_8$	Same as $\mathsf{H}_2 - \mathsf{H}_3$	$\mathsf{H}_{11} - \mathsf{H}_{12}$	Same as $\mathsf{H}_0 - \mathsf{H}_1$
$\mathsf{H}_8 - \mathsf{H}_9$	Same as $\mathsf{H}_5 - \mathsf{H}_4$		

Chapter 5

Verifiable IPE

"trust, but VERIFY! "[1]

We encrypt our data, obfuscate our programs, anonymize our identities, and apply many layers of masking to our images, all while communicating anonymously.

We do, indeed, inhabit a cryptic universe.
Even though we feel safe with much security, there is a dark side to the story. *Too* much security also protects the malicious parties, so we need to start over and we need to think about this question:

Do we feel secure , in a world where everything is encrypted?

Contemplating of this question may imply that security is not sufficient and that, in order to benefit from a secure protocols, we must establish another notion, namely, ***verifiability***.

This chapter will go over verifiability in further detail, followed by introducing our verifiable IPE.

Contents

[1]`https://en.wikipedia.org/wiki/Trust,_but_verify`

As an example of the need for verifiability, let us consider an election with a complex ballot that employs the risk-limiting tally technique [227], discussed in detail in 9, *i.e.*, where only a random subset of the components of the ballots are revealed during the tally phase.

Now, consider that we employ a functional encryption scheme in this election as follows. Each voter encrypts her favorite choice as a 0/1 vector $\mathsf{vote} = (v_1, \ldots, v_5)$; $v_i \in \{0,1\}$, for the candidate list $(\mathsf{c}_1, \ldots, \mathsf{c}_5)$ and submits the ciphertext CT.

Two tellers $\mathcal{T}_1$ and $\mathcal{T}_2$ are given tokens tok_1 and tok_2; each token decrypts different components of CT. Let say tok_1 decrypts $(\mathsf{ct}_1, \mathsf{ct}_3, \mathsf{ct}_4)$ and tok_2 opens $(\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_5)$. In the case of functional encryption, a dishonest party in charge of the token generation algorithm may provide defective tokens for either $\mathcal{T}_1$ or $\mathcal{T}_2$ in order to manipulate the election result or undermine election validity. As a result, the following could occur:

$$\begin{aligned}\mathsf{vote} = (1,1,0,1,0),\ \mathsf{CT} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4, \mathsf{ct}_5) :&\\ \mathcal{T}_1(\mathsf{CT}, \mathsf{tok}_1) &\mapsto (0, *, 0, 1, *),\\ \mathcal{T}_2(\mathsf{CT}, \mathsf{tok}_2) &\mapsto (1, 0, *, *, 0).\end{aligned}$$

As the above computations demonstrate, at least one of the tokens must be defective due to the incompatibility of two outputs. In other words, the ciphertext CT cannot be the encryption of a single message. And in our example, this could result in one of two effects: either the election outcome does not accurately reflect the real voter's choice, or the election's validity is called into doubt.

It is worth mentioning that these flaws and inconsistencies are undetectable in distributed protocols that contain bulk data or stand-alone data. However they have a significant impact on procedure outcomes. This suggests that to take advantage of a secure cryptosystem, we need to define another concept, ***verifiability***. Informally, in the above example, verifiability ensures that there is a unique vote such that for every token, the decryption algorithm outcome would stem from that unique vote.

5.1 Introduction and Research Question

In FE and IPE, the encryptors and the Central Authority (CA) that generate the tokens are assumed to be honest. Indeed, as noticed by Badrinarayanan *et al.*, in the presence of any dishonest party (that is, either the party that generates the token or the party who encrypts the message), the decryption outputs may be inconsistent, and this raises serious issues in practical applications (e.g., auditing). For instance, a dishonest authority might be able to generate a faulty token $\mathsf{Tok}_{\vec{v}}$ for a vector $\vec{v} = (1,0,3,0)$ such that $\mathsf{Tok}_{\vec{v}}$ enables the owner to decrypt the ciphertext, CT_1 for a vector $\vec{x}_1 = (0,1,0,0)$ but using the same token in decryption algorithm for the ciphertext CT_2, generated with respect to the vector $\vec{x}_2 = (0,2,0,1)$, outputs an error. Or a dishonest encryptor might generate a faulty ciphertext that decrypts to an incorrect result with an honestly computed token. These issues are particularly severe in the applications to functional commitments that we will see later.

Verifiable Inner Product Encryption (VIPE) overcomes those limitations by adding strong verifiability guarantees to IPE. VIPE is a special case of Verifiable Functional Encryption (VFE), firstly proposed by Badrinarayanan *et al.* [23] for general functionalities. Informally speaking, in VIPE, there are public verification algorithms to verify that the output of the setup, encryption and token generation algorithms are computed honestly. Intuitively, suppose the master public key MPK and a ciphertext CT pass a public verification test. In this case, it means a message m and a unique vector $\vec{x}$ – up to parallelism – exist such that for all vectors $\vec{v}$, if a token $\mathsf{Tok}_{\vec{v}}$ for $\vec{v}$ is accepted by the verification algorithm then the following holds:

$$\forall \vec{v} : \mathsf{Dec}(\mathsf{Tok}_{\vec{v}}, \mathsf{CT}) = f_{\vec{v}}(\vec{x}, m).$$

The main components we employ for constructing a VIPE scheme are, our perfectly correct IPE scheme 4 and Groth-Sahai NIWI-proof system 3.10.3.

Outline. In Section 5.3, we first formally define the verifiable Inner product Encryption scheme, its security notion and some impossibility result regarding Verifiability. Then as motivation we will present some of its applications in section 5.2. In Section 5.4 we present the relation we use for our construction. And finally we will give a full description of our construction in section 5.5. Finally, this chapter will end with a description of the verification algorithm in Section 5.6.

5.2 Motivating Applications

IPE has numerous applications, including Anonymous Identity-Based Encryption [59], Hidden-Vector Encryption [56], and predicate encryption schemes supporting polynomial evaluation [173]. Badrinarayanan *et al.* [23] show that making FE schemes verifiable enables more powerful applications. As an example, in this section, we show that VIPE can be used to construct what we call a *polynomial commitment scheme*, which corresponds to a functional commitment of Badrinarayanan *et al.* for the polynomial evaluation *predicate*. The same construction can easily be adapted to construct functional commitments for the inner-product predicate.

5.2.1 Perfectly Binding Polynomial Commitments

Using a polynomial commitment scheme (see also [170]), Alice may publish a commitment to a polynomial $\mathsf{poly}(x)$ with coefficients in $\mathbb{Z}_p$. If later Bob wants to know $\mathsf{poly}(m)$ for some value m, that is the evaluation of the polynomial at some point; he sends m to Alice, who replies with the claimed evaluation y and proof that $y = \mathsf{poly}(m)$. The proof guarantees that the claimed evaluation is consistent with the committed polynomial. We require the scheme to be *perfectly binding*.

We construct a polynomial commitment scheme for polynomials of degree at most d from a $\Pi^{\mathsf{vip}} = \langle \mathsf{VIP.SetUp}, \mathsf{VIP.TokGen}, \mathsf{VIP.Enc}, \mathsf{VIP.Dec} \rangle$ scheme for vectors of dimension $d+2$ in the following way:

- ***Commitment Phase:*** To commit to a polynomial $\mathsf{poly} \in \mathbb{Z}_p[x]$ Alice performs the following steps:
  1. Define the vector $\vec{x}$:
  $$\mathsf{poly}(x) = a_d x^d + a_{d-1} x^{d-1} + \ldots + a_1 x + a_0 \in \mathbb{Z}_p[X]$$
  $$\Longrightarrow \overrightarrow{x} := (a_d, a_{d-1}, \ldots, a_1, a_0, 1) \in \mathbb{Z}_p^{d+2}$$
  2. Run $\mathsf{VIP.SetUp}(1^\lambda, d+2)$ to generate $(\mathsf{MPK}, \mathsf{MSK})$

3. Run the encryption algorithm for the attribute $\vec{x}$:

$$\mathsf{CT} \rightarrow \mathsf{VIP.Enc}(\mathsf{MPK}, \overrightarrow{x})$$

4. Output the commitment $\mathsf{com} := (\mathsf{MPK}, \mathsf{CT})$.

- ***Opening phase:***

1. Bob requests a query (m, y) to check if the commitment corresponds to a polynomial poly such that $\mathsf{poly}(m) = y$.
2. Alice, the Committer runs the token-generator algorithm of VIP for vector

$$\overrightarrow{v} := (m^d, m^{d-1}, \ldots, m, 1, -y)$$

and sends $\mathsf{Tok}_{\overrightarrow{v}}$ as the opening.
3. Bob would accept the opening if and only if $\mathsf{VIP.Dec}(\mathsf{CT}, \mathsf{Tok}_{\overrightarrow{v}}) = 0$.

- ***Correctness property*** is implied from the following computations:

$$\begin{aligned}\langle \vec{x}, \vec{v} \rangle &= a_d m^d + a_{d-1} m^{d-1} + \ldots + a_1 m + a_0 - y \\ &= \mathsf{poly}(m) - y \\ &\implies \mathsf{VIP.Dec}(\mathsf{CT}, \mathsf{Tok}_{\overrightarrow{v}}) = 0 \text{ iff } \mathsf{poly}(m) = y\end{aligned}$$

It is straightforward to see that the above algorithms form a functional commitment (in the sense of [23]) for the polynomial evaluation predicate. Therefore we refer the reader to [23] for more details on functional commitments.

5.3 Verifiability in the Context of Functional Encryption

Firstly, we present a formal definition of a VIPE scheme. Essentially, VIPE is similar to IPE, except it is endowed with extra verification algorithms $\mathsf{VerifyCT}, \mathsf{Verifytok}$ and $\mathsf{VerifyMPK}$.

Definition 35. *A verifiable inner product encryption scheme for a message space $\mathcal{M}$ and a family $\Sigma = \{\Sigma_n\}_{n>0}$ of vectors over some field is a tuple of PPT algorithms (here called* VIP*)*

$$\Pi^{\mathsf{vip}} = \{\mathsf{SetUp}, \mathsf{TokGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{VerifyMPK}, \mathsf{VerifyCT}, \mathsf{Verifytok}\} \tag{5.1}$$

with the syntax and properties below:

- $\mathsf{SetUp}(1^\ell, n) \rightarrow (\mathsf{MPK}, \mathsf{MSK})$*: as for IPE.*
- $\mathsf{TokGen}(\mathsf{MPK}, \mathsf{MSK}, \vec{v}) \longrightarrow \mathsf{tok}_{\vec{v}}$*: as for IPE.*
- $\mathsf{Enc}(\mathsf{MPK}, \overrightarrow{x}, m) \rightarrow \mathsf{CT}$*: as for IPE.*
- $\mathsf{Dec}(\mathsf{MPK}, \mathsf{tok}_{\vec{v}}, \mathsf{CT}) \rightarrow m \in \mathcal{M} \cup \{\perp\}$*: as for IPE.*
- $\mathsf{VerifyMPK}(\mathsf{MPK}) \rightarrow \{0,1\}$*: this is a deterministic algorithm that outputs* 1 *if* MPK *was correctly generated, or outputs* 0 *otherwise.*
- $\mathsf{VerifyCT}(\mathsf{MPK}, \mathsf{CT}) \rightarrow \{0,1\}$*: this is a deterministic algorithm that outputs* 1 *if* CT *was correctly generated using the master public key on input some* m *in the message space* $\mathcal{M}$ *and a vector* $\vec{x}$*, or outputs* 0 *otherwise.*

- $\mathsf{Verifytok}(\mathsf{MPK}, \vec{v}, \mathsf{tok}_{\vec{v}}) \longrightarrow \{0,1\}$*: this is a deterministic algorithm that outputs* 1 *if* $\mathsf{tok}_{\vec{v}}$ *was correctly generated using the master secret key on input vector* $\vec{v}$*, or outputs* 0 *otherwise.*

- Perfect correctness: *as for IPE.*

- Verifiability: VIP *is verifiable if for all* $\mathsf{MPK} \in \{0,1\}^*$*, all* $\mathsf{CT} \in \{0,1\}^*$*, there exists* $n > 0, (\vec{x}, m) \in \Sigma_n \times \mathcal{M}$ *such that for all* $\vec{v} \in \Sigma_n$ *and* $\mathsf{tok}_{\vec{v}} \in \{0,1\}^*$*, the following holds:*

$$\Pr\left[\mathsf{Dec}(\mathsf{MPK}, \mathsf{tok}_{\vec{v}}, \mathsf{CT}) = f_{\vec{v}}(\vec{x}, m) \;\middle|\; \begin{array}{l} \mathsf{VerifyMPK}(\mathsf{MPK}) = 1, \\ \mathsf{VerifyCT}(\mathsf{MPK}, \mathsf{CT}) = 1, \\ \mathsf{Verifytok}(\mathsf{MPK}, \vec{v}, \mathsf{tok}_{\vec{v}}) = 1 \end{array}\right] = 1$$

Intuitively verifiability states that each ciphertext (possibly with the malicious generated public key) should be associated with a unique message $(\vec{x}, m)$ and decryption for a function $f_{\vec{v}}$ using any possibly maliciously generated token $\mathsf{tok}_{\vec{v}}$ should result in $f_{\vec{v}}(\vec{x}, m)$ for the unique message associated with the ciphertext [23].

Additional Note. We remark that a verifiability property does hold only in a functional encryption system with the perfectly correct property. Because in case of (even) a negligible probability of error, a non-uniform adversary can choose the randomness (used in encryption or token generation algorithms) where the verification procedure falsely verifies the validity while the outcome of the algorithm does not match the actual values.

5.3.1 Security Notion of VFE

Similar to FE (See 4.2.1), two approaches capture the security notion for functional encryption schemes; Game-based indistinguishability and simulation-based security, both of which come with the flavor of adaptive versus non-adaptive, one versus many, and fully versus selectively concept.

Simulation-Based Security. In [23], they show the implausibility result in a simulation-based secure verifiable functional encryption scheme (where the adversary can request keys arbitrarily). Furthermore, they demonstrate that even the most basic simulation-based security is impossible to achieve for verifiable functional encryption. The definition of simulation-based security for verifiable functional encryptions scheme is the same as for FE; we refer to 4.1 for more details.

Theorem 5.3.1 ([23])**.** *There exists a family of functions, each of which can be represented as a polynomial sized circuit, for which there does not exist any simulation-secure verifiable functional encryption scheme.*

Indistinguishability-Based Security. We define a general version here. For a more detailed definition, we refer to Section 4.2.1:

Definition 36. *A verifiable functional encryption scheme* Π^{vfe} *is* {selective, fully}*-secure if all PPT adversary* $\mathcal{A}$ *has at most a negligible advantage in the experiment 5.1.*

FIGURE 5.1: IND-CPA Security Game $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ind-ip}}(1^{\ell}, n)$

- **Selective Challenge Phase.** $\mathcal{A}(1^{\ell}, n) \longrightarrow \vec{x}_0, \vec{x}_1 \in \Sigma_n$. Then $\mathcal{A}$ sends these two vectors to the challenger.
- **Setup Phase.** The challenger $\mathcal{C}$ generates the pair $(\mathsf{MSK}, \mathsf{MPK})$ by invoking the setup algorithm on input $(1^{\ell}, n)$. Then $\mathcal{C}$ sends MPK to $\mathcal{A}$.
- **Query Phase 1.** $\mathcal{A}$ asks for the token for a vector $\vec{v}_i \in \Sigma_n$.
- **Challenge Phase.** $\mathcal{A}$ sends to the challenger two messages $m_0, m_1 \in \mathcal{M}$ of the same length.
- **Challenge Phase.** $\mathcal{C}$ flips a coin to generate random bit b and send

$$\mathsf{CT} = \mathsf{VIP.Enc}(\mathsf{MPK}, \vec{x}_b, m_b).$$

- **Query Phase 2.** Query Phase 2: same as Query Phase 1.
- **Output Phase.** $\mathcal{A}$ outputs a bit b'.
- **Winning Condition.** $\mathcal{A}$ wins the game if $b' = b$ and the following condition is met. It is required that if $m_0 \neq m_1$, $\langle \vec{x}_0, \vec{v}_i \rangle, \langle \vec{x}_1, \vec{v}_i \rangle \neq 0$ for all the vectors $\vec{v}_i$ queried in both query phase 1 and 2, or $\langle \vec{v}_i, \vec{x}_0 \rangle = 0$ iff $\langle \vec{v}_i, \vec{x}_1 \rangle = 0$ otherwise. If the winning condition is satisfied, the game's output is 1 or 0 otherwise.

5.4 VIP Relations

Our VIPE is based on a perfectly correct IPE (cf. our IPE scheme of Construction 4.6), a perfectly binding commitment scheme such as the commitment scheme proposed in [141] and NIWI proofs for some specific relations that will be detailed below.

Let $n \in \mathbb{N}$ be the vector length and ℓ the security parameter. Let IP be a perfectly correct IPE scheme, Commit be a perfectly binding commitment scheme and

$$\mathrm{NIWI}^{\mathsf{mpk}} = \langle \mathcal{P}^{\mathsf{mpk}}, \mathcal{V}^{\mathsf{mpk}} \rangle, \mathrm{NIWI}^{\mathsf{enc}} = \langle \mathcal{P}^{\mathsf{enc}}, \mathcal{V}^{\mathsf{enc}} \rangle, \mathrm{NIWI}^{\mathsf{tok}} = \langle \mathcal{P}^{\mathsf{tok}}, \mathcal{V}^{\mathsf{tok}} \rangle$$

be NIWI proofs systems for, resp., the relations $\mathsf{R}^{\mathsf{mpk}}$, $\mathsf{R}^{\mathsf{enc}}$ and $\mathsf{R}^{\mathsf{tok}}$, that are essentially instantiations of analogous relations in [23]. The construction of these NIWI systems are provided in Section 5.6.

1.

$$\begin{aligned}\mathrm{R}^{\mathsf{IP}}_{\mathsf{mpk}} = \Big\{(x,w):\ & \\ & x = \mathsf{mpk}, \\ & w = (\mathsf{msk}, \mathsf{r}^{\mathsf{mpk}}), \\ & (\mathsf{mpk},\mathsf{msk}) = \mathsf{IP.SetUp}(1^{\ell}, n; \mathsf{r}^{\mathsf{mpk}}) \\ \Big\} &\end{aligned} \tag{5.2}$$

2.

$$\begin{aligned}\mathrm{R}^{\mathsf{IP}}_{\mathsf{tokk}} = \Big\{(x,w):\ & \\ & x = (\mathsf{mpk}, \mathsf{tok}, \vec{v}), \\ & w = (\mathsf{msk}, \mathsf{r}^{\mathsf{mpk}}, \mathsf{r}^{\mathsf{token}}), \\ & (\mathsf{mpk}, (\mathsf{msk}, \mathsf{r}^{\mathsf{mpk}})) \in \mathrm{R}^{\mathsf{mpk}}_{\mathsf{IP}}, \\ & \mathsf{tok} = \mathsf{IP.TokGen}(\mathsf{MSK}, \vec{v}; \mathsf{r}^{\mathsf{tok}}), \\ \Big\} &\end{aligned} \tag{5.3}$$

3.

$$\begin{aligned}\mathrm{R}^{\mathsf{IP}}_{\mathsf{k,ct}} = \Big\{(x,w):\ & \\ & x = \big((\mathsf{ct}_1, \mathsf{mpk}_1), \ldots, (\mathsf{ct}_k, \mathsf{mpk}_k)\big), \\ & w = \big(\vec{x}, m, \mathsf{r}^{\mathsf{enc}}_1, \ldots, \mathsf{r}^{\mathsf{enc}}_k\big), \\ & k \in [4], \\ & i \in [k]:\ \mathsf{ct}_i = \mathsf{IP.Enc}(\mathsf{mpk}_i, \vec{x}, m; \mathsf{r}^{\mathsf{enc}}_i) \\ \Big\} &\end{aligned} \tag{5.4}$$

4.

$$\begin{aligned}\mathrm{R}^{\mathsf{IP}}_{\mathsf{enc}} = \Big\{(x,w):\ & \\ & x = (\{c_i\}_{i\in[4]}, \{a_i\}_{i\in[4]}, z_0, z_1), \\ & w = (m, \vec{x}, \{\mathsf{r}^{\mathsf{enc}}_i\}_{i\in[4]}, i_1, i_2, \mathsf{r}^{\mathsf{com}}_0, \mathsf{r}^{\mathsf{com}}_1) \\ & \mathsf{P}^{\mathsf{enc}}_1(x,w) \vec{e} \mathsf{P}^{\mathsf{enc}}_2(x,w) \\ \Big\} & \\ \mathsf{P}^{\mathsf{enc}}_1 = \mathsf{True} \iff & \\ & \Big(\big((c_1,a_1),\ldots,(c_4,a_4)\big), (\vec{x}, m, \{\mathsf{r}^{\mathsf{enc}}_i\}_{i\in[4]})\Big) \in \mathrm{R}^{4,\mathsf{ct}}_{\mathsf{IP}} \\ \mathsf{P}^{\mathsf{enc}}_2(x,w) = \mathsf{True} \iff & \\ & i_1, i_2 \in [4] \wedge (i_1 \neq i_2) \wedge \\ & \Big(\big((c_{i_1}, a_{i_1}), (c_{i_2}, a_{i_2})\big), (\vec{x}, m, \mathsf{r}^{\mathsf{enc}}_i)\Big) \in \mathrm{R}^{2,\mathsf{ct}}_{\mathsf{IP}}, \\ & \wedge\ z_0 = \mathsf{Commit}(\{c_i\}_{i\in[4]}; \mathsf{r}^{\mathsf{com}}_0) \wedge\ z_1 = \mathsf{Commit}(0; \mathsf{r}^{\mathsf{com}}_1)\end{aligned} \tag{5.5}$$

5.

$$
\begin{aligned}
\mathsf{R}^{\mathsf{IP}}_{\mathsf{tok}} = \Big\{(x, w) :& \\
& x = (\vec{v}, \{t_i\}_{i\in[4]}, \{a_i\}_{i\in[4]}, z_0, z_1) \\
& w = (\{b_i\}_{i\in[4]}, \{\mathsf{r}^{\mathsf{mpk}}_i\}_{i\in[4]}, \{\mathsf{r}^{\mathsf{tok}}_i\}_{i\in[4]}, i_1, i_2, i_3, \mathsf{r}^{\mathsf{com}}_0, \mathsf{r}^{\mathsf{com}}_1) \\
\mathsf{P}^{\mathsf{tok}}_1 = \mathsf{True}& \\
& \forall i \in [4] : \big((a_i, (b_i, \mathsf{r}^{\mathsf{mpk}}_i)\big) \in \mathsf{R}^{\mathsf{mpk}} \wedge \\
& \big((a_i, t_i, \vec{v}_i), (b_i, \mathsf{r}^{\mathsf{mpk}}_i, \mathsf{r}^{\mathsf{tok}}_i))\big) \in \mathsf{R}^{\mathsf{tok}}_{\mathsf{IP}} \wedge \\
& z_1 = \mathsf{Commit}(1; \mathsf{r}^{\mathsf{com}}_1) \\
\mathsf{P}^{\mathsf{tok}}_2& \\
& i_1, i_2, i_3 \in [4] \wedge \\
& (i_1 \neq i_2) \wedge (i_1 \neq i_3) \wedge (i_2 \neq i_3) \\
& \forall j \in [3] : \big(a_{i_j}, (b_{i_j}, \mathsf{r}^{\mathsf{mpk}}_{i_j})\big) \in \mathsf{R}^{\mathsf{mpk}} \wedge \\
& \Big(\big(a_{i_j}, t_{i_j}, \vec{v}_{i_j}\big), \big(b_{i_j}, \mathsf{r}^{\mathsf{mpk}}_{i_j}, \mathsf{r}^{\mathsf{tok}}_{i_j}\big)\Big) \in \mathsf{R}^{\mathsf{tok}}_{\mathsf{IP}} \\
\wedge& \\
& z_0 = \mathsf{Commit}(\{c_i\}_{i\in[4]}; \mathsf{r}^{\mathsf{com}}_1) \wedge \\
& \exists m \in \mathcal{M}\ \forall i \in [4]\ \mathsf{IP.Dec}(c_i, t_i) = f_{\vec{v}}(m) \\
\Big\}&
\end{aligned}
\tag{5.6}
$$

5.5 Our Verifiable Inner Product Encryption Scheme

Considering the relation, we describe our verifiable inner product encryption scheme as follows:

1. $\mathsf{VIP.SetUp}(1^{\ell}, n) \to (\mathsf{MPK}, \mathsf{MSK})$:
   (a) For $i \in [4]$, run $\mathsf{IP.SetUp}(1^{\ell}, n)$ to generate $(\mathsf{MPK}_i, \mathsf{MSK}_i)$.
   (b) Run the commitment algorithm to generate

   $$\mathsf{Z}_0 = \mathsf{Commit}(0; \mathsf{r}^{\mathsf{com}}_0)\ ,\ \mathsf{Z}_1 = \mathsf{Commit}(1; \mathsf{r}^{\mathsf{com}}_1).$$

   (c) Output

   $$\mathsf{VIP.MPK} = (\{\mathsf{MPK}_i\}_{i\in[4]}, \mathsf{Z}_0, \mathsf{Z}_1),$$
   $$\mathsf{VIP.MSK} = (\{\mathsf{MSK}_i\}_{i\in[4]}, \mathsf{r}^{\mathsf{com}}_0, \mathsf{r}^{\mathsf{com}}_1)$$

2. $\mathsf{VIP.Enc}(\mathsf{MPK}, m, \vec{x}) \to \mathsf{CT}$:
   (a) For $i \in [4]$, run the encryption algorithm to compute:

   $$\mathsf{CT}_i = \mathsf{IP.Enc}(\mathsf{MPK}, m, \vec{x}; \mathsf{r}^{\mathsf{enc}}_i)$$

   .

(b) Set

$$x = (\{\mathsf{CT}_i\}_{i\in[4]}, \{\mathsf{MPK}_i\}_{i\in[4]}, \mathsf{Z}_0, \mathsf{Z}_1),$$
$$w = (m, \vec{x}, \{\mathsf{r}_i^{\mathsf{enc}}\}_{i\in[4]}, 0, 0, 0^{|u_0|}, 0^{|u_1|})$$

(c) Run $\mathcal{P}^{\mathsf{enc}}(x,w)$ to generate π_{ct} for relation $\mathrm{R}^{\mathsf{enc}}(x,w)$. Note that $\mathsf{P}_1^{\mathsf{enc}}(x,w) = \mathsf{True}$.

(d) Output ciphertext

$$\mathsf{CT} = (\{\mathsf{CT}_i\}_{i\in[4]}, \pi_{\mathsf{ct}}).$$

3. $\mathsf{VIP.TokGen}(\mathsf{MPK}, \mathsf{MSK}, f_{\vec{v}})$:

   (a) For $i \in [4]$, run $\mathsf{IP.TokGen}(\mathsf{MSK}, \vec{v}; \mathsf{r}_i^{\mathsf{tok}})$ to generate $\mathsf{tok}_{\vec{v}}^i$.

   (b) For

   $$x = (\vec{v}, \{\mathsf{tok}_{\vec{v}}^i\}_{i\in[4]}, \{\mathsf{MPK}_i\}_{i\in[4]}, \mathsf{Z}_0, \mathsf{Z}_1),$$

   and

   $$w = (\{\mathsf{MSK}_i\}_{i\in[4]}, \{\mathsf{r}_i^{\mathsf{tok}}\}_{i\in[4]}, 0, 0, 0, 0^{|\mathsf{r}_0^{\mathsf{com}}|}, |\mathsf{r}_1^{\mathsf{com}}|)$$

   run $\mathcal{P}^{\mathsf{tok}}$ to generate π_{tok} to prove $\mathrm{R}^{\mathsf{tok}}(x,w) = \mathsf{True}$. Note that $\mathsf{P}_1^{\mathsf{tok}}(x,w) = \mathsf{True}$.

   (c) Output token $\mathsf{tok}_{\vec{v}} = (\{\mathsf{tok}_{\vec{v}}^i\}_{i\in[4]}, \pi_{\mathsf{tok}})$.

4. $\mathsf{VIP.Dec}(\mathsf{MPK}, f_{\vec{v}}, \mathsf{tok}_{\vec{v}}, \mathsf{CT})$:

   (a) Run the verification algorithms $\mathcal{V}^{\mathsf{mpk}}, \mathcal{V}^{\mathsf{enc}}, \mathcal{V}^{\mathsf{tok}}$ on input the corresponding pairs of statements and proof (the proof for the verification of the master public key is set to the empty string). If some verification algorithms fail, stop and output $\perp$ or go to the next step otherwise.

   (b) For all $i \in [4]$, compute

   $$m^{(i)} = \mathsf{IP.Dec}(\mathsf{tok}_{\vec{v}}^{(i)}, \mathsf{CT}_i),$$

   and output the following:

   $$\begin{cases} \texttt{If } \exists i_1, i_2, i_3 \in [4] s.t. m = m^{(i_1)} = m^{(i_2)} = m^{(i_3)} \Rightarrow \texttt{Output } m. \\ \texttt{If } \nexists i_1, i_2, i_3 \in [4] s.t. m^{(i_1)} = m^{(i_2)} = m^{(i_3)} \Rightarrow \texttt{Output } \perp. \end{cases}$$

5. $\mathsf{VIP.VerifyMPK}(\mathsf{MPK})$: Run

   $$\mathcal{V}^{\mathsf{mpk}}(\mathsf{MPK}, \epsilon)$$

   and output its result.

6. $\mathsf{VIP.VerifyCT}\big((\{\mathsf{CT}_i\}_{i\in[4]}, \{\mathsf{MPK}_i\}_{i\in[4]}, \mathsf{Z}_0, \mathsf{Z}_1), \pi_{\mathsf{ct}})\big)$:
   run

   $$\mathcal{V}^{\mathsf{enc}}\big((\{\mathsf{CT}_i\}_{i\in[4]}, \{\mathsf{MPK}_i\}_{i\in[4]}, \mathsf{Z}_0, \mathsf{Z}_1), \pi_{\mathsf{ct}}\big)$$

   and output its result.

7. $\mathsf{VIP.Verifytok}\big((\vec{v}, \{\mathsf{tok}_{\vec{v}}^i\}_{i\in[4]}, \{\mathsf{MPK}_i\}_{i\in[4]}, \mathsf{Z}_0, \mathsf{Z}_1), \pi_{\mathsf{tok}}\big)$:
   run

   $$\mathcal{V}^{\mathsf{tok}}\big((\vec{v}, \{\mathsf{tok}_{\vec{v}}^i\}_{i\in[4]}, \{\mathsf{MPK}_i\}_{i\in[4]}, \mathsf{Z}_0, \mathsf{Z}_1), \pi_{\mathsf{tok}}\big)$$

   and output its result.

Correctness of VIP follows from perfect correctness of IP. IND-Security and Verifiability of VIP follows as a corollary (following theorem 5.5.1) from the verifiability and IND-Security of the construction of [23] for general functions.

Theorem 5.5.1. *If* IP *is a perfectly correct IND-secure IP scheme for message space $\mathcal{M}$ and the set $\mathbb{Z}_p^n$ of vectors of length n over $\mathbb{Z}_p$, and NIWI$^{\mathsf{mpk}}$, NIWI$^{\mathsf{ct}}$, NIWI$^{\mathsf{tok}}$ are NIWI systems resp. for relations $R^{\mathsf{mpk}}, R^{\mathsf{enc}}, R^{\mathsf{tok}}$ and* Commit *is a non-interactive perfectly binding and computationally hiding commitment scheme, then* VIP *is an IND-secure VIPE scheme for the class of inner product functionality over $\mathcal{M}$ and $\mathbb{Z}_p^n$.*

5.6 NIWI Proofs and Verification Algorithms

This section presents the proof systems that we used in our VIP scheme to prove membership of relations R^{mpk}, R^{tok} and R^{enc}. First for each of our relations[2], we need to define a system of equations such that the satisfiability of that system and the membership in the relation are equivalent. Then, the GS generic prover and verifier algorithms, $\text{NIWI}_{\mathbb{GS}} = \langle \mathcal{P}_{\mathbb{GS}}, \mathcal{V}_{\mathbb{GS}} \rangle$, can be used for such equations. In this section, for each of our relations of Section 5.4, we will either define a corresponding system of equations or show how to implement directly (without using GS proofs).

Notations: For the rest of this section, let us fix $n \in \mathbb{N}$ as a vector space dimension and let $i \in [n], b \in [2]$. Note we can efficiently check whether a string is a valid group element. We recall what follows.

$$\begin{aligned}
&\mathsf{mpk} = \big(g, h, \{W_{b,i}, F_{b,i}, T_{b,i}, H_{b,i}, U_b, V_b\}, K_1, K_2, \Lambda\big) \in \mathbb{G}^{4n+8} \times \mathbb{G}_T \\
&\mathsf{msk} = \big(\{w_{b,i}, f_{b,i}, t_{b,i}, h_{b,i}, \delta_b, \theta_b\}, \Omega, k\big) \in \mathbb{Z}_p^{4n+6} \\
&\mathsf{tok} = (K_A, K_B, \{K_{3,i}, K_{4,i}, K_{5,i}, K_{6,i}\}_i) \in \mathbb{G}^{4n+2} \\
&\mathsf{ct} = \Bigg((\mathsf{ct}_1, \mathsf{ct}_2, \begin{Bmatrix} \mathsf{ct}_{3,i} & , & \mathsf{ct}_{4,i} \\ \mathsf{ct}_{5,i} & , & \mathsf{ct}_{6,i} \end{Bmatrix}, \mathsf{ct}_7, \mathsf{ct}_8), \\
&\qquad (\mathsf{ct}'_1, \mathsf{ct}'_2, \begin{Bmatrix} \mathsf{ct}'_{3,i} & , & \mathsf{ct}'_{4,i} \\ \mathsf{ct}'_{5,i} & , & \mathsf{ct}'_{6,i} \end{Bmatrix}, \mathsf{ct}'_7, \mathsf{ct}'_8) \Bigg) \in \mathbb{G}^{8n+6} \times \mathbb{G}_T^2
\end{aligned}$$

5.6.1 Master Public Key Verification

Let $x = \mathsf{mpk}$. Since g and $\mathbf{e}(g,g)$ are generators for the groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p, we can represent all components of x as a power of either g or $\mathbf{e}(g,g)$. That is, there exist:

$$i = 1, \ldots, n,\ b = 1, 2 : \Omega, k', w_{b,i}, f_{b,i}, t_{b,i}, h_{b,i}, \{\delta_b, \theta_b, k_b\} \in \mathbb{Z}_p$$

such that:

$$h = g^{\Omega}, W_{b,i} = g^{w_{b,i}}, F_{b,i} = g^{f_{b,i}}, U_b = g^{\delta_b}, K_b = g^{k_b}$$
$$\Lambda = \mathbf{e}(g,g)^{k'}, T_{b,i} = g^{t_{b,i}}, H_{b,i} = g^{h_{b,i}}, V_b = g^{\theta_b}$$

[2]Actually, we will implement some or part of them not directly using GS proofs.

```
Input: mpk,
Output: 1 if mpk is a well-generated master public key for IP scheme and 0 otherwise

    (1) If $\Lambda \neq \mathbf{e}(K_1, K_2)$. output 0 otherwise go to the next step
    (2) For $i = 1$ to $n$ do :
        $(i.a)$ If $\mathbf{e}(U_1, W_{2,i}) \cdot \mathbf{e}(U_2, W_{1,i})^{-1} \neq \mathbf{e}(h, g)$ output 0 else go to the next step
        $(i.b)$ If $\mathbf{e}(V_1, T_{2,i}) \cdot \mathbf{e}(V_2, T_{1,i})^{-1} \neq \mathbf{e}(h, g)$ output 0 else go to the next step
    (3) Output 1.
```

FIGURE 5.2: Master public key verification algorithm. (R_{IP}^{mpk}'s membership)

The following holds:

$$\begin{aligned}
\mathbf{e}(g,h) &= \mathbf{e}(U_1, W_{2,i}) \cdot \mathbf{e}(U_2, W_{1,i})^{-1} \\
&= \mathbf{e}(V_1, T_{2,i}) \cdot \mathbf{e}(V_2, T_{1,i})^{-1} \\
&\Rightarrow \\
\mathbf{e}(g, g^{\Omega}) &= \mathbf{e}(g^{\delta_1}, g^{w_{2,i}}) \cdot \mathbf{e}(g^{\delta_2}, g^{-w_{1,i}}) \\
&= \mathbf{e}(g^{\theta_1}, g^{t_{2,i}}) \cdot \mathbf{e}(g^{\theta_2}, g^{-t_{1,i}}) \\
&\Rightarrow \\
\Omega &= \delta_1 w_{2,i} - \delta_2 w_{1,i} \\
&= \theta_1 t_{2,i} - \theta_2 t_{1,i}. \\
\mathbf{e}(K_1, K_2) &= \mathbf{e}(g^{k_1}, g^{k_2}) \\
&= \Lambda = \mathbf{e}(g, g^{k'}) \\
&\Rightarrow k' = k_1 k_2
\end{aligned}$$

By defining $g' := g^{k'}, K_1 := g^{k_1}, K_2 := g^{k_2}$, it follows that: $\Lambda = \mathbf{e}(K_1, K_2), K_1 = g^{k}, K_2 = g'^{\frac{1}{k}}$

Hence, we have the verification algorithm in Fig. 5.2 for the master public key:

5.6.2 Token Verification Algorithm

As defined in Section 5.4, there are two relations for tokens, R_{IP}^{tok} and R^{tok}. The algorithm in Fig. 5.3 verifies membership in relation R_{IP}^{tok}.

Correctness of the algorithm: For simplicity, let's assume $v_1 \neq 0$ and $i^* = 1$.

- $\Lambda_1^*, \Lambda_2^* \in \mathbb{G}_T \Rightarrow \exists \lambda_1, \lambda_2 \in \mathbb{Z}_p s.t. : \begin{cases} \Lambda_1^* = e(g,h)^{\lambda_1 v_1}, \\ \Lambda_2^* = e(g,h)^{\lambda_2 v_1} \end{cases}$
- $\forall i \in \{1, 2, \ldots, n\} \, \exists r_i, r_i' \in \mathbb{Z}_p \; s.t. \begin{cases} K_{3,i} = g^{-\delta_2 r_i} \cdot g^{\lambda_1 v_i w_{2,i}}, \\ K_{4,i} = g^{\delta_1 r_i'} \cdot g^{-\lambda_1 v_i w_{1,i}} \end{cases}$

Input: MPK, $\overrightarrow{v} = (v_1, \dots, v_n) \neq \overrightarrow{0}$, tok
Output: 1 if tok is a well-generated token for IP scheme and 0 otherwise

```
1. If $\overrightarrow{v} = \overrightarrow{0}$ output 0 else let $i^*$ be an index such that $v_{i^*} \neq 0$
2. Compute $\Lambda_1^* = \mathbf{e}(K_{3,i}, U_1) \cdot \mathbf{e}(K_{4,i}, U_2)$ and $\Lambda_2^* = \mathbf{e}(K_{5,i}, V_1) \cdot \mathbf{e}(K_{6,i}, V_2)$
3. If $\Lambda_1^* = 1_{\mathbb{G}_T}$ OR $\Lambda_2^* = 1_{\mathbb{G}_T}$ output $\perp$
4. For $i = 1$ to $n$ do:
    (a) If $\left(\mathbf{e}(K_{3,i}, U_1) \cdot \mathbf{e}(K_{4,i}, U_2)\right)^{v_{i^*}} \neq (\Lambda_1^*)^{v_i}$ output 0
    (b) If $\left(\mathbf{e}(K_{5,i}, V_1) \cdot \mathbf{e}(K_{6,i}, V_2)\right)^{v_{i^*}} \neq (\Lambda_2^*)^{v_i}$ output 0
5. If $\Lambda \prod_{i=1}^n \mathbf{e}(K_{3,i}, F_{1,i})^{-1} \cdot \mathbf{e}(K_{4,i}, F_{2,i})^{-1} \cdot \mathbf{e}(K_{5,i}, H_{1,i})^{-1} \mathbf{e}(K_{6,i}, H_{2,i})^{-1} \neq \mathbf{e}(K_A, g)$ output 0.
6. If $\prod_{i=1}^n \mathbf{e}(K_{3,i}, W_{1,i}) \cdot \mathbf{e}(K_{4,i}, W_{2,i}) \cdot \mathbf{e}(K_{5,i}, T_{1,i}) \cdot \mathbf{e}(K_{6,i}, T_{2,i}) \neq \mathbf{e}(h, K_B)^{-1}$ output 0.
7. Output 1.
```

FIGURE 5.3: Token verification algorithm. ($\mathsf{R}_{\mathsf{IP}}^{\mathsf{tok}}$'s membership)

Hence we obtain:

$$\begin{aligned}
\mathbf{e}(K_{3,i}, U_1) \cdot \mathbf{e}(K_{4,i}, U_2) &= \mathbf{e}(g^{-\delta_2 r_i} \cdot g^{\lambda_1 v_i w_{2,i}}, g^{\delta_1}) \cdot \mathbf{e}(g^{\delta_1 r_i'} \cdot g^{-\lambda_1 v_i w_{1,i}}, g^{\delta_2}) \\
&= \mathbf{e}(g, g)^{\delta_1 \delta_2 (r_i' - r_i)} \cdot \mathbf{e}(g, h)^{\lambda_1 v_i} \\
&\Rightarrow \\
\left(\mathbf{e}(K_{3,i}, U_1) \cdot \mathbf{e}(K_{4,i}, U_2)\right)^{v_1} &= \mathbf{e}(g, g)^{v_1 \delta_1 \delta_2 (r_i' - r_i)} \cdot \mathbf{e}(g, h)^{\lambda_1 v_1 v_i}
\end{aligned} \tag{5.7}$$

- **Step 3:** $\Lambda_1^* \neq 1_{\mathbb{G}_T}, \Lambda_2^* \neq 1_{\mathbb{G}_T} \Rightarrow \lambda_1 \neq 0, \lambda_2 \neq 0$
- **Step 4.a:** If $\left(\mathbf{e}(K_{3,i}, U_1) \cdot \mathbf{e}(K_{4,i}, U_2)\right)^{v_1} = (\Lambda_1^*)^{v_i} \Rightarrow$

$$\begin{aligned}
&\mathbf{e}(g, g)^{v_1 \delta_1 \delta_2 (r_i' - r_i)} \cdot \mathbf{e}(h, g)^{\lambda_1 v_1 v_i} = \mathbf{e}(g, h)^{\lambda_1 v_1 v_i} \\
&\Rightarrow \\
&\mathbf{e}(g, g)^{v_1 \delta_1 \delta_2 (r_i' - r_i)} = 1_{\mathbb{G}_T} \\
&\Rightarrow \\
\forall i \in [n] :\ r_i = r_i' \Rightarrow &\begin{cases} K_{3,i} = g^{-\delta_2 r_i} \cdot g^{\lambda_1 v_i w_{2,i}}, \\ K_{4,i} = g^{\delta_1 r_i} \cdot g^{-\lambda_1 v_i w_{1,i}} \end{cases}
\end{aligned}$$

And similar computations show that the equality in step (4.b) holds for all $i \in [n]$. Then we conclude that there exists $\phi_i \in \mathbb{Z}_p$ such that:

$$K_{5,i} = g^{-\theta_2 \phi_i} \cdot g^{\lambda_2 v_i t_{2,i}}, K_{6,i} = g^{\theta_1 \phi_i} \cdot g^{-\lambda_2 v_i t_{1,i}}.$$

- **Step 5**

$$K_A = g' \prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} K_{4,i}^{-f_{2,i}} K_{5,i}^{-h_{1,i}} K_{6,i}^{-h_{2,i}} \iff$$
$$\iff \mathbf{e}(K_A, g) = \mathbf{e}(g' \prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} K_{4,i}^{-f_{2,i}} K_{5,i}^{-h_{1,i}} K_{6,i}^{-h_{2,i}}, g)$$
$$\iff \mathbf{e}(K_A, g) = \Lambda \cdot \prod_{i=1}^{n} \mathbf{e}(K_{3,i}, F_{1,i})^{-1} . \mathbf{e}(K_{4,i}, F_{2,i})^{-1} . \mathbf{e}(K_{5,i}, H_{1,i})^{-1} \cdot \mathbf{e}(K_{6,i}, H_{2,i})^{-1}.$$

- **Step 6**

$$\prod_{i=1}^{n} \mathbf{e}(K_{3,i}, W_{1,i}) \cdot \mathbf{e}(K_{4,i}, W_{2,i}) \cdot \mathbf{e}(K_{5,i}, T_{1,i}) \cdot \mathbf{e}(K_{6,i}, T_{2,i}) = \mathbf{e}(h, K_B)^{-1}$$
$$= \prod_{i=1}^{n} \mathbf{e}(g^{r_i(\delta_1 w_{2,i} - \delta_2 w_{1,i})}, g) \cdot \mathbf{e}(g^{\phi_i(\theta_1 t_{2,i} - \theta_2 t_{1,i})}, g) = \mathbf{e}(h, K_B)^{-1}$$
$$= \prod_{i=1}^{n} \mathbf{e}(g, h)^{r_i + \phi_i} = \mathbf{e}(h, K_B)^{-1} \Rightarrow K_B = \prod_{i=1}^{n} g^{-(r_i + \phi_i)}$$

The second relation is a disjunction of two predicates, $\mathrm{R}^{\mathsf{tok}}(x, w) = P_1^{\mathsf{tok}} \vee P_2^{\mathsf{tok}}$. The proof of membership for this relation can be implemented using the equations for the token verification algorithm for relation $\mathrm{R}_{\mathsf{IP}}^{\mathsf{tok}}$ 5.3 and assuming to have pairing product equations corresponding to the commitments in the two predicates above. We skip further details.

5.6.3 NIWI-Proof for Encryption Algorithm

For the relation $\mathrm{R}_{\mathsf{IP}}^{\mathsf{ct}}$, we first provide proof of satisfiability for a system of equations related to a single ciphertext, that is $k = 1$. We will later extend it to the case of two ciphertexts, that is $k = 2$. For $k > 2$, the algorithm is similar to the case $k = 2$.

Let $x = (\mathsf{mpk}, \mathsf{ct})$. We define the following variables for $i \in [n]$:

$$\mathcal{S}_1 = g^{s_1}, \mathcal{S}_3 = g^{s_3}, \mathcal{S}_4 = g^{s_4}, \mathcal{X}_i = g^{x_i}, \mathcal{S}_1' = g^{s_1'}, \mathcal{S}_3' = g^{s_3'}, \mathcal{U}_1 = U_1^{s_3},$$
$$\mathcal{U}_2 = U_2^{s_3}, \mathcal{V}_1 = V_1^{s_4}, \mathcal{V}_2 = V_2^{s_4}, \mathcal{U}_1' = U_1^{s_3'}, \mathcal{U}_2' = U_2^{s_3'}, \mathcal{K}_1 = K_1^{s_2}, \mathcal{K}_1' = K_1^{s_2'}$$

We have the following Equations related to component $\mathsf{ct}_2 (\mathsf{ct}_2')$:

$$\mathbf{e}(\mathsf{ct}_2, g) = \mathbf{e}(h^{s_1}, g) = \mathbf{e}(h, g^{s_1}) = \mathbf{e}(h, \mathcal{S}_1), \left(\mathbf{e}(\mathsf{ct}_2', g) = \mathbf{e}(h, \mathcal{S}_1')\right)$$

and related equation to $\mathsf{ct}_{3,i}$ for $i \in [n]$: (Same computation results the same equations for $\mathsf{ct}_{j,i}, \mathsf{ct}_{j,i}'$ for $j = 3, 4, 5, 6$)

$$\begin{aligned} \mathbf{e}(\mathsf{ct}_{3,i}, g) &= \mathbf{e}(W_{1,i}^{s_1}, g) \cdot \mathbf{e}(F_{1,i}^{s_2}, g) \cdot \mathbf{e}(U_1^{s_3 x_i}, g) \\ &= \mathbf{e}(W_{1,i}, g^{s_1}) \cdot \mathbf{e}(F_{1,i}, g^{s_2}) \cdot \mathbf{e}(U_1^{s_3}, g^{x_i}) \\ &= \mathbf{e}(W_{1,i}, \mathcal{S}_1) \cdot \mathbf{e}(F_{1,i}, \mathsf{ct}_1) \cdot \mathbf{e}(\mathcal{U}_1, \mathcal{X}_i) \\ \Rightarrow \mathbf{e}(\mathsf{ct}_{3,i}, g) \cdot \mathbf{e}(F_{1,i}, \mathsf{ct}_1)^{-1} &= \mathbf{e}(W_{1,i}, \mathcal{S}_1) \cdot \mathbf{e}(\mathcal{U}_1, \mathcal{X}_i) \end{aligned}$$

The equations show that the exponent of $U_b^{s_3}$ and $V_b^{s_4}$ in $\mathsf{ct}_{3,i}, \mathsf{ct}_{4,i}, \mathsf{ct}_{5,i}, \mathsf{ct}_{6,i}$ are x_i. So we have the following equation:

$$\begin{aligned}
\mathbf{e}(\mathcal{U}_1, U_2) \cdot \mathbf{e}(U_1^{-1}, \mathcal{U}_2) &= \mathbf{e}(U^{s_3}, U_2) \cdot \mathbf{e}(U_1^{-1}, U_2^{s_3}) \\
&= \mathbf{e}(U_1, U_2)^{s_3 - s_3} \\
&= 1_{\mathbb{G}_T} \\
\mathbf{e}(\mathcal{V}_1, V_2) \cdot \mathbf{e}(V_1^{-1}, \mathcal{V}_2) &= \mathbf{e}(V^{s_4}, V_2) \cdot \mathbf{e}(V_1^{-1}, V_2^{s_4}) \\
&= \mathbf{e}(V_1, V_2)^{s_4 - s_4} \\
&= 1_{\mathbb{G}_T}
\end{aligned}$$

The equation related to $\mathsf{ct}_7 = \mathbf{e}(g^{s_3}, g^{s_4})$ is the following:

$$\begin{aligned}
\mathsf{ct}_7 &= \mathbf{e}(g^{s_3}, g^{s_4}) = \mathbf{e}(\mathcal{S}_3, \mathcal{S}_4), \\
\mathsf{ct}_7' &= \mathbf{e}(g^{s_3'}, g^{s_4}) = \mathbf{e}(\mathcal{S}_3', \mathcal{S}_4)
\end{aligned}$$

To prove $s_3 \neq s_3'$, we just need to check whether $\mathsf{ct}_7 \neq \mathsf{ct}_7'$ or not.

$$\mathsf{ct}_7 \neq \mathsf{ct}_7' \Rightarrow \mathbf{e}(g^{s_3}, g^{s_4}) \neq \mathbf{e}(g^{s_3'}, g^{s_4}) \Rightarrow s_3 \neq s_3'.$$

The equation related to $\mathsf{ct}_8, \mathsf{ct}_8'$ is the following:

$$\begin{aligned}
&\mathsf{ct}_8 = \Lambda^{-s_2} \cdot m, \mathsf{ct}_8' = \Lambda^{-s_2'} \cdot m \\
\Rightarrow &\mathsf{ct}_8^{-1} \cdot \mathsf{ct}_8' = \Lambda^{s_2} \cdot m^{-1} \Lambda^{-s_2'} \cdot m \\
&= \Lambda^{s_2 - s_2'} \\
\Rightarrow &\mathsf{ct}_8^{-1} \cdot \mathsf{ct}_8' = \mathbf{e}(K_1, K_2)^{s_2 - s_2'} = \mathbf{e}(K_1, K_2^{s_2}) \cdot \mathbf{e}(K_1^{-1}, K_2^{s_2'}) = \\
&\mathbf{e}(K_1, \mathcal{K}_2) \cdot \mathbf{e}(K_1^{-1}, \mathcal{K}_1')
\end{aligned}$$

And to prove that $\mathsf{ct}_1 = g^{s_2}$ and $\mathsf{ct}_8 = \lambda^{-s_2} \cdot m$, we add the following equation:

$$\begin{aligned}
\mathbf{e}(\mathsf{ct}_1, K_1) &= \mathbf{e}(g, \mathcal{K}_1), \\
\mathbf{e}(\mathsf{ct}_1', K_1) &= \mathbf{e}(g, \mathcal{K}_1')
\end{aligned}$$

So we have the following system of equations for one single ciphertext.

Now we need to provide proof that two ciphertexts $\mathsf{ct}, \hat{\mathsf{ct}}$ are the encryption of a single message m and a single attribute $\overrightarrow{x}$:

$$\begin{aligned}
\mathcal{X}_i = g^{x_i}, \hat{\mathcal{X}}_i = \hat{g}^{x_i} \Rightarrow& \\
&\mathbf{e}(\mathcal{X}_i, \hat{g}) = \mathbf{e}(g, \hat{\mathcal{X}}_i) \\
&\Rightarrow \mathbf{e}(\mathcal{X}_i, \hat{g}) \cdot \mathbf{e}(g, \hat{\mathcal{X}}_i)^{-1} = 1_{\mathbb{G}_T}
\end{aligned}$$

FIGURE 5.4: E_{ct} : Equation for encryption verification algorithm.

$$
\begin{aligned}
&\mathbf{e}(\mathsf{ct}_2, g) = \mathbf{e}(h, \mathcal{S}_1),\\
&\mathbf{e}(\mathsf{ct}'_2, g) = \mathbf{e}(h, \mathcal{S}'_1)\\
&\mathbf{e}(\hat{\mathsf{ct}}_2, \hat{g}) = \mathbf{e}(\hat{h}, \hat{\mathcal{S}}_1),\\
&\mathbf{e}(\hat{\mathsf{ct}}'_2, \hat{g}) = \mathbf{e}(\hat{h}, \hat{\mathcal{S}}'_1)\\
&\mathbf{e}(\mathsf{ct}_{3,i}, g) \cdot \mathbf{e}(F_{1,i}, \mathsf{ct}_1)^{-1} = \mathbf{e}(W_{1,i}, \mathcal{S}_1) \cdot \mathbf{e}(\mathcal{U}_1, \mathcal{X}_i)\\
&\mathbf{e}(\mathsf{ct}'_{3,i}, g) \cdot \mathbf{e}(F_{1,i}, \mathsf{ct}'_1)^{-1} = \mathbf{e}(W_{1,i}, \mathcal{S}'_1) \cdot \mathbf{e}(\mathcal{U}'_1, \mathcal{X}_i)\\
&\mathbf{e}(\mathsf{ct}_{4,i}, g) \cdot \mathbf{e}(F_{2,i}, \mathsf{ct}_1)^{-1} = \mathbf{e}(W_{2,i}, \mathcal{S}_1) \cdot \mathbf{e}(\mathcal{U}_2, \mathcal{X}_i)\\
&\mathbf{e}(\mathsf{ct}'_{4,i}, g) \cdot \mathbf{e}(F_{2,i}, \mathsf{ct}'_1)^{-1} = \mathbf{e}(W_{2,i}, \mathcal{S}'_1) \cdot \mathbf{e}(\mathcal{U}'_2, \mathcal{X}_i)\\
&\mathbf{e}(\mathsf{ct}_{5,i}, g) \cdot \mathbf{e}(H_{1,i}, \mathsf{ct}_2)^{-1} = \mathbf{e}(T_{1,i}, \mathcal{S}_1) \cdot \mathbf{e}(\mathcal{V}_1, \mathcal{X}_i)\\
&\mathbf{e}(\mathsf{ct}'_{5,i}, g) \cdot \mathbf{e}(H_{1,i}, \mathsf{ct}'_2)^{-1} = \mathbf{e}(T_{1,i}, \mathcal{S}'_1) \cdot \mathbf{e}(\mathcal{V}_1, \mathcal{X}_i)\\
&\mathbf{e}(\mathsf{ct}_{6,i}, g) \cdot \mathbf{e}(H_{2,i}, \mathsf{ct}_2)^{-1} = \mathbf{e}(T_{2,i}, \mathcal{S}_1) \cdot \mathbf{e}(\mathcal{V}_2, \mathcal{X}_i)\\
&\mathbf{e}(\mathsf{ct}'_{6,i}, g) \cdot \mathbf{e}(H_{2,i}, \mathsf{ct}'_2)^{-1} = \mathbf{e}(T_{2,i}, \mathcal{S}'_1) \cdot \mathbf{e}(\mathcal{V}_2, \mathcal{X}_i)\\
&\mathsf{ct}_7 = \mathbf{e}(\mathcal{S}_3, \mathcal{S}_4), \mathsf{ct}'_7 = \mathbf{e}(\mathcal{S}'_3, \mathcal{S}_4),\\
&\hat{\mathsf{ct}}_7 = \mathbf{e}(\hat{\mathcal{S}}_3, \hat{\mathcal{S}}_4), \hat{\mathsf{ct}}'_7 = \mathbf{e}(\hat{\mathcal{S}}'_3, \hat{\mathcal{S}}_4)\\
&\mathsf{ct}_8^{-1} \cdot \mathsf{ct}'_8 = \mathbf{e}(K_1, \mathcal{K}_2) \cdot \mathbf{e}(K_1^{-1}, \mathcal{K}'_1),\\
&\hat{\mathsf{ct}}_8^{-1} \cdot \hat{\mathsf{ct}}'_8 = \mathbf{e}(\hat{K}_1, \hat{\mathcal{K}}_2) \cdot \mathbf{e}(\hat{K}_1^{-1}, \hat{\mathcal{K}}'_1)\\
&\mathbf{e}(\mathsf{ct}_1, K_1) = \mathbf{e}(g, \mathcal{K}_1),\\
&\mathbf{e}(\mathsf{ct}'_1, K_1) = \mathbf{e}(g, \mathcal{K}'_1)
\end{aligned}
$$

Notice that $\mathsf{ct}_8, \mathsf{ct}'_8$ are the only components of the ciphertext which are related to the message, m,so we have:

$$\begin{aligned}
\left(\mathsf{ct}_8 = \Lambda^{-s_2} m, \hat{\mathsf{ct}}_8 = \hat{\Lambda}^{-\hat{s}_2} m\right) &\Rightarrow \mathsf{ct}_8 \hat{\mathsf{ct}}_8^{-1} = \Lambda^{-s_2} \cdot \hat{\Lambda}^{\hat{s}_2} \\
\mathbf{e}(K_1^{s_2}, K_2^{-1}) \cdot \mathbf{e}(\hat{K}_1^{\hat{s}_2}, \hat{K}_2) &= \mathbf{e}(\mathcal{K}_1, K_2^{-1}) \cdot \mathbf{e}(\hat{\mathcal{K}}_1, \hat{K}_2) \\
&= \mathbf{e}(K_1^{-1}, \mathcal{K}_2) \cdot \mathbf{e}(\hat{K}_1, \hat{\mathcal{K}}_2) \\
&= \Lambda^{-s_2} \cdot \hat{\Lambda}^{\hat{s}_2} \\
&= \mathsf{ct}_8 \hat{\mathsf{ct}}_8^{-1}
\end{aligned}$$

So, the prover has to provide proof for the following system of equations:

$$\mathsf{E}_{\mathsf{ct}-\hat{\mathsf{ct}}} : \begin{cases} \mathsf{ct}_8 \hat{\mathsf{ct}}_8^{-1} = \mathbf{e}(\mathcal{K}_1, K_2^{-1},) \cdot \mathbf{e}(\hat{\mathcal{K}}_1, \hat{K}_2) \\ \mathsf{ct}_8 \hat{\mathsf{ct}}_8^{-1} = \mathbf{e}(K_1^{-1}, \mathcal{K}_2) \cdot \mathbf{e}(\hat{K}_1, \hat{\mathcal{K}}_2) \\ \mathbf{e}(g, \mathcal{K}_1) = \mathbf{e}(\mathsf{ct}_1, K_1) \\ \mathbf{e}(\hat{g}, \hat{\mathcal{K}}_1) = \mathbf{e}(\hat{\mathsf{ct}}_1, \hat{K}_1) \\ \mathbf{e}(\mathcal{X}_i, \hat{g}) \cdot \mathbf{e}(g, \hat{\mathcal{X}}_i)^{-1} = 1_{\mathbb{G}_T} \end{cases}$$

Summing up, to provide the NIWI-proof system for the encryption algorithm, the prover uses Groth-Sahai proof-system for the system of equations, $\mathsf{E}_{\mathsf{CT}} = \mathsf{E}_{\mathsf{ct}} \wedge \mathsf{E}_{\mathsf{ct}-\hat{\mathsf{ct}}}$.

5.7 Conclusion

Our main contribution is the first *efficient* verifiable (attribute-hiding) IPE scheme from bilinear groups. The privacy of our scheme is based on the standard DLin assumption whereas its verifiability is unconditional. Towards this goal, we also constructed the first perfectly correct inner product encryption scheme for plaintexts of arbitrary length. Our VIPE scheme is selectively secure only; we leave as an interesting open problem the construction of a fully secure one.

Part II

Verifiable Secure E-Voting Protocols

Developing an electronic voting system with all of the desired features is undoubtedly challenging, if not impossible; secrecy and privacy on one side, verifiability and coercion resistance on the other, and, most importantly, usability and efficiency, a complicated challenge.
That is correct. Democracy is not achieved for a small price, even in the electronic world!
In the second part of my thesis, I present my research on secure and verifiable electronic voting protocols. I have mainly worked on instantiating e-voting protocols with cryptographic primitive from the first part, providing a security analysis of the system while always keeping eyes on the efficiency and the voter-friendly property of the protocol.

Chapter 6

Building Blocks; Verifiable E-Voting Protocols

An electronic voting protocol (e-voting for short) can be considered a distributed system. Extensive research has been conducted to investigate the properties and requirements of e-voting protocols both from a technical [46, 85, 92, 241, 221] and legal point of view [254, 187, 208, 99].
In this chapter we give an overview of an e-voting protocol and its requirements. Later we, formally present the computational framework and formal definition as we will use later in our research.

Contents

6.1 Introduction

In order to be able to analyze and evaluate an e-voting scheme using accurate mathematical tools and concepts, we must first develop a solid foundation for an e-voting system, then describe all of the related concepts and properties, establish a robust framework, and choose an appropriate computing model. To begin to establish such a foundation, we consider an electronic voting system to take the form of distributed cryptographic protocols. These protocols make use of cryptographic primitives (e.g. encryption, signature) to provide privacy, protect voters against malicious parties.

As any distributed protocol, specifically an e-voting protocol, is operated by a group of participants and follows some specified algorithms. We begin with a generic description of e-voting protocol, including protocol's participants, roles, and phases.

6.1.1 Protocol Participants and Procedures

An e-voting protocol, in its broadest sense, consists of the setup and registration phase, the ballot-casting ceremony, the tally phase, and the verification phase, all of which are carried out by the entities listed below. We should highlight that any e-voting system can either put a variety of roles on a single party or completely abolish a role.

Protocol Participants.

- *Election Authority* is a party in charge of running the election, declaring election policies, and appointing other parties.
- *Election Trustee* produces and publishes public and private key pairs for the election. This is a simplified version of the real-world protocol that all of its keys are generated distributively among trustees.
- *Voters* who intend to vote may use a Voter Supporting Device (vsd) such as a desktop computer or smartphone. vsd is a device that plays a role as an interface between the voter and the voting server. It generates voters' ballots, casts the ballot on the bulletin board, and could also perform the verification step. We let $\{\mathsf{v}_1, \ldots, \mathsf{v}_n\}$ refer to the set of legitimate voters; By a legitimate voter, we mean the voter who is registered legally in the registration phase.
- *A voting server* that plays the role of a ballot box and collects the ballots from all voters.
- A set of *tabulation tellers* are in charge of the tallying process. They collect the ballots, tally them and output the election result (see below).
- *Bulletin Board*, $\mathfrak{BB}$ operates as a public append-only broadcast channel to distribute all election-related materials, such as public parameters, cast ballots, and tally phase information. Some e-voting schemes consider a set of ballot boxes to which voters cast their votes and a bulletin board that publishes the public information. In our work, the bulletin board is in charge of both tasks. An honest bulletin board keeps a list of all the input it receives from arbitrary participants and returns the list upon request. In a standard model of e-voting scheme, we assume a single trustworthy bulletin board. An honest bulletin board keeps a record of all the input it receives from random participants and distributes it on demand.

As we will see, most models (as well as many protocols) assume a single reliable bulletin board.

- A set of *supervised registrars* (SR) are in charge of administrating the electoral register. They register legitimate voters and provide them with their credentials.
- *Auditors/judges* verify particular information at the end or throughout the election in order to detect malicious behaviour. Typically, these checks are based simply on publicly available data and so can be performed by any party.
- The *coercer* is a party that maliciously attempts to force some parties to deviate their own choice. The coercer is not formally a participant in e-voting protocols, however, implicitly we consider its role which can be carried out by any participant of the protocol or some external party.

Protocol Phases: The following is a brief description of a conventional e-voting protocol.

- *Setup phase:* In this phase, the election authority members declare the election's policy and trustees. Election policy establishes procedures for re-voting and all the election parameters such as the election identifier, list of candidates, list of eligible voters, opening and closing times, as well as defining the resultant functions. Technically, the election authority determines the cryptographic primitives and algorithms for the protocol during this step. Then the Election trustees run the related algorithms and procedures to generate the election parameters, both public and private parameters such as the election's secret key. At the end of this phase, the public parameter of election, such as election policy, algorithms and procedures are published on the bulletin board.
- *Registration phase:* In this phase, all protocol participants generate their key material and publish the respective public parts. Each voter $\mathtt{v}_i$ is assigned her or his credentials, including the public and the private key. After offline and online registration phases, a list of legitimate voters along their public parameters is published on the bulletin board according to the election policy.
- *The voting ceremony* is the phase where voters generate their ballot and cast their ballot by sending it to the append-only public bulletin board.
- *Tally phase:* The order of this phase depends on the election policy. Nevertheless, a generic e-voting scheme has the following steps in a different order.

  1. Verify the validity of each ballot, either illegitimate voter or invalid ballot.
  2. Remove the invalid ballots.
  3. Detect whether any voter casts more than one ballot and then choose one ballot for each voter based on the election policy.
  4. Anonymize the ballots by removing the public-id of each ballot and performing some mix-net servers.
  5. Apply the result function and compute the election result.

  Additional Note. A *result function*, $f_{\mathtt{res}}$, is a mapping that takes a pair $(\mathtt{id}, \mathtt{vote}_{\mathtt{id}})$ (voter identifiers and their vote) as input and returns a value (bit-string) that represents the election result. For instance, if the election policy states that the winner is the person who receives the most votes, then:

$$f_{\mathsf{res}}\big((5,A),(2,A),(4,B),(6,B),(1,A),(3,C)\big) \mapsto A$$

Of course, we can write another function, g_{res}, that on the mentioned input returns

$$g_{\mathsf{res}}\big((5,A),(2,A),(4,B),(6,B),(1,A),(3,C)\big) \mapsto (A:3,B:2,C:2),$$

resulting in the same winner, but from a security standpoint, we regard f_{res} and g_{res} to be two different result functions.

- *Verification phase:* During this phase, the whole process of the election should be verified, and it consists of two distinct operations. The first is conducted by individual voters using some secret parameters. The second one is formally carried out by some auditors/judges, but it must also be available for public verification. This means that any individual can verify the election process without requiring any secret information.

6.1.2 Security Notions in the Context of E-Voting Protocols

There have been numerous properties proposed to describe the level of security and reliability of an electronic voting protocol. Among them, privacy and verifiability are the most significant ones. The terms ***privacy*** and ***verifiability*** in an e-voting system refer to a broad concept incorporating various characteristics. As a result, we distinguish between the following properties, each capturing a different aspect of privacy and security.

- **Privacy** as a fundamental human right [249] guarantees that all voters cast their ballot as intended without any external force and also ensures that no information about the voter's intention is leaked, aside from what can be deduced from the election result. Hence, in the context of e-voting protocol, we consider the following properties:

  1. **Ballot Privacy:** The protocol system must not leak any additional information on how each voter cast the ballot [226, 177, 92], aside from the information available from the election result. This also includes hiding whether a particular voter has participated in the election [144, 84].
  2. **Receipt-Freeness:** It states the protocol should not provide any evidence that helps voters prove how they voted to a third party [211, 40]. In fact, it expresses a stronger notion of privacy, stating that my vote should stay hidden even if I am eager to disclose it. More specifically, receipt-freeness implies that, even if an attacker forces the voter to reveal all of the information about the election (e.g., any confirmation message), he should not be able to tell whether the voter provided her true vote or a forged one.
  3. **Coercion-Resistance:** The protocol should preserve privacy even for the voter under coercion. Coercion resistance is essential in voting protocols because it protects voters from hostile parties such as coercers or vote-buyers who attempt to impose them to vote in a certain way [226, 168, 80].
  4. **Fairness:** The voting system should not reveal any partial results before the voting is finished [177].

- **Verifiability** captures the fact that the outcome of the tally phase is an actual election result, and it can be verified. As defined by Sako and Killian [230], a verifiable e-voting protocol should fulfil the following two properties:

  1. **Individual Verifiability:** All voters can verify that their cast ballots are included in the set of all votes. This phase is carried out in a designated-verifier way and is not necessary a public-verifier.
  2. **Universal Verifiability:** Any observer can verify that the tally has been correctly computed from the sets of votes. In contrast to individual verifiability, this phase must be publicly verifiable.

  The above two properties can be broken down into three distinct steps: *cast-as-intended*, *recorded-as-cast* and *tallied-as-recorded*. Any electronic voting system that includes a verification mechanism for all these three steps is referred to as *End-to-End Verifiable* protocol.

6.2 Formal Definitions

Before we can formally define, analyse and examine the properties of electronic voting protocols in our study, we must formally define the computational model. Here, we present our computational model, which is a simplification of the general and comprehensive computational framework introduced in [182]. Consider the protocol description Π^{evoting}. We define the following concepts.

6.2.1 Computational Model

In this section, we present a generic definition of an e-voting protocol and the computational model that we will use later to formalize the properties of our e-voting protocols. we customize the KTV framework introduced in [182] which is based on the interactive Turing machines (ITMs) communicate via tapes. The ITMs may perform probabilistic polynomial-time computations in the length of the security parameter and the input received so far. Following [182] an e-voting protocol specifies the *process* (program) carried out by honest voters and honest voting authorities such as honest registration tellers, tallying tellers, bulletin board, etc along with some public parameters pp. First we recall the main concepts used in KTV framework.

Process is a set of probabilistic polynomial-time Interactive Turing Machines (ITMs) that can perform internal computation and can communicate with other processes by sending messages via (external) input/output channels. A single session-run of a protocol includes a description of the corresponding process, the security parameter, and all random coins used by the program. Namely a run is determined by the random coins used by the programs in the process. As a result, the process defines a family of probability distributions over a run, which are indexed by the security parameter. Each process is initiated by a program known as a master program, and at each point during the execution of a process, only one program is active. We present the process by

$$\pi^{\ell} = \mathsf{p}_1 \| \mathsf{p}_2 \| \ldots \| \mathsf{p}_t$$

where p_is are PPT programs and each of them take as input the security parameter ℓ. We assume all program p_i has a running time polynomial in term of ℓ. Additionally, we let $\pi_1 \| \pi_2$ to refer the composition of two processes π_1 and π_2.

An e-voting protocol is a tuple of $\Pi^{\mathsf{evoting}} = \langle n_{\mathsf{v}}, \mathsf{Agent}, \Pi^{\mathsf{h}}, \mathsf{cList}, \Gamma^{\mathsf{elc}}, f_{\mathsf{res}} \rangle$[1] with n_{v} voters where:

- Agent is a finite set of protocol participants (parties). In a typical e-voting protocol, Agent includes voters $\{\mathsf{v}_1, \ldots, \mathsf{v}_{n_{\mathsf{v}}}\}$, election trustee $\mathfrak{T}$, registrar $\mathfrak{R}$ and a bulletin board $\mathfrak{BB}$. We also assume Agent includes *scheduler* $\mathfrak{s}$ which acts as the master program of the protocol process and trigger the protocol participants and the adversary in the appropriate order. We consider a set of channels which agents communicate through them with each other and also a channel that the adversary connects to the agent through that.

  We use a positive boolean formula over propositions of the form $\mathsf{hon}[a]$ for an agent $a \in \mathsf{Agent}$ to describe a group or groups of participants that can guarantee, when running their honest programs, that a protocol's goal is achieved. For example the following formula is true if the bulletin board, the scheduler and at least one of the voters are honest:

  $$\mathsf{hon}[\mathfrak{BB}] \wedge \mathsf{hon}[\mathfrak{s}] \wedge \big(\mathsf{hon}[\mathsf{v}_1] \vee \ldots \vee \mathsf{hon}[\mathsf{v}_n]\big) \tag{6.1}$$

  Additional Note. Although the protocol description does not provide any specific program for coercer, it is treated as an implicit agent. It is required that in security assumptions (or treat model), the coercer's limitations are specifically described, such as computationally bounded coercer. Formally, a coercer is a PPT interactive algorithm which connects to protocol agents, instructs them to perform a dummy strategy dum that represents all the possible programs the coercer can ask to achieve the specific goal ρ^*. Any participant of the protocol or some external party can carry out the coercer program.

- $\Pi^{\mathsf{h}} = \{\hat{\pi}[a] : a \in \mathsf{Agent}\}$ is a finite set where for $a \in \mathsf{Agent}$, $\hat{\pi}[a]$ is a description of a program that agent a runs (if a is honest).We distinguish between honest and dishonest agent. The honest parties follow the programs determined in the protocol, $\hat{\pi}[a]$, whereas dishonest parties who are under the control of the coercer, deviate from the pre-determined program in arbitrary ways. We use π^*_a to model potential dishonest behavior of a. We assume the coercer completely controls the dishonest voters but a voter under coercer can either run the dum strategy or run the counter strategy to achieve her own goal ρ(see 37). Furthermore, the voter under observation is an honest voter who runs the program honestly with her honest choice, while an observer attempts to guess her choice.

  An instance of protocol Π^{evoting} is a process of the form $\hat{\pi}_\Pi \| \pi_{\mathcal{A}}$, where

  $$\hat{\pi}_\Pi = \hat{\pi}_{a_1} \| \ldots \| \hat{\pi}_{a_k} \text{ with } a_i \in \mathsf{Agent},$$

  and π_A represents an arbitrary probabilistic polynomial-time program that an adversary $\mathcal{A}$ may perform, including all dishonest agents, $\pi^*_1 \| \ldots \| \pi^*_m$. A run of Π^{evoting} is a run of some instance of the protocol.

[1]In the original definition of the protocol in [182] they mention some other parameters such as the set of channels, but to avoid the heavy notation and more importantly since we don't use them in our analyze, we don't include them in our definition of the protocol.

- cList is the set of all possible valid candidate (choices)[2] including $\perp$, a special symbol that indicates the vote is invalid or has not been cast, so does not have to be counted.

- Γ^{elc} is probability distribution on the possible choices, including abstention *i.e.*, if the possible choice number is k then

$$\Gamma = (p_0, p_1, \ldots, p_k)\,;\ \sum_{i=0}^{k} p_i = 1$$

  where p_0 is the probability of abstention. An honest voter makes her choice according to this distribution.[3] We suppose that the coercer is aware of candidate distribution, maybe through the use of opinion surveys, and that he employs this information in his strategy. According to intensive research, the precise distribution is crucial for a system's level of coercion-resistance. Hence we make this distribution explicit.

- f_{res} is a result function that takes a list of ballots/votes as input and returns a string, on which the election results are determined. For instance, a result function may be one that computes the total number of votes cast for each candidate.

Definition 37. *Property γ of a protocol is a subset of all runs of Π^{evoting}. For a property γ we let $\Pr\left[\Pi^{\mathsf{evoting}}(\ell) \mapsto \gamma\right]$ to denote the probability that a run of Π^{evoting} with security parameter ℓ belongs to γ.*

Goal: In the KTV framework, concepts such as verifiability and coercion-resistance are interpreted based on the desired goals of a protocol. As an example for voting protocols, the desired goal could be that the announced election result reflects voters' actual votes. Formally, a goal ρ is a protocol property. If the property ρ is supposed to imply that the coerced voter wishes to vote for candidate c^* then ρ would contain all runs in which the coerced voter voted for this candidate, and this vote is in fact tallied.

Definition 38. *The ideal protocol is a protocol in which voters $\mathsf{v}_1, \ldots \mathsf{v}_n$ send their choice $\mathsf{c}_1, \ldots, \mathsf{c}_n$ directly to the fully trusted party which then computes $f_{\mathsf{res}}(\mathsf{c}_1, \ldots, \mathsf{c}_n) = \mathsf{res}$ and outputs the result without revealing any additional information.*

Now based on the KTV framework we define verifiability and coercion resistance properties.

6.2.2 Coercion-Resistance

In [181], the authors present a definition of quantitative coercion-resistance following similar ideas as in Definition 40. We will here use their strategy version. The idea is that the parameter δ^{cr} bounds the ability of a coercer being able to distinguish whether a coerced voter complied the coercion request or cast her own choice of ballot using some counter strategy. We let γ denote a property defining the goal of the coerced voter, e.g. to vote for a specified candidate. The parties are an observer O, who can use public data, n_h honest voters and an additional voter under observation v_{obs}, whose vote the observer tries to guess

[2]In this thesis, we often use the terms "candidate" and "choice" interchangeably.

[3]Choices can also be preference lists of candidates, etc.

Definition 39. $\Pi^{\mathsf{evoting}} = \langle n_{\mathsf{v}}, \mathsf{Agent}, \Pi^{\mathsf{h}}, \mathsf{cList}, \Gamma^{\mathsf{elc}}, f_{\mathsf{res}} \rangle$ *achieves* δ^{cr}*-coercion-resistance if for all dictated coerced strategies* $\pi_{\mathsf{v}_{co}} \in \Pi^{\mathsf{h}}_{\mathsf{v}}$ *there exists a counter-strategy* $\tilde{\pi}_{\mathsf{v}_{co}}$ *s.t. for all coercer programs* π_c*:*

- $\Pr[(\pi_c || \tilde{\pi}_{\mathsf{v}_{co}} || \pi_v)^{(l)} \mapsto \gamma]$ *is overwhelming.*
- $\Pr[(\pi_c || \pi_{\mathsf{v}_{co}} || \pi_v)^{(l)} \mapsto 1] - \Pr[(\pi_c || \tilde{\pi}_{\mathsf{v}_{co}} || \pi_v)^{(l)} \mapsto 1]$ *is* δ^{cr}*-bounded,*

with bounded and overwhelming defined in the security parameter. The first part says that the voter is able to achieve her goal (e.g. vote for a specific candidate) and the second part says that the coercer's distinguishing power is bounded by δ^{cr}. This level of coercion-resistance depends on several parameters especially the probability distribution on the candidates. However this definition basically gives the advantage in the probability of the coercer correctly pointing to the voter, when not following the coercer's instruction, compared to when she does.

Whereas this definition gives a level of coercion-resistance, it does not tell the full story.

To give some insight on this let us consider the following example two different election systems. System A outputs voter names and corresponding votes with probability $\frac{1}{2}$, completely breaking privacy, and otherwise it only outputs the election result. Neglecting the information obtained from the election result we get $\delta^A = 1/2$. In system B the voter secretly gets a signed receipt of her vote with probability 1/2 and otherwise the protocol works ideally. In this case a coerced voter can always cast her own choice and claim that no receipt was received. A voter following the coercer's instruction will with probability 1/2 give the corresponding receipt, i.e. we again have $\delta^B = 1/2$. In this case we also get $\delta^B = 1/2$ e.g. if the adversary always outputs "1" when getting a receipt.

However, the two systems are very different from the point of view of the voter: in system A the coerced voter gets caught cheating with probability 1/2, whereas in system B, the voter always has plausible deniability.

6.2.3 Privacy

With the above notation we define the δ-privacy as follows:

Definition 40 (δ-privacy [183])**.** $\Pi^{\mathsf{evoting}} = \langle n_{\mathsf{v}}, \mathsf{Agent}, \Pi^{\mathsf{h}}, \mathsf{cList}, \Gamma^{\mathsf{elc}}, f_{\mathsf{res}} \rangle$ *achieves* δ*-privacy if*

$$\Pr[(\pi_O || \pi_{\mathsf{v}_{obs}}(v_0^O) || \pi_v)^{(l)} \to 1] - \Pr[(\pi_O || \pi_{\mathsf{v}_{obs}}(v_1^O) || \pi_v)^{(l)} \to 1]$$

is δ*-bounded as a function of the security parameter* ℓ *for all vote choices* v_0^O *and* v_1^O *of the observed voter.*

The value δ will depend on the chosen vote distribution.

6.2.4 Receipt-Freeness

Following [181], definition 39 also covers receipt-freeness. However, we again argue that modelling some variants is useful. The following definition is based on a swap of $\pi_{\mathsf{v}_{co}}$ and $\tilde{\pi}_{\mathsf{v}_{co}}$ in Definition 50, and models vote buyers who do not want to pay a "free lunch" to vote sellers who follow their own goal. The voter's goal γ, here can be to cast a specified vote or set of votes.

Definition 41 (Weak Vote Buying Resistance)**.** *For a given small* p_{fl}*,* S *achieves* δ^{wvb}*-coercion-resistance if for all dictated coerced strategies* $\pi_{\mathsf{v}_{co}} \in V_S$ *there exists a counter-strategy* $\tilde{\pi}_{\mathsf{v}_{co}} \in V_S$ *s.t. for all coercer programs* $\pi_c \in C_S$*:*

- $\Pr[(\pi_c \| \tilde{\pi}_{\mathsf{v}_{co}} \| \pi_v)^{(l)} \mapsto \gamma]$ *is overwhelming.*
- $\Pr[(\pi_c \| \pi_{\mathsf{v}_{co}} \| \pi_v)^{(l)} \mapsto 1] - \Pr[(\pi_c \| \tilde{\pi}_{\mathsf{v}_{co}} \| \pi_v)^{(l)} \mapsto 1]$ *is* δ^{wvb}*-bounded and* $\Pr[(\pi_c \| \tilde{\pi}_{\mathsf{v}_{co}} \| \pi_v)^{(l)} \mapsto 1]$ *is* p_{fl}*-bounded.*

We here interpret outputting "1" as paying the vote seller and this definition bounds how often an instruction-following vote seller gets paid by a vote-buyer (by $\delta^{wvb} + p_{\mathrm{fl}}$), but under the condition that a voter who casts another vote is only paid with a (very) small probability p_{fl}. This is a weakened vote-buyer model but interesting since a vote buyer should avoid vote sellers going for a "free lunch". If the probability of an honest vote seller getting paid is low, it would help curb vote selling (even though the vote buyer could increase the price and create a "vote selling lottery"). In this definition, it also makes sense to drop the quantification over the coercer's strategies to see the resistance to vote buying for different vote choices.

6.3 Verifiability in the Context of E-Voting Protocol

Numerous problems with e-voting systems have been reported, according to documented cases, in various countries, where votes could have been dropped or miscounted (see, e.g., [95, 103, 176, 244, 261, 147, 243]). It is therefore crucial to ensure that, if the final election result does not correspond to the voters' votes, then this can be detected publicly. This basic security property is called *public end-to-end verifiability*, and, ideally, it should be guaranteed without having to trust any of the election authorities.

In order to establish a proper framework for defining the concept of verifiability, a comprehensive investigation has been conducted into the development of an electronic voting system with end-to-end verifiability security properties. End-to-end verifiability features imply that if the election protocol passes a particular procedure, the published election result is correct, i.e., corresponds to the votes cast by the voters, even if voting devices and servers have programming flaws or are malicious.

A formal definition of a verifiable e-voting system is proposed by Benaloh in 1980 [36]. However, since then, various definitions have been proposed in the literature that capture an e-voting protocol's verifiability. These definitions differ in many aspects, such as underlying models (Computational or symbolic [182]) and assumptions. In [85] Cortier *et al.* review all formal definitions of verifiability provided in the literature and place them within the framework proposed by [182], resulting in a consistent approach of verifiability. Additionally, we use this framework to analyze the verifiability of our protocols. We will briefly describe the concept and the notations in this section. We refer to [85] for additional information.

6.3.1 Verifiability; Formal Definition

The definition of verifiability requires an agent $\mathfrak{J} \in \mathsf{Agent}$ responsible for the verification phase, which includes a variety of checks (depending on the protocol specification), including verification of all zero-knowledge proofs (if any) and consideration of voter complaints. In any protocol, the judge may be a regular protocol agent or an external party provided with information by (possibly untrusted) protocol participants.

Informally, the KTV framework expresses that the protocol Π^{evoting} is considered as a verifiable scheme if $\mathfrak{J}$ accepts a run only if the protocol's goal is achieved. We emphasize two subtle pointers linked to this interpretation before formally defining verifiability in the KTV framework. We emphasize that if the judge never accepts a run, these conditions would be easily met in any protocol. As a result, the definition of verifiability should include the conditions under which the aim must be achieved, and the judge should accept runs.

Now we describe our customized definition of verifiability based on the generic verifiability definition proposed in [182]. Although we simplified the framework, it is still sufficient for our setting.

We consider the goal ρ simply as a set of protocol runs for which "the announced election result corresponds to the actual choices of the voters" where the description of a run contains the description of the protocol, the adversary with which the protocol runs, and the random coins used by these entities. Considering the terminology and notation introduced in [36] we denote by

$$\Pr\Big[\,(\hat{\pi}_{\mathsf{P}} \| \pi_{\mathcal{A}})^{(\ell)} \mapsto (\mathfrak{J}: \mathsf{accept})\,\Big]$$

the probability that process π, with security parameter ℓ produces a run which is accepted by $\mathfrak{J}$. Analogously, by

$$\Pr\Big[\,(\hat{\pi}_{\mathsf{P}} \| \pi_{\mathcal{A}})^{(\ell)} \mapsto \neg\gamma \mid (\mathfrak{J}: \mathsf{accept})\,\Big]$$

we denote the probability that π produces a run which is not in ρ but nevertheless is accepted by $\mathfrak{J}$. The verifiability definition requires that the later probability be negligible in term of the security parameter.

Definition 42 (Verifiability [182]). *For e-voting protocol Π^{evoting} with the $\mathfrak{J} \in \mathsf{Agent}$, and $\phi = \mathsf{hon}[\mathsf{Agent}]$, let $\delta \in [0,1]$. We say that a goal γ is guaranteed in Π^{evoting} by ϕ and it is δ-verifiable by the judge $\mathfrak{J}$ if for all instance $\pi = \hat{\pi}_P \| \pi_{\mathcal{A}}$ and for all PPT adversary $\mathcal{A}$, the following conditions are satisfied:*

1. *If $\phi = \mathsf{True}$ in π then $\Pr\big[\pi^{(\ell)} \mapsto (\mathfrak{J}: \mathsf{accept})\big]$ is overwhelming as a function of the security parameter.*
2. $\Pr[(\hat{\pi}_{\mathsf{P}} \| \pi_{\mathcal{A}})^{(\ell)} \mapsto \neg\gamma, (\mathfrak{J}: \mathsf{accept})]$ *is δ-bounded as a function of ℓ.*

Expressing the goals as properties of a protocol is a powerful and flexible tool, which allows capturing different aspects of verifiability in e-voting protocol as follows:

- In this case, the goal aims the ***individual verifiability***, the voter also plays the role of judge by running a private (designated verifier) rather than a public procedure. As a concrete example for this case, we can mention a goal that includes all runs in which the voters vote is counted as cast [230].
- By considering the goal that includes all runs in which the ballots shown on a bulletin board are counted correctly, the concept of verifiability capture the ***universal verifiability*** [230].
- The KTV definition of verifiability captures the ***eligibility verifiability*** if the goal includes those runs where only eligible voters vote at most once [240].

However, we can interpret the three definitions above into a single goal known as ***global verifiability***, which encompasses all runs in which the announced result exactly corresponds to the votes cast by eligible voters.

For our subsequent verifiability analysis of protocols in 7.9.5 and 8.6.5, we instantiate the verifiability definition with the goal $\gamma(\varphi)$ proposed in [85]. This goal captures the intuition of γ given before. The parameter φ is a Boolean formula to describe which protocol participants are assumed honest. The goal $\gamma(\varphi)$ is defined formally as described next.

Definition 43 (Goal $\gamma(\varphi)$ [85])**.** *Let* P *be a voting protocol. Let I_h and I_d denote the set of honest and dishonest voters, respectively, in a given protocol run. Then, $\gamma(\varphi)$ consists of all those runs of the voting protocol* P *where either*

- *φ is false (e.g., the adversary corrupted a voter that is assumed to be honest), or*
- *φ holds true and there exist (valid) dishonest choices $(\mathsf{c}_i)_{i\in I_d}$ such that the election result equals $(\mathsf{c}_i)_{i\in I_h\cup I_d}$ (modulo permutation), where $(\mathsf{c}_i)_{i\in I_h}$ are the honest voters' choices.*

6.4 Simulation-Based Security in E-Voting Protocol

We analyze our protocol using the definition, notion, and notation defined in [46]. In this paper, Bernhard *et al.* propose a new game-based definition of privacy called bPRIV. Additionally, they identify new properties, strong consistency, and strong correctness, which demonstrate that tallying does not leak sensitive information. They validate this security notion by showing that bPRIV, strong consistency and correctness for voting scheme implies its security in a simulation-based sense. We briefly present the necessary notation and definition here and later conduct the security analysis for our protocol.

It's worth noting that the primary reason we use this setting to establish our protocol's privacy level is that our protocol heavily relies on the re-voting policy, and we need to demonstrate that this re-voting policy has no effect on the ballot's privacy.

6.4.1 bPRIV Property

Ballot privacy intuitively captures the idea that an e-voting protocol does not reveal information about votes cast beyond what is unavoidably leaked (e.g., the result of election) during its execution. They formalize this idea in [46] by playing a game between adversary and challenger in which the adversary attempts to differentiate between two worlds. This game is formalized through experiments with the oracles listed below:

- Oboard oracle presents the adversary's ability to see the publishable portion of the ballot box, i.e. the bulletin board. Oracle returns the value $\mathsf{Publish}(\mathfrak{BB})$. We consider a single visible bulletin board to be a single bulletin board in our protocol. As a result, we will not consult this oracle.
- OvoteLR, the left-right oracle, takes two plausible votes $(\mathsf{vote}_0, \mathsf{vote}_1)$ for voter $\mathtt{v_{id}}$ to generate ballots ballot_0 and ballot_1 and then submit them to the bulletin board (ballot_β to the bulletin board $\mathfrak{BB}_\beta$).
- Ocast oracle allows the adversary to cast ballot on behalf of any voter. In the original definition, the ballot is published on the bulletin board as long as the ballot is valid with respect to $\mathfrak{BB}_\beta$. However in our protocol, any ballot will be published on the bulletin board.

- Otally return the result of the election which in both worlds it is obtained by tallying $\mathfrak{BB}_0$ along with the additional information (e.g. proof of the tally-correctness). In the first experiment, the adversary is provided by the real additional information (related to the $\mathfrak{BB}_0$) while in the second experiment, the additional information is simulated.

The adversary is allowed to call oracles OvoteLR, Ocast, Oboard in any sequence and as many times as he wishes, but the adversary can make a single call to Otally. After the adversary receives the response from the oracle Otally, the adversary returns his guess, β^*.

Definition 44. *[46] Consider a voting scheme Π^{voting} for a set I of voter identities and a result function f_{res}. We say the scheme has ballot privacy if there exists an algorithm* Sim *such that no efficient adversary can distinguish between games* $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{bprive}}(1^{\ell})[\beta = 0, 1]$ *defined by the oracles in Figure 6.1.is negligible.*

Additional Note. First, we emphasize that the bPRIV notion can be considered in Common reference string and random oracle models. Both models require a global setup that initiates a set of public parameters through the Global.init and describes some algorithms through Global.Setup that all parties, including adversary, can access. For example, in CRS-model the algorithm GlobalSetUp.init generates a CRS, and in the random oracle model, GlobalSetUp is a truly random function to which only oracle parties have access.

Having the setup assumption enables us to generate a simulated setup consisting of a simulator, Sim with additional powers that can be used to create reduction proofs. For instance, in the CRS model, the simulated setup would consist of generating a simulated CRS with a trapdoor that enables the generation of valid-looking proofs for false statements, whereas in the RO model, the simulator would program the oracle in such a way that it outputs a specific value on some values of interest. It is self-evident that for security reasons, the simulator outputs must be indistinguishable from a normal one to a PPT adversary.

As stated in [46], requiring this global assumption is necessary since the security level given by bPRIV would be too strong without it. Furthermore, the existence of a simulator capable of falsifying the proof in the absence of a global setup would violate the protocol's tally uniqueness.

6.4.2 Strong Consistency

The privacy given by the bPRIV concept relies heavily on the fact that the election result appears to be independent of the auxiliary data Π returned in the tally phase. The contradiction concealed behind this point might be interpreted as follows: while res should correspond to the expected result, it should not contain any sensitive data. To capture this property, in [46] the authors propose a bPRIV companion definition called strong consistency, which ensures that the result always corresponds to the result function applied to the votes and nothing else. It also mitigates the harm that an intentionally leaky re-vote policy could implicitly cause while tallying. Informally, strong consistency prohibits an adversary from embedding instructions in her ballots, resulting in the tallying leaking information on the honest votes or preventing the validation of honestly generated ballots. They formalize the concept by requiring the existence of an "extraction" algorithm, which with the help of the secret key outputs the underlying vote for each valid ballot.

FIGURE 6.1: bPRIV experiment: $\mathsf{Exp}^{\mathsf{bPriv}}_{\mathcal{A}}(1^{\ell})$

Experiment steps:

1. The challenger run the set up algorithm:

$$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^{\ell})$$

2. The adversary interact with oracle $\mathcal{O} \in \{\mathsf{OvoteLR}, \mathsf{Ocast}, \mathsf{Oboard}, \mathsf{Otally}\}$:

$$\beta^* \leftarrow \mathcal{A}^{\rightleftharpoons \mathcal{O}}(1^{\ell}, \mathsf{pk}, \mathsf{pp}_{\mathsf{election}})$$

3. The challenger choose the challenge bit: $\beta \xleftarrow{\$} \{0,1\}$

Oracles:

- $\mathsf{OvoteLR}(\mathtt{id}, \mathsf{vote}_0, \mathsf{vote}_1)$

```
Let $\mathsf{ballot}_0 = \mathsf{Vote}[\mathtt{id}, \mathsf{vote}_0]$
    $\mathsf{ballot}_1 = \mathsf{Vote}[\mathtt{id}, \mathsf{vote}_1]$
If $\mathsf{Valid}(\mathfrak{BB}_\beta, \mathsf{ballot}_\beta) = \perp$ return $\perp$
Else $\mathfrak{BB}_0 \leftarrow \mathfrak{BB}_0 \| \mathsf{ballot}_0$ and $\mathfrak{BB}_1 \leftarrow \mathfrak{BB}_1 \| \mathsf{ballot}_1$
```

- $\mathsf{Ocast}(\mathtt{id}, \mathsf{ballot})$

```
If $\mathsf{Valid}(\mathfrak{BB}_\beta, \mathsf{ballot}) = \perp$ return $\perp$
Else $\mathfrak{BB}_0 \leftarrow \mathfrak{BB}_0 \| \mathsf{ballot}$ and $\mathfrak{BB}_1 \leftarrow \mathfrak{BB}_1 \| \mathsf{ballot}$
```

- Oboard

```
return $\mathsf{Publish}(\mathfrak{BB}_\beta)$
```

- Otally :

$\beta = 0$:	$\beta = 1$:
$(\mathsf{res}, \pi_{\mathsf{res}}) \leftarrow \mathsf{Tally}(\mathfrak{BB}_0, \mathsf{sk})$	$(\mathsf{res}, \pi_{\mathsf{res}}) \leftarrow \mathsf{Tally}(\mathfrak{BB}_0, \mathsf{sk})$
	$\pi^*_{\mathsf{res}} \mathsf{SimProof}(\mathfrak{BB}_1, \mathsf{res})$
return $(\mathsf{res}, \pi_{\mathsf{res}})$	return $(\mathsf{res}, \pi^*_{\mathsf{res}})$

FIGURE 6.2: $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{sCons}}(1^{\ell})$

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathsf{sCons}}(1^{\ell})$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SetUp}(1^{\ell})$
$\mathfrak{BB} \leftarrow \mathcal{A}$
$(\mathsf{res}; \Pi) \leftarrow \mathsf{Tally}(\mathsf{sk}, \mathfrak{BB})$
If $\mathsf{res} \neq f_{\mathsf{res}}(\mathsf{Extract}(\mathsf{sk}; \mathsf{ballot}_1), \ldots, \mathsf{Extract}(\mathsf{sk}; \mathsf{ballot}_n))$
    Then return 1
    Else return 0
```

Definition 45 (**Strong Consistency** [46]). *A scheme* Π^{voting} *relative to a result function* f_{res} *has strong consistency if there exist*

- *an* **extraction** *algorithm* Extract *that takes as input a secret key* sk *and a ballot* ballot *and outputs* $(id, v) \in \mathsf{idSetcList}$ *or* $\perp$,
- *a ballot* **validation** *algorithm* ValidInd *that takes as input the public key of the election* pk*, a ballot* ballot*, and outputs* accept *or* reject*,*

which satisfy the following conditions:

1. *For any* $(\mathsf{pk}, \mathsf{sk})$ *that are in the image of Setup and for any* $(id, \mathsf{vote}) \in \mathsf{idSet} \times \mathsf{cList}$ *if*

$$\mathsf{ballot} \leftarrow \mathsf{Vote}(\mathsf{pk}, id, \mathsf{vote})$$

   then $\mathsf{Extract}(\mathsf{sk}, \mathsf{ballot}) = (id, \mathsf{vote})$ *with overwhelming probability.*

2. *For any* $(\mathfrak{BB}; \mathsf{ballot}) \leftarrow \mathcal{A}$, $\mathsf{Valid}(\mathfrak{BB}; \mathsf{ballot}) = \mathsf{accept}$ *implies* $\mathsf{ValidInd}(\mathsf{ballot}) = \mathsf{accept}$.
3. *Consider an adversary* $\mathcal{A}$ *and the experiment* $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{sCons}}(1^{\ell})$ *shown in figure 6.2.*

As stated in [46] if the result function does not use voter identifiers, as is the case with our protocol, it merely requires the following:

$$f_{\mathsf{res}}\big((\mathtt{id}_1, \mathsf{vote}_1), \ldots, (\mathtt{id}_n, \mathsf{vote}_n)\big) = f_{\mathsf{res}}\big((\mathtt{id}_{\sigma(1)}, \mathsf{vote}_1); \ldots, (\mathtt{id}_{\sigma(n)}, \mathsf{vote}_n)\big)$$

for some permutation σ, and so the Extract algorithm does not have to return the voter identifier.

6.4.3 Strong Correctness

Following [46] the third property required for security analysis is strong correctness, which reflects the strong independence between honestly created ballots and the ballot box (and the voting scheme's global setup). Informally, it protects the validity of an honestly-cast ballot, even in the presence of an adversarial manufactured ballot box.

Definition 46 (**Strong correctness** [46]). *Consider an adversary* $\mathcal{A}$ *that takes as input* pk *and has access to a global setup* GlobalSetUp *generated as expected. Then,*

$$\Pr\big[\,(id; \mathsf{vote}; \mathfrak{BB}) \leftarrow \mathcal{A}(\mathsf{pk}); \mathsf{ballot} \leftarrow \mathsf{Vote}(id, \mathsf{vote}) \mid \mathsf{Valid}(\mathsf{ballot}, \mathfrak{BB}) \neq \mathsf{accept}\,\big] < \mathsf{negl}(\ell)$$

where the probability is over the coins used by Kgen*,* $\mathcal{A}$*, and* Vote *algorithms.*

In [46] it is proved that if a voting protocol with the result function f_{res} is strongly correct, strongly consistent, and ballot private, then it has a simulation-based privacy.

Chapter 7

Practical and Usable Coercion-Resistant Remote E-Voting

Developing electronic voting systems for realistic elections that provide a minimum level of security and protect voters against malicious coercers is a difficult task. On top of this, making the system usable for non-skilled voters is even more challenging. Numerous secure electronic voting systems have been proposed to mitigate the possibility of coercion; however, many of them do so at the expense of efficiency, usability, all of which are essential for real-world elections.
This chapter aims to design a verifiable, voter-friendly and practical electronic voting system thst still achieves a good level of security and coercion-resistance.

Contents

7.1 Research Question and Our Contribution

Our contribution in this chapter is to extend electronic e-voting protocols towards a protocol that tolerates a level of human errors without sacrificing privacy requirements. In order to do this, we start by investigating the seminal coercion-resistant e-voting protocol by Juels, Catalano and Jakobsson proposed in [168], also known as the JCJ-protocol. In JCJ-based protocols, the voter needs to handle cryptographic credentials and be able to fake these in case of coercion.

From a security standpoint, we know that JCJ protocols rely on a long random secret string to maintain a reasonable level of security. However, research shows even typing the password (secret credential) correctly, without typos, can be a daunting task for many people, let alone storing the password in a safe place or memorizing it.

In a series of three papers, Neumann *et al.* [206, 207, 100] analyzed the usability of JCJ. They developed and implemented a practical credential handling system utilizing a smart card that unlocks the valid credential via a PIN code or faking the credential via faking the PIN.

Numerous attacks and issues with the security of this protocol are demonstrated in [98], most notably an attack on coercion-resistance caused by information leakage during duplicate ballot removal. Another issue Neumann et al. raised but did not resolve is that PIN typos occur frequently and would invalidate the casting vote without the voter being aware of it. Additionally, the smart card is a trusted component that a coercer can withdraw to force abstention, thereby creating a single point of failure.

Our research develops a new protocol that overcomes these difficulties by allowing credentials to be stored in traditional ways while being PIN-based and resistant to PIN errors. The PIN is a short string of letters and numbers that the voter must memorize. To protect voters from human error such as typos such as fat-finger-errors, we consider a set of allowable PIN errors. Moreover, we design the protocol so that a voter's ballot is valid if it contains the exact PIN or a permitted PIN error but is invalid for all PINs. We also instantiate our protocol by introducing proper cryptographic primitives to prove our proposed protocol's efficiency and usability.

Outline. This chapter is structured as follows. The outline of paper is as follows. After an introduction in Section 7.2 we give an overview of the orignal NV12 scheme in Section 7.3. Our improved protocol is presented in Section 7.4. In Sections 7.6, 7.7 and 7.8 we instantiate our protocol with three different cryptosystems, Paillier BGN and Functional Encryption scheme. In Section 7.9 we analysis the privacy and verifiability of our protocol. Finally we conclude in Section 7.10. We would like to emphasize that the research presented in this chapter was previously published in (E-Vote-ID-2021)[1] [98].

7.2 Introduction

One of the main threats in remote electronic voting is that they are inherently susceptible to coercion-attacks due to the lack of a voting booth. In their seminal paper, Juels, Catalano and Jakobsson [168] gave a formal definition of coercion-resistance and further devised a protocol (JCJ) satisfying this strong security property. To achieve this, JCJ assumes a coercion-free setup phase where the voter gets a credential which is essentially a cryptographic key. To cast a valid ballot this key needs to be entered correctly

[1]The International Conference for Electronic Voting

together with the vote. In case of coercion, the voter can simply give a fake random credential to the coercer and even cast a vote together with the coercer using this fake credential – the corresponding vote will be removed in the tally process.

However, the tally process of weeding out the ballots with fake credentials and duplicates suffers from a quadratic complexity problem in the number of voters and cast ballots. Several paper are devoted to reducing the tally complexity in JCJ, see e.g. [224, 17, 136, 245]; however, each with it drawbacks. Moreover, JCJ and similar constructions- we consider them under the term JCJ-type-protocols, however also suffer from usability deficits, see also [205]. Especially, the voter intrinsically cannot directly check if a cast ballot is valid and will be counted [160].

Moreover, the handling and storing of long credentials is a notorious usability problem, getting even harder with a coercer present. The usability was analyzed by Neumann *et al.* [206, 207, 100], which led to a protocol using smart cards for handling voter's credentials. The stored credential is combined with a PIN code to produce the full credential, compared with the credential stored by the authorities on the bulletin board. In [98], we revisit this protocol and present several attacks on coercion-resistance and verifiability, but also possible repairs.

Whereas the smart card provides a solution to the usability problem, it also comes with strong trust assumptions and problems such as:

- Their security model assumes that the smart card is generally needs to be trusted. A malicious card could e.g. use the wrong credential invalidating the cast ballot without detection, and we cannot let the voter check if the ballot is correct without introducing coercion threats.
- The coercer can take the smart card away from the voter to force abstention.
- It is more expensive, less flexible, and harder to update than a pure software solution.
- One of the attacks we found is that a coercer can use the smart card to cast ballots. This not only endangers coerced voter's real vote, but due to a leak of information in the weeding phase, the coercer can also detect, with non-negligible probability, whether the coerced voter has cast an independent ballot against his instructions.

We will present protocols that repair or at least diminish the attack probability of, the last point's attack probability by constructing new duplicate removal methods in JCJ. Furthermore, the protocols constructed in this chapter are hardware-independent: they could use a smart card or be implemented using a combination of a digitally stored cryptographic length key and a PIN only known by the voter. The long credential could be stored in several places – or hidden via steganography.

At ballot casting time, the software will take as input the digital key and the password to form the credential submitted with the vote. Depending on the level of coercion, the coerced voter can either fake the long credential or, for stronger levels of coercion, the voter can reveal the digitally stored credential to the coercer but fake the PIN. Due to our improved tally, the coercer will not know if he got faked credentials or PINs.

Another major problem with the original construction, already discussed as an open problem in [206], is the high chance of users making a PIN typo error which will invalidate the vote and remain undetected. Note that naively giving feedback on the correctness of the PIN is not possible for coercion-resistance as it would allow the coercer to check whether he got a fake PIN or not. Instead, we will define a set of allowed PIN errors (e.g. chosen by the election administrator), and we will consider a ballot as valid both if it has a correct PIN or an allowed PIN error, but invalid for other PINs. We construct protocols that at the tally time secretly check whether a given PIN is in the set of allowed PINs and will sort out invalid ballots. The protocols can accommodate general PIN error policies, however Wiseman *et al.* [260] studied usual errors in PIN entries. Two frequent errors are transposition errors (i.e. entering "2134" instead of "1234") and wrong digit number errors (i.e. entering "1235" instead of "1234"). However, correcting for both of these errors is however problematic, as we will see since the set of independent PINs becomes small.

7.3 A Brief Overview of NV12

Following the proposal of the JCJ voting scheme in 2005, it has gained much attraction. Due to JCJ's security properties, particularly its coercion-resistance properties, intensive research has been conducted to improve the JCJ from theoretical and practical points of view, such as the Civitas scheme [80]. Although Civitas improved the JCJ scheme by developing its instantiation of cryptographic components and implementation, it has practical and usability shortcomings. To overcome these drawbacks in a series of three papers Neumann *et al.* [206, 207, 100] carried out a usability analysis of JCJ. They developed and implemented a practical credential handling system (NV12) based on a smart card that unlocks the valid credential via a PIN number, while also faking the PIN. In [207], a few modifications were made to prevent side-channel attacks, an efficiency analysis was done, and finally [100], presented a prototype implementation and efficiency.

In [98] we demonstrate many attacks and issues with the protocol's security, most notably a coercion-resistance attack caused by information leakage caused by the removal of duplicate votes. Moreover they discuss how to repair these issues. These shortcomings include the Benaloh challenge problem, Brute force attack, Leaky duplicate removal, Fake election identifier and Smart card removal. Finally, to motivate our protocol in this section, we briefly describe two attacks that our protocol will repair.

Leaky duplicate removal , This is an attack on coercion-resistance but can also be an attack on verifiability. In the simplest form, the coercer uses the smart card to cast a vote with some trial PIN. The coercer wants to determine if this trial PIN is correct. According to the protocol the voter will cast her true vote using the correct PIN at some secret point during the voting phase. However, in the tally phase, credentials are weeded using plaintext equivalence tests (PETs) of the encrypted credentials directly on the submitted ballots.[2] If the coercer now sees an equivalence with his submitted trial ballot, he can guess that it was the voter casting the other ballot, probably with the correct PIN. Thus he has determined the correct PIN and that the voter defied his instructions in one go. To boost the attack he can simply try several PINs.[3] In standard JCJ, such an attack would not work since the submitted trial credential would have the same probability

[2]In general, this is not good for coercion-resistance since a coercer might detect a voter not following instructions across elections, see [160].

[3]Note that the coercer does not have to let the voter know that he follows this strategy. The voter only knows that the coercer has access to the card for some short time. Based on this, she could also decide not to cast her true vote at all, but then the protocol could not really be called coercion-resistant since the coercer has a very efficient strategy to force abstention.

of being identical to the coerced voter's credential as for it to be identical to any other voter's credential. Furthermore, the probability would be negligible.

A local adversary getting access to the smart card could also follow this strategy to try to know the PIN and cast valid votes. The voter might actually detect this if the voter checks the weeding on $\mathfrak{BB}$ and sees a duplicate of his own vote (note this was also mentioned in [223]), but the voter is not instructed to do this in the protocol. Thus the PIN is not protecting against unauthorized use of the smart card.

It is actually surprisingly hard to make a tally protocol that does not leak information to prevent this attack. The original JCJ protocol relies on guessing the real full credential can only happen with a negligible chance. A first repair could be to mix the ballots before weeding but after verifying the Zero-Knowledge proofs. This makes it difficult to implement certain policies, like the last valid vote counts; however, it fits nicely with the policy that a random selection from the valid votes count. Unfortunately, this does not prevent the attack. The coercer could mark his ballot by casting it a certain number of times which is likely to be unique. He then checks if he sees this number of duplicates or one more. Even if mix between each duplicate removal, which would be horrible from an efficiency perspective, we do not get a leak-free tally. The distribution of time until a PET reveals a duplicate will depend on whether the PIN was correct or not. Especially the coercer could cast a lot of votes with the same trial PIN, which would make detecting this more visible. There are other methods to limit the information leak in the tally which we will present below.

The protocol we will present in this chapter does not leak information about the number of duplicates per voter and has linear tally complexity (compared to the quadratic in JCJ) but has an obfuscated form of participation privacy.

Smart card removal. An obvious forced abstention attack is that the coercer simply demand to hold the smart card during the election period. This problem seems quite inherent to the smart card approach. We could let the voter hold several smart cards. However, holding several cards would be physical evidence which a voter with a local coercer probably would not want to risk. Furthermore, the number of cards allowed per voter could necessarily not be bounded. If each voter were allowed to hold e.g. 5 cards, the coercer would simply ask for five cards. If this is troublesome it seems better to leave the smart card only approach and allow the voter to also hold the credential as a piece of data as in standard JCJ. This can more easily be hidden (steganography could be an option here) even though theoretically this also has problems [236]. Our protocols below can be implemented with or without smart cards.

Additionally to the above attacks, some problems with the protocol do not fall under the category of attacks. The main usability and verifiability problem with the protocol is that PIN entry is error-prone, as was already stressed in the papers by Neumann et al. An obvious solution is to have a PIN check, e.g. a checksum check. However, this would mean that only certain PINs are valid PINs, and for a voter to present a fake PIN to a coercer, she would first have to prepare a valid fake PIN, which is less usable.

An option with higher usability is to have a policy of allowed PIN errors and accept full credentials corresponding to the PIN being entered with allowed errors. This is the approach we will essentially follow in this paper; however, our solutions will also work for checksum checks.

If JCJ had a method of verifying the cast votes, we would also be able to detect such PIN errors. Such a verification mechanism was suggested in [160] using the Selene approach. However, this check can only be made after vote casting has ended; thus too late to update a PIN typo.

Another problem is the assumption that the smart card is trustworthy. This does not seem like a valid assumption, at least for important elections. The smart card could simply use the wrong credential in a ballot, invalidating the vote. Further, this cannot be detected since the smart card is the only credential holder. At least the PIN encryption could be Benaloh tested, but not the credential. Furthermore, the smart card reader is also trusted. However, this might not be reasonable in practice. As an example, if the middleware on the reader allows the voter's computer or the network to display messages on the screen, e.g., to say it is waiting for a connection, then it could e.g., try to display fake hash values. A corrupted smart card could also easily break privacy by using encryption as a subliminal channel for vote choice. The smart card can also be seen as *a single point of failure* in light of this. We will thus focus on hardware-independent protocols.

7.4 Pin-Based JCJ E-Voting Protocol

To overcome the unsatisfying state-of-affairs described before in [98] we propose a practical and usable e-voting scheme, **PIN-JCJ** that satisfies verifiability, and coercion-resistance properties.

Compared to the previous electronic voting technique, our protocol has two significant innovations. First, it is a hardware-independent protocol, as we replace the smart card with the voter's supporting device, which can be easily duplicated. Second, we construct it in such a way that some human error is tolerated.

7.4.1 The Intuition Behind the PIN

In a nutshell, in our voting scenario the voter has two keys: a long key (a.k.a. long credential) which is stored on her supporting device (smart card or another device) and a short key, namely PIN, which should be memorized by the voter. Our goal is to design a voting protocol that tolerates some level of human error. In other words, the validity of a ballot and the result of the decryption algorithm tolerate human errors, such as typos (fat finger errors) regarding the PIN. The naive solution to do this is for each PIN, "a" we generate an $\mathsf{ErrorList}_a$, and in the tally phase, to verify the legitimacy of the ballot, check whether the input PIN is in the set of error list or not.

$$\mathbf{pin} = a : \ \mathsf{ErrorList}_a = \{a_1 = a, a_2, \dots, a_k\}, \left|a\right| = \left|a_i\right| \tag{7.1}$$

This method has several significant privacy and efficiency issues, and to overcome these drawbacks, we propose the following new approach:

Our approach is based on polynomial evaluation, which allows us to determine whether or not a PIN is legitimate efficiently. This is accomplished by generating a list of approved PINs based on the user's PIN a and the election policy. From now on, we refer to this list as an ***Error List***. We emphasize that to have a constant degree polynomial for all voters, the error list must have the same number of PINs, which might contain duplicates. From this, we generate the ***pin-Polynomial***, as follows, which has all $\mathsf{ErrorList}_a$ members as its root:

$$\text{pin-polynomial: } \mathsf{poly}_{\mathbf{pin}}(x) = \prod_{i=1}^{k} (x - a_i) = \sum_{i=0}^{k} p_i x^i \tag{7.2}$$

In order to check the validity of the PIN typed by the voter, it is then sufficient to check whether the polynomial value on this **pin** is equal to zero or not.[4]

It is obvious that this polynomial must be kept secret at all times to prevent the coercer from recovering the PIN by factorizing it. As a result, we must operate with encrypted polynomials, which brings us to our next challenge: polynomial evaluation under this encryption. Namely, given the polynomial encryption as its encrypted coefficient,

$$\mathsf{poly}_{\mathbf{pin}}(x) = \sum_{i=0}^{k} p_i x^i \Rightarrow \mathsf{Enc}(\mathsf{poly}_{\mathbf{pin}})(x) = \sum_{i=0}^{k} \mathsf{cp}_i x^i,$$

as well as a ciphertext $\mathsf{CT}_{\mathbf{pin}} = \mathsf{Enc}(\hat{a})$ that is the encryption of the entered PIN, we need to compute $\mathsf{Enc}(\mathsf{poly}_{\mathbf{pin}}(\hat{a}))$.

Therefore in the next step, we have to find a way to prove (publicly) that the individual voter's polynomial is correctly evaluated without endangering the coercion-resistance.

Further, while solving this problem, we will also focus on efficient protocols to obtain a practical JCJ scheme with (almost) linear tally time in the number of voters. To obtain this we need to sacrifice perfect privacy. We only have participation privacy in the first scheme by obfuscation inspired by [136, 180]. Here ballots are submitted with an ID, and homomorphic Paillier encryption can then be used to evaluate the polynomial. However everybody, e.g. an independent authority, can cast votes labeled with ID, which will later be discarded as invalid. Thus the actual participation of the voter is obfuscated, and the voter can deny having participated in the election. Optionally, we could also follow the JCJ alternative method in [136] to achieve perfect privacy; however, the cost will be that the voters twice have to defy the coercer and interact with the voting system. In the second scheme using BGN encryption, the information leak from duplicate removal will not be negligible but bounded, and this scheme does not satisfy linear tally efficiency.

The next section will present the protocol description by introducing the cryptographic building blocks and their algorithm. Also, for simplicity, we describe the protocol with a single trusted party, but it is possible to run this protocol distributively. We will also not specify all parts of the distributed registration phase and the Benaloh challenges; this can be implemented as in the NV12 scheme with some obvious modifications and the repairs presented in [98].

7.5 Protocol Description; Participants, Primitives and Framework

We now present the **PIN-JCJ** e-voting protocol on the conceptual level. In section 7.5.4, we will then instantiate our protocol by introducing concrete cryptographic primitives to demonstrate the efficiency and usability of the protocol.

7.5.1 Protocol Participants

The **PIN-JCJ** protocol is run among the following participants: *Election authority* in charge of running elections and declaring elections. *Election trustees* are in charge of the tallying process, and they are the only parties in possession of the election secret

[4]Note there is a small problem here since we are in composite order groups and the polynomials might have more roots than the allowed PINs. However, the probability in general is negligible.

keys. n_{v} *voters*, each having their own *voting supporting device*,vsd. A set of *supervised registrars* administrate the electoral register and provide the voters with their credentials. A set of mix-servers (mix-net) and an append-only bulletin board $\mathfrak{BB}$.

We also presume that all parties communicate with one another via authenticated channels. An authenticated channel from each voter to the bulletin board allows the voter to post data on the bulletin board, for example, cast her ballot, or file a complaint. To describe the **PIN-JCJ** protocol, we will first go through the cryptographic primitives that are employed in the scheme. We separate two sets of primitives for instantiation: those that use standard public encryption schemes and those that use the public functional encryption scheme.

7.5.2 Cryptographic Primitives

We use the following cryptographic primitives:

- **An IND-CPA-secure public-key encryption scheme:** $\Pi^{\mathsf{pke}} = (\mathsf{Kgen}, \mathsf{Enc}, \mathsf{Dec})$. Our protocol requires performing a polynomial evaluation on encrypted data; namely, the underlying public-key cryptosystem must support re-encryption. To this end, we consider three approaches: A public-key encryption scheme that allows multiplication on the ciphertext or a homomorphic public-key encryption scheme or functional encryption scheme. We detail the instantiation of the protocol with these three cryptosystems in section 7.5.4. However, for the time being, we are only considering a public-key cryptosystem with IND-CPA-secure property as the protocol's security and privacy are dependent upon it.

  Additional Note. As mentioned above, the underlying public-key cryptosystem needs to have an extra algorithm, as a re-encryption algorithm. On the other hand, for security analysis, we require the cryptosystem to be non-malleable, which contradicts the re-encryption requirement. We employ a public-key cryptosystem with a re-encryption algorithm to address this issue, w, but we require that each ciphertext contain a zero-knowledge proof of knowledge. We refer to [44] for more detail on the CCA-secure cryptosystem for e-voting protocol.

- **Non-interactive Zero-Knowledge proof:** For the protocol's verifiability, we use a non-interactive zero-knowledge proof system to prove the correctness and plaintext knowledge for all encrypted data, such as "Proof of Plaintext Knowledge" and "Proof of Decryption Validity." We may utilize either a zero-knowledge proof system or a witness-indistinguishable proof system for this aim. Formally we will use the following proofs in our protocol:

  1. A *proof of correct key generation*, i.e., a *non-interactive zero-knowledge proof* (NIZKP) π_{Kgen} for proving the correctness of a public key pk w.r.t. Π^{pke}. The underlying relation $\mathrm{R}_{\mathsf{KeyGen}}$ is

  $$\big(x = \mathsf{pk}, w = (\mathsf{random}, \mathsf{sk})\big) \in \mathrm{R}_{\mathsf{KeyGen}} \Leftrightarrow (\mathsf{pk}, \mathsf{sk}) = \mathsf{Kgen}(\mathsf{random}). \tag{7.3}$$

  2. The *proof of correct encryption* π_{Enc}, i.e., a NIZK proof of knowledge for proving the correctness of a ciphertext ct w.r.t. election key pk, voter public key pk_{v}.

3. A *proof of shuffle*, π_{Shuffle} with respect to the cryptosystem Π^{pke}, i.e., NIZK proof of knowledge for the following relation $\mathrm{R}_{\mathsf{shuffle}}$:

$$\begin{gathered}\big(x = ((\mathsf{ct}_i)_{i=1}^n, (\mathsf{ct}'_i)_{i=1}^n), w = ((\mathsf{r}_i)_{i=1}^n, \sigma)\big) \in \mathrm{R}_{\mathsf{shuffle}} \\ \Updownarrow \\ \big(\sigma\colon [n] \to [n] \text{ bijective }\big) \wedge \big(\forall i \in [n]\colon \mathsf{ct}'_i = \mathsf{ReEnc}(\mathsf{pk}, \mathsf{ct}_{\sigma(i)}; \mathsf{r}_i)\big).\end{gathered} \tag{7.4}$$

In other words, the public statement of a proof of shuffle consists of two ciphertext vectors $(\mathsf{ct}_i)_{i=1}^n$ and $(\mathsf{ct}'_i)_{i=1}^n$ of the same size, and if the prover's output is valid, then this implies that the prover knows random coins $(\mathsf{r}_i)_{i=1}^n$ and a permutation σ over $[n]$ such that ct'_i is a re-encryption of $\mathsf{ct}_{\sigma(i)}$ (using randomness r_i).

4. A *proof of correct decryption* π_{Dec} with respect to Π^{pke}, i.e., a NIZK for the following relation $\mathrm{R}_{\mathsf{dec}}$:

$$\begin{gathered}\big(x = (\mathsf{pk}, m, \mathsf{ct}), w = \mathsf{sk}\big) \in \mathrm{R}_{\mathsf{dec}} \\ \Updownarrow \\ \big(m = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})\big) \wedge (\mathsf{pk}, \mathsf{sk}) \leftarrow \Pi^{\mathsf{pke}}.\mathsf{Kgen}.\end{gathered} \tag{7.5}$$

For all the above relations we can use either a zero-knowledge proof system or a witness-indistinguishable proof system for this purpose.

- **EUF-CMA-secure signature scheme:** We assume that every message encrypted by a protocol participant contains some election parameters such as the election identifier, and to avoid the heavy notation, we use the following convention: Signing some message m implies that the signature is computed on the tuple $(m; \mathsf{pp})$ where the second components are public election parameters including an election identifier.

7.5.3 Protocol Framework

A protocol run consists of the following phases:

- **Setup Phase.** This procedure is carried out in the same manner as a standard e-voting protocol. The election authority sets up the election, establishes all the policies, such as re-voting policy and the type of error that the protocol will tolerate, and publishes details about the ballot design and other procedures. For example the re-vote policy specifies whether voters can cast only one ballot or many ballots. Furthermore, which ballot will be counted as the voter vote in the tally phase in the latter case.

  Then the election trustees distributively generate the election key through distributed threshold secret sharing and publish the public part of it on the bulletin board, together with proof of data correctness.

- **Registration Phase.** During this phase, the voter, v, personally consults a so-called supervised registrar (SR). This authority verifies that the voter is not being coerced directly and then generates a unique credential. This credential is then split into two parts: the long credential and the short credential, also referred to as the PIN.[5]

[5]Note that users are generally not good at choosing random PINs, as revealed in PIN frequency analyses. Thus, we recommend that the PIN be generated uniformly at random and not chosen by the voter.

Following that, the supervised registration authority develops the PIN's error list(as in 7.1) according to the election policy and determines the pin polynomial (as defined in 7.2). Finally, the voter's long credential and encryption of the pin-polynomial (encryption of polynomial coefficients) are transmitted to (stored on) the voter's supporting device. Notably, the PIN is not stored in the voting device or on the bulletin board, and it is necessary to be memorized by the voter .Along with the voter credential, the registrar provides proof that satisfies only voter, v, that all associated data was generated correctly, whether published on the bulletin board or stored on the voter's device.

- **Voting Phase.** In this phase, the voter device asks the voter to enter her PIN, $\hat{a}$ and her preferred choice, vote. Then it generates the ballot of the form:

$$\begin{aligned}\mathsf{ballot} = (\mathsf{CT}_{\mathsf{crd}}, \mathsf{CT}_{\mathsf{vote}}, \pi_{\mathsf{ballot}}) :& \\ &\mathsf{CT}_{\mathsf{crd}} = \mathsf{Enc}(\mathsf{voterCredential}) = (\mathsf{CT}_1, \mathsf{CT}_2), \\ &\mathsf{CT}_{\mathsf{vote}} = \mathsf{Enc}(\mathsf{vote})\end{aligned} \tag{7.6}$$

  together with some election's public parameters, such as the election identifier. In the ballot form, CT_1 and CT_2 denote the encryption of voter long credential and the pin she entered, with respect to the election public key, respectively. Moreover, π_{ballot} denotes the proof of the ciphertexts' well-formedness, which consists of proof of plaintext knowledge of both the credential and vote.. In the next step, voter can choose to either audit her ballot her vsd and conduct the *Benaloh Challenge* [35] procedure to ensure that her ballot has the correct vote and pin or to submit it.

  Device Auditing: If the voter wishes to audit option, the smart card commits to encryption of the ballot by displaying $\mathsf{hash}(\mathsf{ballot}; \{\mathsf{random}\})$. The voter notes down this hash, and if the encryption is challenged, the smart card releases all the randomness numbers $\{\mathsf{random}\}$ to the voter's computer. The voter can verify the hash indeed was consistent with the vote choice via a third device. This challenging procedure can be reiterated.

  Ballot Casting: When voters get sufficient confidence in the honesty of their device, they sign the ballot 7.6 and submits it via an authentic anonymous channel to the bulletin board.

- **Tally Phase.** This includes several steps. First, any exact duplicate ballots and ballots with incorrect proofs or signatures are removed. The teller should then detect duplicate ballots from a single voter, weeding the duplicates while ensuring the remaining ballot matches the valid pin (if existing). The weeding step is done according to the re-vote policy of the election. For example, suppose the re-vote policy specifies that each voter's last (according to the time-line) valid ballot must be counted in the tally phase. In that case, this step must ensure that the last valid ballot remains on the bulletin board and the rest are removed.

  The several mix-net servers carry out the following phase: each mix-net receives a list of ballots as input, re-encrypts each ballot, and then performs a secret permutation on the re-encrypted list to produce the new list. Finally, the decryption trustees process the output of the last mix-net to compute the election outcome.

We emphasize that the in-charge parties must provide proof of correctness in all of the preceding procedures.

- **Public Verification Phase.** In this phase, every participant, including the voters or external observers, can verify the correctness of the previous phases, particularly the correctness of all NIZKPs published during the setup, registration, voting, and tallying phase.

  In section 7.5.4, we propose three instantiation of our protocol. The main difference in these instantiations is how we perform the tally phase.

7.5.4 Protocol Instantiations

This section provides three instantiations of the generic **PIN-JCJ** protocol with concrete cryptographic primitives. While all three institutions adhere to the protocol framework 7.5.3, we emphasize that they differ in some steps, particularly the tally phase.

We will explain only the basic building blocks and their algorithm and suppress some details about ballot integrity and non-malleability from the zero-knowledge proofs, e.g. the inclusion of election identifiers and the correct form of the Fiat-Shamir transformations. Also, for simplicity, we describe the protocol with a single election trustee, but it is possible to run this protocol distributively. Finally, we also will not specify all parts of the distributed registration phase and the Benaloh challenges, this can; be implemented as in the NV12 scheme with some obvious modifications and with the repairs mentioned in [98].

Before proceeding, we will establish some specific notations for this chapter. First, we refer to the $\mathsf{ct}[m]$ as the cipher-text containing the message m, i.e. $\mathsf{ct} = \mathsf{Enc}(m)$, and by $\mathsf{ballot}[\mathsf{crd}, \mathsf{vote}]$, we mean a ballot that contains both the crd's and the vote's encryption.

7.6 Instantiation with Paillier Cryptosystem

The first instantiation relies on the standard threshold variant Paillier public-key cryptosystem[210] which its security is based on the hardness of the decisional composite residuosity assumption 3.

The main reason for choosing Paillier [216] instead of exponential ElGamal [150] (as in the original JCJ protocol) is that its plaintext-homomorphic property (see 2.3) allows us to evaluate the polynomial without decrypting the coefficients of the polynomials. Further, it allows an efficient multi-party computation protocol to compare and (hence sort) ciphertexts by plaintext values without decryption [192]. Furthermore, this algorithm is linear in the bit length, i.e. logarithmic in the security parameter, and can be made public verifiable [184]. Using this technique allows us to empower this instantiation to secure and efficient the weeding process, but at the cost of all ballots submitted with a voter identifier. Thus, obfuscating votes [136, 180] must also to be cast to achieve participation privacy.

We also use the secure multi-party computation protocol MPC_{min} to compare and (hence sort) ciphertexts by plaintext values without decryption introduced by Lipmaa and Toft [192] which secretly evaluates equality of two secret integers. In addition, the MPC protocol that is used in tally-hiding e-voting protocol such as Ordinos [184]. We refer to [184] for the public verifiable multi-party computation details.

Additional Note. We consider the protocol in the context of a single trusted party, namely a single election trustee running the key generation algorithm, rather than distributing trust among a group of election trustees; however, it can be implemented in a distributed fashion, as was already mentioned above, using the methods introduced in [210, 152]. Furthermore we emphasize that in the Paillier cryptosystem, proof of the correct key generation, π_{Kgen}, is actually the bi-primary test of the modulus n. Numerous methods for bi-primary testing have been proposed in single- and multi-party settings, including the method introduced by Boneh and Franklin in [52]. Additionally, Vitto [253] introduces a protocol for efficiently generating certificates of semi-primality, which enables us to generate distributively semi-primes with unknown factorization.

For the proof of knowledge of the vote and the well-formedness of the ballot, we use a non-interactive version of the sigma protocol for Paillier encryption scheme 3.3. Finally, we can use any EUF-CMA-secure signature scheme.

The detail of the protocol is as follows:

- ***Set Up Phase:*** The election authority run the key generation algorithm of Paillier encryption scheme, $\Pi^{\mathsf{Pai}}.\mathsf{Kgen}$ to generate the pair of keys and publish them on the bulletin board along with the election parameter:

$$\mathsf{key}_{\mathsf{election}} = (\mathsf{PK},\mathsf{SK}); \mathsf{PK} = \left(n = pq, \mathbb{G}, g, \mathsf{pp}_{\mathsf{election}}\right), \mathsf{SK} = (p,q)$$

- ***Registration Phase:*** At the beginning of this phase, an empty list of voters credential, VoterCredentialList is initiated. Then for voter $\mathtt{v_{id}}$ the registrar, does the following steps:

  1. Pick a random number, $\mathsf{crd} \xleftarrow{\$} \mathbb{Z}_n$. according to the uniform distribution such that $\mathsf{crd} \notin \mathsf{VoterCredentialList}$. For sake of privacy we can store a hash value of the credential in this list rather than the credential. Append the new credential crd to the VoterCredentialList and store it on the voter's device. We refer to crd as the voter's long credential.
  2. Pick randomly a number $a \xleftarrow{\$} \textbf{PinSpace}$ and declare it to the voter $\mathtt{v_{id}}$. We refer to a as voter's short credential or voter's pin.
  3. A registrar, based on the error-tolerance policy of the election, compute the set $\mathsf{ErrorList}_a$ as in (7.1)

$$\mathsf{poly}_{\mathtt{id},a} = \prod_{i=1}^{k} (x - \mathsf{crd} - a_i) = \sum_{i=0}^{k} p_i x^i \tag{7.7}$$

     We refer to the set $\mathsf{ErrorList}_a$ as the error list for **pin** = a and $\mathsf{poly}_{\mathtt{id},a}$ the pin-polynomial for voter id. If the voter's id is clear from the context we will omit the index id.
  4. Encrypt the polynomial with respect to the election public key. We stress that by encryption of the polynomial we mean the encryption of its coefficients, throughout this chapter. Namely, we consider the encryption as a vector with lent $k+1$.

$$\mathsf{Enc}(\mathsf{poly}_{(\mathtt{id},a)}) = \vec{\mathsf{CP}} = (\mathsf{cp}_0, \ldots, \mathsf{cp}_k) \,:\, \mathsf{cp}_i = \mathsf{Enc}(p_i) \text{ for } i = 0, \ldots k. \tag{7.8}$$

5. Provide a proof π_{poly} for the following relation:

$$\begin{aligned} \mathrm{R}_{\mathsf{poly}} = \Big\{ &(x, w), x = \big(\mathtt{id}, a, \mathsf{crd}, \mathsf{ErrorList}_a, \vec{\mathsf{CP}} = (\mathsf{cp}_i)_{i\in[k]}\big) \\ &w = \big((\mathsf{random}_i)_{i\in[k]}\big) : \\ &\mathsf{cp}_i = \mathsf{Enc}(\mathsf{PK}, p_i; \mathsf{random}_i) = g^{p_i} \cdot \mathsf{random}_i{}^n, \\ &\mathsf{poly}_{\mathtt{id},a} = \sum_{i=0}^{k} p_i x^i = \prod_{i=0}^{k} (x - \mathsf{crd} - a_i), \\ &i = 1, \dots, k : a_i \in \mathsf{ErrorList}\Big\} \end{aligned} \tag{7.9}$$

The registrant presents π_{poly} the voter as proof of the polynomial's validity (well-formedness and correctness). This proof is done in a designated way.

6. Store $(\mathsf{crd}, \mathsf{CP})$ on voter device.
7. Publish $\mathsf{v}_{\mathtt{id}} : \big(\mathsf{CP} = (\mathsf{cp}_0, \dots, \mathsf{cp}_k), \mathsf{Enc}(\mathsf{crd})\big)$ on bulletin board.

- ***Casting Ballot:*** In this step, every voter v_i decides to abstain from the election or vote for some candidate. In the latter scenario, the voter indicates her choice after entering her PIN, $\hat{a}$ to her vsd device. Then the vsd encrypts the vote, the voter credential, and the entered $\hat{a}$, under the public key of the election (Paillier public key) using random coins random_j, resulting in the following ciphertexts: $\mathsf{CP}^* = (\mathsf{cp}_0^*, \dots, \mathsf{cp}_k^*), \mathsf{ct}[\mathsf{vote}]$ and $\mathsf{ct}[\mathsf{crd}]$.

$$\begin{aligned} i \in [k] : \mathsf{cp}_i^* &= \mathsf{cp}_i^{(\hat{a}+\mathsf{crd})^i} \cdot \mathsf{random}_i^{*\,n} = (g^{p_i} \cdot \mathsf{random}_i^n)^{(\hat{a}+\mathsf{crd})^i} \cdot \mathsf{random}_i^{*\,n} \\ &= (g^{p_i \cdot (\hat{a}+\mathsf{crd})^i} \cdot \big(\mathsf{random}_i^{(\hat{a}+\mathsf{crd})^i} \cdot \mathsf{random}_i^*\big)^n \\ &= \mathsf{Enc}(\mathsf{PK}, p_i \cdot (\hat{a} + \mathsf{crd})^i) \\ \mathsf{ct}[\mathsf{vote}] &= \mathsf{Enc}(\mathsf{PK}, \mathsf{vote}; \mathsf{random}) = g^{\mathsf{vote}} \cdot \mathsf{random}^n, \\ \mathsf{ct}[\mathsf{crd}] &= \mathsf{Enc}(\mathsf{PK}, \mathsf{crd}) = g^{\mathsf{crd}} \cdot \mathsf{random}'^n \end{aligned} \tag{7.10}$$

As previously stated, the voter's vsd contains the voter's long credential (plain) but not the polynomial. In fact it only has the encryption of the polynomial. However, because of the homomorphic property of the Paillier encryption scheme, the vsd is capable of performing a blind multiplication of the polynomial coefficient with the $(\mathsf{crd} + \hat{a})$.

In the next step, vsd provides a proof, π_{ballot}, (also proof of knowledge), ensures that everything was generated honestly, for relation $\mathrm{R}_{\mathsf{ballot}}^{\mathsf{paillier}}$.

$$\begin{aligned} \mathrm{R}_{\mathsf{ballot}}^{\mathsf{paillier}} = \Big\{ &(x, w), x = \big(\mathsf{ct}[\mathsf{vote}], \mathsf{ct}[\mathsf{crd}], \vec{\mathsf{CP}} = (\mathsf{cp}_i)_{i\in[k]}, \vec{\mathsf{CP}}^* = (\mathsf{cp}_i^*)_{i\in[k]}\big) \\ &w = \big(\mathsf{vote}, \mathsf{random}_{\mathsf{vote}}, \hat{a}, \mathsf{crd}, \mathsf{random}_{\mathsf{crd}}, \{\mathsf{random}_i^*\}_{i\in[k]}\big) : \\ &\mathsf{ct}[\mathsf{vote}] = g^{\mathsf{vote}} \cdot \mathsf{random}^n, \mathsf{vote} \in \mathsf{cList}, \\ &\mathsf{ct}[\mathsf{crd}] = g^{\mathsf{crd}} \cdot \mathsf{random}'^n, \\ &i = 1, \dots, k : \mathsf{cp}_i^* = \mathsf{cp}^{(\mathsf{crd}+\hat{a})^i} \cdot \mathsf{random}_i^{*\,n}\Big\} \end{aligned} \tag{7.11}$$

This proof can be implemented efficiently using Sigma protocols and relies on the DDH assumption and it can be made non-interactive using the strong Fiat-Shamir heuristic.

At the end of this phase, voter signs her ballot of the following form:

$$\mathsf{ballot}_{\mathtt{v_{id}}} = (\mathsf{ct}[\mathsf{crd}], \mathsf{ct}[\mathsf{vote}], \vec{\mathsf{CP}}^* = (\mathsf{cp}_1^*, \ldots, \mathsf{cp}_k^*), \pi_{\mathsf{ballot}}) \tag{7.12}$$

and submits it to the bulletin board.

Re-Vote Policy In this instantiation, each voter is permitted to submit multiple ballots, and in the tally phase, one of the valid ballots is randomly picked and counted.

Additional Note. We stress that we do not describe some steps in detail here, such as the "Benaloh challenge" to audit the voter's supporting device. Furthermore, in this instantiation, we use the obfuscate method introduced in [136, 180] to protect the voter against the coercer who compels a voter not to vote (abstain from voting) i.e. a separate (distributed) authority will submit dummy ballots on behalf of all voters. Since this authority does not know the long credential all of these dummy ballots will be invalid but will obfuscate whether a specific voter did vote or not.

- ***Tally Phase:*** Using the Paillier encryption scheme allows us to efficiently sort the ciphertexts based on plaintext values without decrypting them using MPC among the Tally Tellers (see [192] for details). Additionally, this algorithm can be implemented in a multi-party computation, strengthen the elections. In our protocol, we take advantage of this multi-party computation to sort the encryption of polynomial evaluation.

  We refer by $\mathsf{MPC}_{\mathsf{min}}$ the algorithm that takes as input the ciphertexts:

  $$\mathsf{ct}_1 = \mathsf{Enc}(m_1), \mathsf{ct}_2 = \mathsf{Enc}(m_2), \ldots, \mathsf{ct}_t = \mathsf{Enc}(m_q),$$

  and outputs the index i^* such that

  $$\mathsf{ct}_{i^*} = \mathsf{Enc}(m_{i^*}) : m_{i^*} = \min\{m_1, \ldots, m_t\}.$$

  We use this algorithm in the Tally phase.

  1. *Ballot Validity Check:* We remove exact ballot duplicates and all ballots with invalid proof and the signature in the first step. In the next step, we need to remove extra ballots for each voter, making sure a valid ballot is kept if existing.
  2. *Weeding:* Since each voter will be associated with possibly more than one ballot, we need to weed them in a way that at least one of the ballots associated with the valid credential and pin will remain - if existing.

     Assume the voter $\mathtt{v_{id}}$ casts q ballots, each includes choices $\mathsf{vote}_{\mathtt{id},i}$:

     $$\mathtt{v_{id}} : \mathsf{ballot}_{\mathtt{id},1} : \left(\vec{\mathsf{CP}}_1^*, \mathsf{ct}[\mathsf{vote}_{\mathtt{id},1}]\right), \ldots, \mathsf{ballot}_{\mathtt{id},q}\left(\vec{\mathsf{CP}}_q^*, \mathsf{ct}[\mathsf{vote}_{\mathtt{id},q}]\right)$$

     each contains $\vec{\mathsf{CP}}_t^* = \left(\mathsf{cp}_{(t,1)}^*, \ldots, \mathsf{cp}_{(t,k)}^*\right)$ and the pin $\hat{a}_t$.

We homomorphically combine the public ciphertext cp_0 with the submitted encryptions to obtain an encrypted polynomial evaluation for each ballot:

$$\begin{aligned}
&\text{For } t = 1, \dots q: \\
&\mathsf{cp}_0 \cdot \prod_{j=1}^{k} \mathsf{cp}^*_{t,j} = g^{p_0} \cdot \mathsf{random}_0^n \cdot \prod_{i=0}^{k} g^{p_i \cdot (\hat{a}_t + \mathsf{crd})^i} \cdot \mathsf{random}_i^n \\
&\qquad = g^{\sum_{i=0}^{k} p_i (\mathsf{crd} + \hat{a}_t)^i} \cdot \mathsf{random} \\
&\qquad = \mathsf{Enc}(\mathsf{PK}, \mathsf{poly}_{\mathtt{id},a}(\mathsf{crd} + \hat{a}_t))
\end{aligned} \tag{7.13}$$

Note that the above ciphertext would be the encryption of zero if the ballot has a valid credential and pin.

$$\mathsf{Enc}(\mathsf{PK}, 0) \iff \hat{a}_t \in \mathsf{ErrorList}_a$$

We now verifiably mix the pairs:

$$\mathsf{ct}_t[\mathsf{poly}_{\mathtt{id},a}(\hat{a}_t + \mathsf{crd})], \mathsf{ct}[\mathsf{vote}_{\mathtt{id},t}]$$

and run the $\mathsf{MPC}_{\min}$ protocol:

$$\mathsf{MPC}_{\min}\Big(\mathsf{ct}_{\sigma(j)}[\mathsf{poly}_{\mathtt{id},a}(\hat{a}_{\sigma j} + \mathsf{crd})]\Big)_{j \in [q]} \mapsto \mathsf{ct}[\mathsf{poly}_{\mathtt{id},a}(\hat{a}_m + \mathsf{crd})] \tag{7.14}$$

which securely outputs the minimum value:

$$\forall t = 1, \dots, q : \mathsf{poly}_{\mathtt{id},a}(\mathsf{crd} + \hat{a}_m) \le \mathsf{poly}_{\mathtt{id}}(\mathsf{crd} + \hat{a}_t),$$

We keep this ciphertext and the corresponding encrypted vote and discard the rest. Note that $\mathsf{MPC}_{\min}$ selects a valid ballots having $\mathsf{poly}_{\mathtt{id},a}(\hat{a}_m + \mathsf{crd}) = 0$ if it exists.[6]

3. *Ballot Anonymization:* We observe that at this stage, there is only one ballot left on the bulletin board for each voter, which is composed of two components:

$$\begin{aligned}
\mathsf{v}_{\mathtt{id}} : \tilde{\mathsf{ballot}}_{\mathtt{id},m} = (\tilde{\mathsf{ct}}_{\mathtt{id}}, \tilde{\mathsf{ct}}'_{\mathtt{id}}) &: \\
&\tilde{\mathsf{ct}}_{\mathtt{id}} = \mathsf{ct}_{\mathtt{id}}[\mathsf{poly}] = \mathsf{Enc}(\mathsf{poly}_{\mathtt{id},a}(\hat{a}_m + \mathsf{crd})), \\
&\tilde{\mathsf{ct}}'_{\mathtt{id}} = \mathsf{ct}_{\mathtt{id}}[\mathsf{vote}_{\mathtt{id},m}].
\end{aligned}$$

In fact, $\mathsf{ballot}_{\mathtt{id},m}$ is the output of the MPC_{min} protocol. To avoid the heavy notation, we will no longer use the index m and we refer to $\mathsf{ballot}_{\mathtt{id},m}$ by $\mathsf{ballot}_{\mathtt{id}}$.

Two steps must be completed in order to anonymize the ballots:

Step 1. Raise the first component to the power r_1 and re-encrypt the second component:

$$\tilde{\mathsf{ct}}_{\mathtt{id}} \mapsto \mathsf{ct}_{\mathtt{id}} = \tilde{\mathsf{ct}}_{\mathtt{id}}^{\mathsf{r}_1} = \mathsf{Enc}\big(\mathsf{poly}_{\mathtt{id},a}(\hat{a}_m + \mathsf{crd})^{\mathsf{r}_1}\big) \tag{7.15}$$

[6]This will give a random correct vote. The "Last valid vote counts" policy can be implemented by adding the received order to the plaintext.

and provide a proof $\pi_{\mathsf{randomization}}$ for the relation $\mathrm{R}_{\mathsf{randomization}}$:

$$\begin{aligned}\mathrm{R}_{\mathsf{randomization}} = \Big\{&(x,w), x = (\widetilde{\mathsf{ballot}}, \mathsf{ballot}), w = \mathsf{r}_1 : \\ &\widetilde{\mathsf{ballot}} = (\tilde{\mathsf{ct}}, \mathsf{ct}'), \mathsf{ballot} = (\mathsf{ct}, \mathsf{ct}'), \\ &\mathsf{ct} = \tilde{\mathsf{ct}}^{\mathsf{r}_1} \\ &\Big\}\end{aligned} \tag{7.16}$$

We multiplicatively randomize the polynomial evaluation to avoid coercion side channels, but still be able to determine if a ballot was valid by checking if the value is zero. At the end of this step, we obtain the following list, which includes a single ballot for each voter:

$$\mathbf{L}_1 = \Big[\big(\mathsf{ct}_{\mathtt{id}} = \mathsf{ct}[\mathsf{poly}_{\mathtt{id},a}(\hat{a}_{\mathtt{id}} + \mathsf{crd}_{\mathtt{id}})], \mathsf{ct}'_{\mathtt{id}} = \mathsf{ct}[\mathsf{vote}_{\mathtt{id}}]\big)\Big]_{\mathtt{id}\in\mathsf{idSet}}$$

Step 2. In this step first we delete the voter identifier. Then we insert the list $\mathbf{L}_1$ into a verifiable parallel mix-net to apply the re-encryption and permutation procedure on the list $\mathbf{L}_1$. It is required that each mix-net provides proof of shuffle, π_{Shuffle}, for the relation, $\mathrm{R}_{\mathsf{shuffle}}$ (8.2). At the end of this step we obtain $(\mathbf{L}_2, \pi_{\mathsf{Shuffle}})$ where:

$$\mathbf{L}_2 = \Big[(\mathsf{ct}_{\sigma(\mathtt{id})}[\mathsf{poly}_{\sigma(\mathtt{id}),a}(\hat{a}_{\sigma(\mathtt{id})} + \mathsf{crd}_{\sigma(\mathtt{id})})^{\mathsf{random}}], \mathsf{ct}'[\mathsf{vote}_{\sigma(\mathtt{id})}])\Big]_{\mathtt{id}\in\mathsf{idSet}}$$

4. *Final PIN-Credential Check:* In this step, first we decrypt the first component of each ballot:
$$\mathsf{ct}_{\mathtt{id}}[\mathsf{poly}_{\sigma(\mathtt{id}),a}(\hat{a}_{\sigma(\mathtt{id})} + \mathsf{crd}_{\sigma(\mathtt{id})})^{\mathsf{random}}]$$
to retrieve the polynomial evaluation. All ballots with non-zero polynomial evaluation will be discarded.
5. *Vote Extraction:* Decrypt the remaining vote ciphertexts and provide proof of decryption 7.5.
6. *Result Computation:* Obtain the election result by applying the f_{res} over all plaintext votes.

Error tolerance property in this instantiation results from the correctness of the encryption scheme and the MPC_{min}. First, consider the following computation:

$$\left.\begin{aligned}7.8 &: \mathsf{cp}_i = g^{p_i} \cdot \mathsf{r}_i^n \\ 7.10 &: \mathsf{cp}_i^* = \mathsf{cp}_i^{(\hat{a}+\mathsf{crd})^i} \cdot \mathsf{r}_i^{*\,n}\end{aligned}\right\} \Rightarrow \mathsf{cp}_i^* = g^{(\hat{a}+\mathsf{crd})^i p_i} \cdot \mathsf{r}_i'^{\,n}$$

$$\begin{aligned}\Rightarrow \mathsf{ct}[\mathsf{poly}_{\mathtt{id},a}(\hat{a}_t + \mathsf{crd})] := \mathsf{cp}_0 \cdot \prod_{i=1}^{k} \mathsf{cp}_i^* \\ = g^{\sum_{i=0}^{n}(\hat{a}+\mathsf{crd})^i p_i} \cdot \mathsf{r}^n \\ = g^{\mathsf{Poly}_{\mathtt{id}}(\mathsf{crd}+\hat{a})} \cdot \mathsf{r}^n\end{aligned}$$

Due to the definition of **pin**-polynomial (7.2):

$$\mathsf{ct}[\mathsf{poly}_{\mathtt{id},a}(\hat{a}_t + \mathsf{crd})] = \mathsf{Enc}(0) \text{ if and only if } \mathsf{crd} +_{\hat{a}_t} \in \mathsf{ErrorList}_a + \mathsf{crd}_{\mathsf{registered}}$$

which with the overwhelming probability implies that $\hat{a} \in \mathsf{ErrorList}_a$. Additionally, the correctness property of the multi-party computation protocol $\mathsf{MPC}_{\min}$ in (7.14) guarantee that the ballot with a minimum value of $\mathsf{poly}_{\mathtt{id},a}(\hat{a}_t + \mathsf{crd})$ remains in the weeding step.

Note that $\mathsf{poly}_{\mathtt{id},a}$ has a range in non-negative integers. Therefore if there is any ballot with valid credentials and PIN, the output of $\mathsf{MPC}_{\min}$ will be a valid ballot, namely $\mathsf{ct}[\mathsf{poly}_{\mathtt{id},a}(\hat{a}_t + \mathsf{crd})] = \mathsf{Enc}(0)$.

Additional Note. The main advantage of this instantiation is sorting the ciphertexts without decrypting them (using $\mathsf{MPC}_{\min}$ protocol). Because this approach does not reveal whether any ballot has a valid PIN or not, thus sidestepping the attack on the standard duplicate removal.

7.7 Instantiation with BGN cryptosystem

The second instantiation is based on composite order groups introduced by [54] and the Groth-Sahai NIWI-proof system [143] with security based on the Subgroup decision assumption.

The main point of using those primitives in this instantiation are, BGN is a homomorphic encryption scheme supports a single multiplication which is what we need and also it can be efficiently implemented in a bilinear group. Having bilinear map allows us to do the polynomial evaluation in an efficient and secure way and also having the efficient NIWI-proof system.

Definition 47. ***BGN Cryptosystem*** *works as follows. Its key generation algorithm,* Kgen *outputs a pair of keys:* $\big(\mathsf{pk} = (n, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, h = g'^{q}), \mathsf{sk} = (p, q)\big)$ *where* $\mathbb{G} = \langle g \rangle$ *and* $\mathbb{G}_T$ *are two groups of order* n *and the secret key consists of two primes* p, q *such that* $n = pq$. $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ *is bilinear* $(\forall a, b \in \mathbb{Z}, g \in \mathbb{G} : \mathbf{e}(g^a, g^b) = \mathbf{e}(g, g)^{ab})$*, non-degenerate* $(\mathbb{G} = \langle g \rangle \Rightarrow \mathbf{e}(g, g) \neq 1_{\mathbb{G}_T})$ *and commutable map. A ciphertext on message* $m \in [T]$*, for* $T < q$ *has the form* $\mathsf{CT} = g^m h^{\mathsf{random}} \in \mathbb{G}$ *for some random number* random*. Decryption: raise the ciphertext to power* p *and compute the discrete log.*

The detailed of the protocol is as follows:

- ***Set Up Phase:***

  1. The election authority run the key generation algorithm of BGN encryption scheme, $\Pi^{\mathsf{BGN}}.\mathsf{Kgen}$ to generates the pair of keys:

  $$(\mathsf{sk}_{\mathsf{BGN}} = p, q, \mathsf{pk}_{\mathsf{BGN}} = (n, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, h)$$

  2. Then chooses a random group element $f \xleftarrow{\$} \mathbb{G}$. Note that $\mathbb{G} = \langle g \rangle$ is a cyclic group so there exists a unique integers z such that $f = g^z$.
  3. Set the election key as:

  $$\mathsf{PK}_{\mathsf{election}} = (n, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, h)\,,\ \mathsf{SK}_{\mathsf{election}} = (p, f)$$

  4. Publish the public key on the bulletin board along with the election parameter.

- ***Registration Phase:*** The registrar performs the following steps to generate the credential for the voter, $\mathtt{v_{id}}$

  1. Generate credential and PIN: crd, a as in the Paillier instantiation.
  2. Generate the list of errors and the pin-polynomial. Note that in this instantiation the polynomial $\mathsf{poly}_{\mathtt{id},a}$ only contains the polynomial, not the long credential:

  $$\mathsf{ErrorList}_a = \{a_1 = a, a_2, \ldots, a_k\}$$
  $$\mathsf{poly}_{\mathtt{id},a} = \prod_{i=1}^{k}(x - a_i) = \sum_{i=0}^{k} p_i x^i$$

  3. Run the encryption algorithm, $\Pi^{\mathsf{bgn}}.\mathsf{Enc}$ to obtain the following ciphertexts:

  $$\forall i \in [k]: \mathsf{cp}_i = \mathsf{Enc}(p_i) = g^{p_i} h^{\mathsf{random}_i},$$
  $$\mathsf{cp}_0 = g^{p_0} \cdot f^{\mathsf{crd}} h^{\mathsf{random}}$$
  $$= \mathsf{Enc}(p_0 + \mathsf{crd} \times z).$$

  Note that, technically cp_0 is the encryption of $p_0 + \mathsf{crd} \times z$. Although z is not a known value to any parties, the registrar can compute cp_0 without knowing its exact value.

  4. Generates the proof of validity of the polynomial $\mathsf{poly}_{\mathtt{id},a}$ and $(\mathsf{cp}_i)_{i=0,\ldots k}$, similar to the Paillier instantiation.
  5. Store

  $$\left(\mathsf{CP} = (\mathsf{cp}_0, \mathsf{cp}_1, \ldots, \mathsf{cp}_k), \mathsf{CRD} = g^{\mathsf{crd}}\right)$$

  in the user device.

  6. Publish $\mathtt{v_{id}}: \left(\mathsf{Enc}(\mathsf{crd}) = g^{\mathsf{crd}} \cdot h^{\mathsf{random}}, \mathsf{CP}\right)$ on bulletin board.

- ***Casting Ballot:*** During this phase, the voter can perform an audit on her device to ensure its accuracy. This process is similar to the Paillier instantiation 2.7.2.2, therefore we skip the details and only describe the ballot creation step.

  Voter, $\mathtt{v_{id}}$, indicates her choice after entering her PIN, $\hat{a}$ to her vsd device. Then the vsd encrypts the vote, the voter credential, and the entered $\hat{a}$, under the public key of the election(See 3) using random coins random_j, resulting in the following ciphertexts:

  1. Compute:

  $$\mathsf{CT}_{\mathsf{vote}} = \mathsf{Enc}(\mathsf{vote}), \mathsf{CT}_{\mathsf{crd}} = \mathsf{Enc}(\mathsf{crd}) = \mathsf{CRD} \cdot h^{\mathsf{random}}$$

  2. Encrypt the PIN:

  $$\text{For } i = 1, \ldots, k : \mathsf{ca}_i = \mathsf{Enc}(\hat{a}^i).$$

  3. Re-randomize the polynomial coefficients:

  $$\text{For } i: 0, 1, \ldots, k: \mathsf{cp}_i^* = \mathsf{cp}_i \cdot h^{\mathsf{random}_i^*}$$

4. Set

$$\begin{aligned}\mathsf{CA} &= (\mathsf{ca}_1, \ldots, \mathsf{ca}_k),\\ \mathsf{CP}^* &= (\mathsf{cp}_0^*, \ldots, \mathsf{cp}_k^*)\end{aligned}$$

and provide a proof (proof of knowledge), π_{ballot} for the following relation, including a joint proof of plaintext-knowledge for all the other ciphertexts in the ballot and include the rest of the ballot in the hash for non-malleability. This proof can be generated using the Groth-Sahai technique. (See Appendix 9.6)

$$\begin{aligned}\mathrm{R}^{\mathsf{bgn}}_{\mathsf{ballot}} = \Big\{(x, w) :\\ & x = \big(\mathsf{pp}_{\mathsf{election}}, \mathsf{CT}_{\mathsf{vote}}, \mathsf{CT}_{\mathsf{crd}}, \mathsf{CA}\big),\\ & w = \big(\mathsf{vote}, \mathsf{random}_{\mathsf{vote}}, \mathsf{CRD}, \mathsf{r}_{\mathsf{crd}}, \hat{a}, \{\mathsf{r}_i\}_{i\in[k]}\big) :\\ & \mathsf{CT}_{\mathsf{vote}} = g^{\mathsf{vote}} \cdot h^{\mathsf{random}_{\mathsf{vote}}},\\ & \mathsf{vote} \in \texttt{List of candidats},\\ & \mathsf{CT}_{\mathsf{crd}} = \mathsf{CRD} \cdot h^{\mathsf{r}_{\mathsf{crd}}},\\ & \{\mathsf{ca}_i = g^{(\hat{a})^i} \cdot h^{\mathsf{r}_i}\}_{i\in[k]}\\ \Big\}\end{aligned} \tag{7.17}$$

5. At the end of this phase, the voter signs her ballot of the following form:

$$\mathsf{ballot} = (\mathsf{CT}_{\mathsf{vote}}, \mathsf{CT}_{\mathsf{CRD}}, \mathsf{CA}, \mathsf{CP}^*, \pi_{\mathsf{ballot}})$$

and submits it to the bulletin board.

Re-Vote Policy. In this instantiation, each voter is permitted to submit multiple ballots, and in the tally phase, unlike to Paillier instantiation, it is possible to consider the last valid ballot cast by the voter.

1. *Ballot Validity check:* In the first step, we remove exact ballot duplicates and then we check the validity of the proofs and the signatures. In case any of any failure, the ballot will be discarded. In the next step we need to remove extra ballots for each voter, making sure a valid ballot is kept, if existing.
2. *Polynomial Evaluation:* Compute the encrypted polynomial evaluation as follows:

$$\begin{aligned}\mathsf{Enc}_T(t) &= \mathbf{e}(\mathsf{CT}_{\mathsf{crd}}, f)^{-1} \cdot \mathbf{e}(\mathsf{cp}_0^*, g) \cdot \mathbf{e}(\mathsf{cp}_1^*, CA_1) \cdots \mathbf{e}(\mathsf{cp}_k^*, CA_k)\\ &= \mathsf{Enc}_T(\mathbf{e}(g^{\mathsf{poly}_{\mathtt{id},a,g}(\hat{a}_{\mathtt{id}})}, g))\end{aligned} \tag{7.18}$$

We refer to this value by $\mathsf{Enc}_T(t)$ with t being the polynomial evaluation which can be seen as an encryption in the target space (group $\mathbb{G}_T$).
3. *Mixing:* Now verifiably mix the tuples $(\mathsf{CT}_{\mathsf{crd}}, \mathsf{CT}_{\mathsf{vote}}, \mathsf{Enc}_T(t))$.
4. For each ballot we create $\mathsf{Enc}_T(\mathsf{crd} + t)$:

$$\begin{aligned}\mathbf{e}(\mathsf{CT}_{\mathsf{crd}}, g) \cdot \mathsf{Enc}_T(t) &= \mathsf{Enc}_T(\mathbf{e}(g^{\mathsf{crd}}, g)) \cdot \mathsf{Enc}(\mathbf{e}(g^{\mathsf{poly}_{\mathtt{id},a}(\hat{a})}, g))\\ &= \mathsf{Enc}_T(\mathbf{e}(g^{\mathsf{crd}+\mathsf{poly}_{\mathtt{id},a}(\hat{a})}, g))\end{aligned}$$

Now, we need to remove ballots having the same value $\mathbf{e}(g^{\mathsf{crd}+\mathsf{poly}_{\mathtt{id},a}(\hat{a})}, g)$, which basically means same credential and same $\hat{a}$. This can be done by running a PET or using some hash function. At the end of this step we have a list of ballots of the following form:

$$\left(\mathsf{Enc}_T(\mathbf{e}(g^{\mathsf{crd}+\mathsf{poly}_{\mathtt{id},a}(\hat{a})}, g)), \mathsf{CT}_{\mathsf{vote}}\right) \tag{7.19}$$

Additional Note. If we have a small number of voters, it is possible (efficient) to do this via PETs. Further we can mix between each duplicate removal. For a larger number we suggest to split the board in two, remove duplicates separately, then mix and do duplicate removal again. This will decrease the information from the distribution of confirmed duplicates to a coercer carrying out the *leaky duplicate removal attack* mentioned in Sec. 7.3. Additionally for the sake of privacy we can replace the first component in (7.19) with $\mathsf{hash}_{\mathsf{hk}}(\mathbf{e}(g^{\mathsf{crd}+\mathsf{poly}_{\mathtt{id},a}(\hat{a})}, g)(\hat{a}))$ where $\mathsf{hash}_{\mathsf{key}}$ is a keyed-hash function with secret key hk.

5. We now need to select eligible valid votes. We mix the above list and the list of registered encrypted credential. To this end, compare $\mathsf{crd} + \mathsf{poly}_{(\mathtt{id},a)}(\hat{a})$ to the list of registered credentials. Any ballot that has the match with a registered credential would be a ballot with valid credential where its polynomial evaluation is equal to zero with overwhelming probability. There is only a negligible chance of that invalid crd and invalid PIN generate an invalid full credential. Next, we do a further PET against the short credentials. This will reveal malicious authorities creating valid polynomial on their own. If this is positive too, we decrypt the vote and continue to the next registered credential.

Error tolerance property. The following computation shows the correctness of equation 7.18, in other words it shows how to evaluate the polynomial on the input value $\hat{a}$:

$$\begin{aligned}
&\mathbf{e}(\mathsf{CT}_{\mathsf{crd}}, f)^{-1} \cdot \mathbf{e}(\mathsf{cp}_0^*, g) \cdot \mathbf{e}(\mathsf{cp}_1^*, CA_1) \cdots \mathbf{e}(\mathsf{cp}_k^*, CA_k) = \\
&\mathbf{e}(\mathsf{CRD} \cdot h^{\mathsf{r}}, f)^{-1} \cdot \mathbf{e}(g^{p_0}(f)^{\mathsf{crd}} h^{\mathsf{r}_0}, g) \cdot \mathbf{e}(g^{p_1} h^{r_1}, g^{\alpha_i} h^{\gamma_i}) \cdot \ldots \mathbf{e}(g^{p_k} h^{\mathsf{r}_k}, g^{\alpha_k} h^{\gamma_k}) = \\
&\mathbf{e}(\mathsf{CRD}, f)^{-1} \cdot \mathbf{e}(h, f)^{-\mathsf{r}} \mathbf{e}(g^{p_0} f^{\mathsf{crd}} h^{\mathsf{r}_0}, g) \cdot \mathbf{e}(g^{p_1} h^{\mathsf{r}_1}, g^{a^i} h^{\gamma_i}) \cdot \ldots \mathbf{e}(g^{p_k} h^{\mathsf{r}_k}, g^{a^k} h^{\gamma_k}) = \\
&\mathbf{e}(\mathsf{CRD}, f)^{-1} \mathbf{e}(f, h^{\mathsf{r}}) \mathbf{e}(f, \mathsf{CRD}) \mathbf{e}(g^{p_0} h^{\mathsf{r}_0}, g) \cdot \mathbf{e}(g^{p_1} h^{\mathsf{r}_1}, g^{a^i} h^{\gamma_i}) \cdot \ldots \mathbf{e}(g^{p_k} h^{\mathsf{r}_k}, g^{a^k} h^{\gamma_k}) = \\
&\mathbf{e}(h^{\mathsf{r}}, f)(\prod_{i=0}^{k} \mathbf{e}(g^{p_i}, g^{\alpha_i})) \cdot (\prod_{i=0}^{k} \mathbf{e}(g^{p_i}, h^{\gamma_i})) \cdot (\prod_{i=0}^{k} \mathbf{e}(g^{\alpha_i}, h^{\mathsf{r}_i}))(\prod_{i=0}^{k} \mathbf{e}(h^{\gamma_i}, h^{\mathsf{r}_i})) \\
&\mathbf{e}(g, g^{\sum_{i=0}^{k} p_i \alpha_i})) \cdot \mathbf{e}(g, h^{\mathsf{r}}) = \mathbf{e}(g, g^{\mathsf{poly}_a(\hat{a})}) \cdot \mathbf{e}(g, h^{\mathsf{r}})
\end{aligned}$$

As a result, if we raise the above term to the power p, the result is 1 if and only if $\mathsf{poly}_{\mathtt{id},a}(\hat{a}) = 0$. Also, due to the secret f and zero-knowledge proof, π_{ballot}, malicious voters cannot construct a fake polynomial resulting in zero-evaluation.

Additional Note. The main advantage of this instantiation is that voters can vote anonymously; thus, unlike the first instantiation, no identity obfuscation is required. Additionally, by utilizing Groth-Sahai proof techniques, the implementation becomes more efficient. Particularly in an election with many candidates and voters, compared to the sigma protocol, the proof of membership will be more effective. Moreover, we can implement the mix-net components using the Groth-Bayer mix-net servers [28].

7.8 Instantiation with Functional Encryption Scheme

Our third instantiation[7] relies on the functional encryption scheme, precisely, the inner product encryption scheme 4. The primary advantage of this approach is that it simplifies the protocol during the registration and tally phases.

Recall the polynomial evaluation of an encrypted value x. The polynomial can be considered as a vector of its coefficient

$$\mathsf{poly}_{\mathbf{pin}} = p_0 + p_1 x + \ldots + p_k x^k \approx \vec{P} = (p_0, \ldots, p_k)$$

and the point x can be represented as the vector $\vec{x}$:

$$\vec{x} = (1, x, \ldots, x^k).$$

Then the evaluation of $\mathsf{poly}(x)$ is obtained by computing the inner product of these two vectors:

$$\mathsf{poly}_{\mathbf{pin}}(x) = \langle \vec{P}, \vec{x} \rangle$$

During the registration phase of the first two protocol instantiations, a registrant generates a valid credential and a valid pin, retrieves the polynomial, and stores the polynomial's encryption on the voter device. However, anyone can carry out the procedure because all encryption steps are performed with respect to some public-key cryptosystem. To be specific, anyone can generate the credential and pin and store the encrypted data on the voter device. In other words, for the sake of eligibility and privacy, it is necessary that the voter credential, her **pin**, and the encryption of her vote are all linked together and cannot be generated by someone who does not know some secret value. This aspect complicates the tally process because it is critical to match the credentials of all ballots to the registered (legitimate) voters.Using the functional encryption scheme, we may consider a different scenario.

Motivation. Consider the polynomial as a function. In the context of the functional encryption scheme, while everyone can encrypt the polynomial, only the owner of the master secret key can generate the polynomial's token. Now, rather than the encryption of the polynomial, its secret token is stored on the voter device. This ensures that the polynomial evaluation is zero if and only if the ballot was cast by a legal (registered) voter with a high probability. As a result, this simplifies the protocol, particularly during the tally step.

Another motivation for using the IPE is that the voter's token generated by the election trustees is essential in the verification phase, both for individual and universal verifiability. It allows the voter to trace her ballot, and at the tally phase, it can be substituted with proof of the decryption procedure's correctness.

Cryptographic Primitives In this instantiation we have all the primitives listed in 7.5.2 except that we replace the public key cryptosystem with the inner product encryption scheme.

[7]This instantiation is not included in our publication [98]

We consider the inner product encryption scheme described in section 4.6:

$$\Pi^{\mathsf{IP}}(k, 1^{\ell}) = \langle \mathsf{IP.SetUp}, \mathsf{IP.TokGen}, \mathsf{IP.Enc}, \mathsf{IP.Dec} \rangle$$

for vector space $\mathbb{Z}_p^k$, where $k-1 = \left|\mathsf{ErrorList}_{\mathbf{pin}}\right|$ is the number of errors tolerated by the protocol. Further by $\Pi^{\mathsf{GS}} = \langle \mathsf{Prove}, \mathsf{Verify} \rangle$ we refer to a non-interactive witness-indistinguishable proof technique presented in [143].

Considering the notation form 4.6 the protocol is detailed as follows:

- **Setup Phase.** This procedure is conducted by the election authority and election trustees distributively in the following steps:
  1. First run the setup algorithm of inner product encryption scheme, Algorithm 4.6, to generate $(\mathsf{msk}_{\mathsf{IP}}, \mathsf{mpk}_{\mathsf{IP}})$.
  2. For $i \in [k], b \in [2]$ generate random integers $f'_{i,b}, h'_{i,b} \leftarrow \mathbb{Z}_p^*$ to obtain:

     $$F'_{i,b} = f'_{i,b}, H'_{i,b} = h'_{i,b}, i = 1, \ldots, k, b = 1, 2$$

     These extra random numbers are needed to randomize the voter's token.
  3. To prove the correct construction of $\mathsf{MPK}_{\mathsf{election}}$, we may use the verification algorithm for the IPE scheme described in 5.6.1. This proof should be published.
  4. Output $\mathsf{Key}_{\mathsf{electionelection}} = (\mathsf{MSK}_{\mathsf{election}}, \mathsf{MPK}_{\mathsf{election}})$ where:

     $$\mathsf{MPK}_{\mathsf{election}} = \left((p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}), \mathsf{mpk}_{\mathsf{IP}}, \mathsf{pk}_{\Pi}, \{F'_{i,b}, H'_{i,b}\}_{i,b}\right),$$
     $$\mathsf{MSK}_{\mathsf{election}} = \left(\mathsf{msk}_{\mathsf{IP}}, \mathsf{sk}_{\Pi}, \{f'_{i,b}, h'_{i,b}\}_{i,b}\right)$$

     respectively are the master public key and the master secret key of the election.

- **Registration Phase:** At the beginning of this phase an empty list of voters credential
  VoterCredentialList is initiated. Then for voter $\mathtt{v_{id}}$ the registrar, does the following steps:
  1. Generating the long credential: Same as Paillier instantiation.
  2. Generating PIN, the short credential: Pick randomly a number $a \xleftarrow{\$} \mathbf{PinSpace}$ and declare it to the voter $\mathtt{v_{id}}$.
  3. Compute the error list for a based on the election policy: A registrar based on the error-tolerance policy of the election obtain the set $\mathsf{ErrorList}_a$ as (7.1)and then set the polynomial as follows:

     $$\mathsf{poly}_{\hat{a}} = \prod_{i=1}^{k}(x - \mathsf{crd} - a_i) = \sum_{i=0}^{k} p_i x^i \implies \vec{\mathsf{poly}}_a = (p_0, \ldots, p_k)$$

  4. Run the encryption algorithm $\mathsf{IP.Enc}$ and encrypt the credential of the voter with respect to the polynomial vector $\vec{\mathsf{poly}}_a = (p_0, \ldots, p_k)$:

     $$\mathsf{Enc}(\mathsf{MPK}_{\mathsf{IP}}, \mathsf{poly}_a, \mathsf{crd}) = \mathsf{CT}[\mathsf{poly}_a, \mathsf{crd}]$$

5. Run the token generation algorithm $\mathsf{IP.TokGen}$ and generate the token for the vector $\vec{\mathsf{poly}}$ to obtain the following:

$$\begin{aligned}&\mathsf{Tok}_a = \mathsf{Tok}_{\vec{\mathsf{poly}}_a} \leftarrow \mathsf{IP.TokGen}(\mathsf{MSK}, \mathsf{poly}_a)\\ &\mathsf{Tok}_a = \big(K_A, K_B, \{K_{1,i}, K_{2,i}, K_{3,i}.K_{4,i}\}\big)_{i\in[k]},\\ &\mathsf{partial.Tok}_a\{K_B, K^*_{1,i} = K_{1,i}^{f'_{1,i}}, K^*_{2,i} = K_{2,i}^{f'_{2,i}}, K^*_{3,i} = K_{3,i}^{h'_{1,i}}, K^*_{4,i} = K_{4,i}^{h'_{2,i}}\}\end{aligned}$$

6. Send $\mathsf{partial.Tok}_a, \mathsf{crd}$ to the voter. We do not send all components of the token to the voter device because having the entire token enables the voter (and coercer) to retrieve PIN and the vote. This raises a serious issue if the coercer also has access to the voting account. Namely, the coercer can attempt several pins, encrypt some random inputs with respect to the pin. Then, by comparing the decryption's output to the encryption inputs, the coercer can easily identify which pins are valid and which are not.
7. Provide a proof π_{poly} for the following relation:

$$\begin{aligned}\mathrm{R}_{\mathsf{poly}} = \Big\{(x, w) :\\ &x = \big(\mathtt{id}, a, \mathsf{crd}, \mathsf{ErrorList}_a, \mathsf{Tok}^*_a\big)\\ &w = \big((f'_{i,b}, h'_{i,b})_{i\in[k]}, (\mathsf{r}_i)_{i\in[k]}\big) :\\ &\big(K_A, K_B, \{K_{1,i}, K_{2,i}, K_{3,i}.K_{4,i}\}\big)_{i\in[k]} = \mathsf{IP.TokGen}(\mathsf{MSK}_{\mathsf{IP}}, \mathsf{poly}_a; \mathsf{r}_i)\\ &\wedge\\ &\mathsf{partial.Tk} = \{K_B, K^*_{1,i} = K_{1,i}^{f'_{1,i}}, K^*_{2,i} = K_{2,i}^{f'_{2,i}}, K^*_{3,i} = K_{3,i}^{h'_{1,i}}, K^*_{4,i} = K_{4,i}^{h'_{2,i}}\}\\ &\Big\}\end{aligned}$$

The above proof can be done using the algorithm 5.6.2 and π_{poly} is sent to the voter and verified by the voter.

8. Publish $\mathsf{IP.Enc}\Big(\mathsf{MPK}, \vec{\mathsf{poly}}_a, \mathsf{crd})\Big)$ on the bulletin board. Note that here we encrypt the credential with respect to polynomial vector. This enables us to generate the token related to the pin for the voter, which can be considered her verification key.

- **Casting Ballot:** In this step, the voter $\mathsf{v}_{\mathtt{id}}$ indicates her choice by entering her pin $\hat{a}$ on her device. Then the vsd encrypts the vote, the voter credential, with respect to the vector

$$\vec{\mathsf{CRD}}_{\hat{a}} = \big(1, (\mathsf{crd} + \hat{a}), \ldots, (\mathsf{crd} + \hat{a})^k\big) \tag{7.20}$$

under the master public of the election using random coins random_j, resulting in the following ciphertexts:

$$\begin{aligned}\mathsf{CT}_{\mathsf{vote}} &= \mathsf{IP.Enc}(\mathsf{mpk}_{\mathbf{IP}}, \big(1, (\mathsf{crd} + \hat{a}), \ldots, (\mathsf{crd} + \hat{a})^k\big), \mathsf{vote}; \mathsf{random}),\\ \mathsf{CT}_{\mathsf{crd}} &= \mathsf{IP.Enc}(\mathsf{mpk}_{\mathbf{IP}}, \big(1, (\mathsf{crd} + \hat{a}), \ldots, (\mathsf{crd} + \hat{a})^k\big), \mathsf{crd}; \mathsf{random})\end{aligned}$$

It also re-encrypt Tok^* as follows:

$$\mathsf{partial.Tok}_{\mathbf{pin}} = \left(K_B^{\mathsf{random}}, \{K_{1,i}^{\mathsf{random}}, K_{2,i}^{\mathsf{random}}, K_{3,i}^{\mathsf{random}}.K_{4,i}^{\mathsf{random}}\}\right)_{i\in[k]}$$

Notice that the IPE scheme in 4.6 allows re-encrypting without knowing the plaintext.

In the next step, vsd provides a proof, π_{ballot}, (also proof of knowledge) for the following relation:

$$\begin{aligned}\mathrm{R}^{\mathsf{ip}}_{\mathsf{ballot}} = \Big\{(x,w) :\ & \\ & x = \left(\mathsf{CT}_{\mathsf{vote}}[\mathsf{crd}+\hat{a}], \mathsf{CT}_{\mathsf{crd}}[\mathsf{crd}+\hat{a}], \mathsf{re.Enc}(\mathsf{Tok}^*)\right)\\ & w = \left(\mathsf{crd}, \hat{a}, \mathsf{vote}, \{\mathsf{random}\}\right) :\\ & \mathsf{CT}_{\mathsf{vote}} = \mathsf{IP.Enc}(\mathsf{MPK}, \vec{\mathsf{CRD}}_{\hat{a}}, \mathsf{vote}; \mathsf{random})\\ & \mathsf{CT}_{\mathsf{crd}} = \mathsf{IP.Enc}(\mathsf{MPK}, \vec{\mathsf{CRD}}_{\hat{a}}, \mathsf{crd}; \mathsf{random})\\ & \Big\}\end{aligned}$$

We can implement this proof by using the Groth-Sahai proof technique (See 3.10.3).

The VSD submits $\mathsf{ballot}_{\mathtt{id}}$ as voter's ballot to the election server on an authenticated channel. $\mathsf{ballot} = (\mathsf{CT}_{\mathsf{crd}}, \mathsf{CT}_{\mathsf{vote}}, \mathsf{Tok}^* = (\mathsf{cp}_1^*, \ldots, \mathsf{cp}_k^*), \pi)$

The Benaloh challenge is done the same as Paillier instantiation 7.6.

- **Tally Phase:**

  1. We remove exact ballot copies and all ballots with invalid proof and the signature In the first step.
  2. Since each voter will be associated with possibly more than one ballot, we need to weed them. We make sure a valid ballot is chosen - if existing.
  3. We run a mix-net to create a secret re-encryption shuffle of the remaining ballots.
  4. In this step we have a list of ballots in the form of(see (7.20))

     $$\left(\mathsf{CT}[\vec{\mathsf{CRD}}_{\hat{a}}, \mathsf{crd}], \mathsf{CT}[\vec{\mathsf{CRD}}_{\hat{a}}, \mathsf{vote}], \mathsf{partial.Tok}_{\mathsf{a}}\right)$$

     $$K_A = \prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} \cdot K_{4,i}^{-f_{2,i}} \cdot K_{5,i}^{-h_{1,i}} \cdot K_{6,i}^{-h_{2,i}}$$

     For each ballot compute the following:

     $$\mathsf{IP.Dec}(\mathsf{Tok}_{\mathbf{pin}}, \mathsf{CT}[\vec{\mathsf{CRD}}_{\hat{a}}, \mathsf{crd}]) = \mathbf{e}(g,h)^{\mathsf{random}\cdot\mathsf{poly}(\mathsf{crd}+\hat{a})} \cdot \mathbf{e}(g,g)^{\mathsf{crd}}.$$

     And they remove all ballots with the same value except one.
  5. Run another mix-net and anonymize the remaining ballots.
  6. In this step we have a list of ballots in the form of $(\mathsf{CT}[\mathsf{vote}, \hat{a}], \mathsf{Tok}^*)$; in this step, the election trustee compute the last component of the token for each

ballot, add the following component to each ballot:

$$K_A = \prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} \cdot K_{4,i}^{-f_{2,i}} \cdot K_{5,i}^{-h_{1,i}} \cdot K_{6,i}^{-h_{2,i}}$$

and decrypt all ballots. The output of the decryption algorithm will be a valid vote if the pin is valid; otherwise, it gives us a random string.

Correctness and Error Tolerance Property. The correctness of the protocol and its error-tolerance property are the easy corollaries of the correctness of the inner product encryption scheme.

7.9 Security Analysis

In this part, we conduct a security analysis on our scheme by providing the security model, stating the security criteria we wish to guarantee, and determining the security assumptions needed to support them.

Furthermore, we demonstrate that the presence of the underlying assumption guarantees that our protocol has the required feature. We here focus on the instantiations with Paillier encryption scheme.Following the definition 6.2.1, we formally describe our protocol by

$$\Pi^{\mathsf{pinJCJ}} = \langle n_{\mathsf{v}}, \mathsf{Agent}, \Pi^{\mathsf{h}}, \mathsf{cList}, \Gamma^{\mathsf{elc}}, f_{\mathsf{res}} \rangle$$

where,

- n_{v} is the number of voters (and their voter supporting devices vsd),
- Agent is the set of participants including honest and dishonest voters, scheduler, $\mathfrak{s}$, election trustees, $\mathfrak{T}$, registrars, $\mathfrak{R}$, judges, $\mathfrak{J}$, the bulletin board, $\mathfrak{BB}$. We consider the adversary and the coercer as part of the set Agent.
- cList denotes the set of choices or the list of candidates.
- Γ^{elc} denotes a probability distribution on the set of vote choices.
- Π^{h} is the set of algorithms that are defined by the protocol description.
- f_{res} is the result function.

7.9.1 Security Model

Our protocol aims to preserve the JCJ-based protocol's security model. Specifically, we assume the trust assumptions made in [207].

1. The adversary is computationally bounded.
2. In case of using the threshold cryptosystem, we require that the majority of the election trustee and also the majority of the registrarsare trustworthy. Namely, the adversary cannot corrupt a threshold set of election trustee. By using single party, we assume the election trustee and the registrar are honest.
3. At least one mix server is honest.
4. There is a point in the voting phase, where the adversary cannot control the voter.

5. Voter's vsd does not leak the voter's private credentials to the adversary.

6. The adversary cannot control the voter's computer or voter's device. However, considering the Benaloh challenges in the protocol, this assumption will reduce: The adversary cannot simultaneously control the voting and the verification environment[206].

7. The channel to the ballot boxes is anonymous.

In this section, we formally establish the level of verifiability, privacy, and coercion resistance our protocol provides. We demonstrate that **PIN-JCJ** is more resistant to coercion than the original JCJ protocol and smart-card based JCJ. We use the general notion of E2Everifiability described in 6.3.1 for verifiability analysis and the bPRIV notion of security proposed in [46], presented in Section 6.4.1 for privacy analysis.

7.9.2 Privacy Proof

Now we analyze the privacy of the **PIN-JCJ** protocol, instantiated with Paillier cryptosystem, sigma protocol, π^{σ}, and multiparty computation protocol, MPC_{min}.

To this end, we adapt the definition, notion, and notation presented in section 6.4 based on the paper [46]. In this paper, the authors propose a new game-based definition of privacy called bPRIV. Additionally, they identify new properties, strong consistency(see Section 6.4.3 above), and strong correctness (see Section 6.4.3 above). In this section we first prove our protocol satisfies all three properties and then based on theorem in [46] we conclude the simulation privacy of our protocol.

It is worth noting that the one reason we use this setting to establish our protocol's privacy level is that our protocol heavily relies on the re-voting policy, and we need to demonstrate that this re-voting policy has no effect on the ballot's privacy.

Assumptions: To prove the bPRIV-privacy of **PIN-JCJ**, we make the following assumptions about the cryptographic primitives we use: (see also Section 7.6)

A.1 In this case the underlying public key encryption scheme, Paillier cryptosystem (BGN and IPE in other instantiations), is IND-CPA-secure. Considering the theorem 2.7.2, this assumption will reduce to the hardness of residue problem. 3

A.2 The signature scheme is EUF-CMA-secure (See Definition 15).

A.3 The multi-party computation protocol, MPC_{min}, realizes an ideal functionality. This essentially means that it takes a vector of ciphertexts and returns the ciphertext related to the plaintext with the minimum value without any leak.

A.4 All the proofs, $\pi_{\mathsf{Kgen}}, \pi_{\mathsf{Shuffle}}, \pi_{\mathsf{Dec}}$ and π_{Enc} are Zero-Knowledge simulation sound extractable with respect to expected polynomial-time adversaries.

A.5 The scheduler $\mathfrak{s}$, the bulletin board, $\mathfrak{BB}$, the majority of election trustees (in a distributed setting) and all registrars ($\mathsf{r} \in \mathfrak{R}$) are honest. In the following we assume $\lceil \frac{n_{\mathsf{t}}}{2} \rceil < t$ and $\mathsf{a}_{i_j} \in \mathfrak{T}$ which the set of election trustees contains n_{t} trusted parties involved in generating the secret key of the election(distributively). Following from definition (6.1) we formulate the above assumption in:

$$\varphi = \mathsf{hon}(\mathfrak{s}) \wedge \Big[\bigvee_{\substack{i_1<\ldots<i_t \\ i_j \in [n_{\mathsf{t}}]}} (\mathsf{hon}(\mathsf{a}_{i_1}) \wedge \ldots \wedge \mathsf{hon}(\mathsf{a}_{i_t}))\Big] \wedge \Big[\bigwedge_{\mathsf{r} \in \mathfrak{R}} (\mathsf{hon}(\mathsf{r}))\Big] \wedge \mathsf{hon}(\mathfrak{BB})$$

To avoid future heavy notation, we presume a single party, resulting in the following formula:

$$\varphi = \mathsf{hon}(\mathfrak{s}) \wedge \mathsf{hon}(\mathfrak{T}) \wedge \mathsf{hon}(\mathfrak{R}) \wedge \mathsf{hon}(\mathfrak{BB})$$

The bPRIV definition is only considers a single authority holding the secret key for simplicity and we will follow this approach here. Note that this is especially simplifying for the $\mathsf{MPC}_{\mathsf{min}}$ protocol, where we simply assume that the election authority decrypts privately to find the minimal plaintext value and provides a Zero-Knowledge proof of correctness for public validation.

A.6 The polynomial evaluation process is correct regarding the error-list. Namely, the zero set of polynomial only contains the error list described in the election policy document except with negligible probability.

Theorem 7.9.1. *Considering the assumption* **[A.1-5]**, $\Pi^{\mathsf{pinJCJ}} = \langle n_{\mathsf{v}}, \mathsf{Agent}, \Pi^{\mathsf{h}}, \mathsf{cList}, \Gamma^{\mathsf{elc}}, f_{\mathsf{res}}\rangle$ *has* bPRIV *property (See 44) in the presence of any PPT adversary.*

Here we present a proof sketch of the above theorem and we are not going through the details.

Proof sketch.[8]

In order to prove the theorem, namely to prove that any PPT adversary $\mathcal{A}$ distinguishes the two experiments (See 6.1), $\mathsf{Exp}^{\mathsf{bPriv}}_{\mathcal{A}}(1^{\ell})[\beta]$ for $\beta = 0, 1$ with negligible probability, we define a sequence of games:

$$\mathsf{Exp}^{\mathsf{bPRIV}}_{\mathcal{A}}(1^{\ell})[\beta=0] = \mathsf{Game}_0, \mathsf{Game}^*_0, \mathsf{Game}^*_1, \ldots, \mathsf{Game}^*_n = \mathsf{Game}_1 = \mathsf{Exp}^{\mathsf{bPRIV}}_{\mathcal{A}}(1^{\ell})[\beta=1].$$

We start with Game_0 in which the adversary interacting with the $\mathsf{Exp}^{\mathsf{bPRIV}}_{\mathcal{A}}(1^{\ell})[\beta=0]$ and end up with the adversary interacting with the $\mathsf{Exp}^{\mathsf{bPRIV}}_{\mathcal{A}}(1^{\ell})[\beta=1]$ in game Game_1. Every two consecutive games can only be distinguished by the PPT adversary with negligible probability. As a result, we will obtain a bound on the bPRIV distinguishing advantage of any adversary by a negligible quantity in term of the security parameter, effectively showing that **PIN-JCJ** has the bPRIV property.

Before starting the sequence of games, we establish our notations. At any point, since the visible bulletin board is built through a succession of queries ($\mathsf{OvoteLR}(\mathtt{id}, \mathsf{vote}_0, \mathsf{vote}_1)$ and $\mathsf{Ocast}(\mathtt{id}, \mathsf{ballot})$ queries), our bPRIV reduction can associate with any entry $(\mathtt{id}_i, \mathsf{ballot}_i)$ in the visible bulletin board, a tuple $(\mathtt{id}_i, \mathsf{ballot}^0_i, \mathsf{ballot}^1_i, \mathsf{vote}^0_i, \mathsf{vote}^1_i)$ where:

$$\mathsf{ballot}^0_i[\mathsf{vote}^0_i]\ ,\ \mathsf{ballot}^1_i[\mathsf{vote}^1_i]$$

are the ballots seen by the adversary when $\beta = 0, 1$ respectively. Further, vote^0_i and vote^1_i are the votes contained in the corresponding ballots. Typically, if the i-th entry in the bulletin board has been built through a successful $\mathsf{Ocast}(\mathtt{id}, \mathsf{ballot})$ query, then $\mathsf{ballot}^0_i = \mathsf{ballot}^1_i$ and $\mathsf{vote}^0_i = \mathsf{vote}^1_i$. If $(\mathtt{id}_i, \mathsf{ballot}_i)$ is the i-th entry in the visible board, then $\mathsf{ballot}^{\beta}_i = \mathsf{ballot}_i$. We also let $\mathfrak{BB}^i_0$ present the contents of the bulletin board $\mathfrak{BB}_0$ at Game^*_i.

We also remind that we use the Paillier encryption scheme, and Sigma-protocol with perfectly hiding property in our first instantiation, which allows us to fake a proof with the help of the simulator and its trapdoor.

[8]Our proof for bPRIV property is inspired by the privacy proof of Helios e-voting protocol demonstrated in [46]

Game_0 is the bPRIV game corresponding to $\mathsf{Exp}^{\mathsf{bPRIV}}_{\mathcal{A}}(1^{\ell})[\beta = 0]$. In this game, the adversary sees the ballot box $\mathfrak{BB}_0$ with the real result and real auxiliary data.

Game_0^* is the game obtained by introducing the following change in Game_0. In the latter game, the adversary expects to see the output $(\mathsf{res}, \pi_{\mathsf{res}}) \leftarrow \mathsf{tally}(\mathfrak{BB}_0, \mathsf{sk})$. The change consists on simulating the tallying proof $(\mathsf{res}, \Pi^*) \leftarrow \mathsf{SimTally}(.)$ by programming the Random Oracle. Thanks to the Zero-Knowledge property of the Sigma-protocol, the distinguishing probability of the bPRIV adversary in Game_0 negligibly is close to that in Game_0^*. From now on, the tallying proof is always simulated. This proof contains several proofs, such as all decryption proofs and the proofs in the PETsand each change should, in principle, be a hybrid, but we suppress this for simplicity.

For $i = 1, \ldots, n$ where n is the number of entries in the bulletin board $\mathfrak{BB}_0$. we define games Game_i^* by changing contents of $\mathfrak{BB}_0$ from Game_{i-1}^1 to Game_i^1.

Game_i^* is obtained from Game_{i-1}^1 by taking one of two possible actions, depending on the contents of the tuple $(\mathtt{id}_i, \mathsf{ballot}_i^0, \mathsf{ballot}_i^1, \mathsf{vote}_i^0, \mathsf{vote}_i^1)$ that the security reduction keeps internally:

1. If $\mathsf{ballot}_i^0 = \mathsf{ballot}_i^1$ do nothing.
2. If $\mathsf{ballot}_i^0 \neq \mathsf{ballot}_i^1$; replace i-th entry $(\mathtt{id}_i, \mathsf{ballot}_i^0)$ in $\mathfrak{BB}_0$ with $(\mathtt{id}_i, \mathsf{ballot}_i^1)$.
3. Uses the random oracle $\mathbb{H}$ programming to simulate the tally proof. The result function that is used for bPRIV in the case of our schemes is simply outputting all plaintext votes in randomized order.

Game_1 is defined the same as Game_n^*, which in fact is the experiment $\mathsf{Exp}^{\mathsf{bPRIV}}_{\mathcal{A}}(1^{\ell})[\beta = 1]$.

□

Lemma 1. *Assuming the Zero-Knowledge property of the underlying Zero-Knowledge proof system (sigma protocol in our protocol),* Game_0 *and* Game_0^* *are computationally indistinguishable for any PPT adversary.*

Proof. This fact follows from the Zero-Knowledge property of the NIZKP-proof system because any PPT adversary who can distinguish these two games with a non-negligible advantage, can be used to violate the Zero-Knowledge property of the proof system with non-negligible probability. □

Lemma 2. *Assuming the proof of knowledge and Zero-Knowledge property of the underlying Zero-Knowledge proof system, and IND-CPA property of the public-key encryption scheme* Game_i^* *and* Game_{i+1}^* *are computationally indistinguishable from the PPT adversary view, for* $i = 1, \ldots, n$.

Proof. We assume there exists a PPT adversary $\mathcal{A}$ who can distinguish these two games with non-negligible probability:

$$\mathsf{Adv}[\mathcal{A}(\ell)] = \left|\Pr\left[\mathcal{A}(.)^{\rightleftharpoons \mathsf{Exp}^{\mathsf{bPRIV}}_{\mathcal{A}}(1^{\ell})[0]} \mapsto 0\right] - \Pr\left[\mathcal{A}(.)^{\rightleftharpoons \mathsf{Exp}^{\mathsf{bPRIV}}_{\mathcal{A}}(1^{\ell})[1]} \mapsto 1\right]\right|$$

Then we use this adversary to describe a simulator $\mathcal{A}^*$, which uses $\mathcal{A}$ as its subroutine to break the non-malleability property of a non-malleable cryptosystem. Based on this contradiction, we conclude that the advantage of $\mathcal{A}$ should be negligible.

Following [43], a cryptosystem includes an IND-CPA secure encryption scheme and a Zero-Knowledge proof of knowledge system for the encryption relation is non-malleable CPA-secure.[9]As a result assuming **A.4**(See 7.9.2), we consider our cryptosystem a non-malleable cryptosystem.

$\mathcal{A}$ take a challenge ciphertext ct from the NM-CPA and incorporate it into the bPRIV hybrid as ciphertext ct_i. Notice that the $\mathcal{A}$ cannot get information from polynomial evaluation since we randomized it. Now $\mathcal{A}$ submit all $\mathsf{ballot}_{\mathtt{id}_j}$ except $\mathsf{ballot}_{\mathtt{id}_i}$ to NM-CPA for decryption oracle (cannot be equal to ciphertext), and it uses that for output of decryption.Then it simulates proofs.Now, if the adversary $\mathcal{A}$ has a non-negligible advantage in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{bPRIV}}(1^{\ell})$, then $\mathcal{A}^*$ also has a non-negligible advantage in $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{non-mall}}(1^{\ell})$ and based on our assumption, we conclude that $\mathsf{Adv}(\mathsf{Exp}_{\mathcal{A}}^{\mathsf{bPRIV}}(1^{\ell}))$ should be negligible.

Now assuming there exists a PPT adversary $\mathcal{A}$ that distinguishes Game_i^* and Game_{i+1}^* with non-negligible advantage, then we will have an adversary $\mathcal{A}^*$ whose advantage in experiment 2.5 is non-negligible.

To summarize the above arguments, we assume that the adversary $\mathcal{A}^*$, interacts with the challenger $\mathcal{C}^{\mathsf{non-mal}}$ in experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{non-mall}}(1^{\ell})$ 2.5. On the other side, $\mathcal{A}^*$ is interacting with $\mathcal{A}$ as a challenger in the bPRIV experiment 6.1, shown in figure 7.1. Put simply, adversary $\mathcal{A}^*$ set up an election using the public key from $\mathcal{C}^{\mathsf{nm-ma.}}$

□

7.9.3 Strong Consistency Property

Following definition 45 to argue that **PIN-JCJ** protocol achieves strong consistency, we need to prove the following probability in the $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{sCon}}(1^{\ell})$ 6.2 is negligible with respect to the required algorithms Extract, ValidInd and the re-vote function ρ:

$$\Pr\left[\,\mathsf{res} \neq f_{\mathsf{res}}\Big(\rho\big(\{\mathsf{Extract}(\mathtt{id}_i, \mathsf{ballot}_i)\}_{i\in[n]}\big)\Big) \mid \mathsf{Exp}_{\mathcal{A}}^{\mathsf{sCon}}(1^{\ell})\,\right] < \mathsf{negl}(\ell)$$

For the Paillier instantiation of **PIN-JCJ** protocol, these required algorithms and functionality are detailed as follows:

Extract Algorithm. Recall that in **PIN-JCJ** protocol the ballot has the form (7.12):

$$\mathsf{ballot}[\mathsf{crd}, \hat{a}, \mathsf{vote}] = \big(\mathsf{ct}[\mathsf{crd}], \mathsf{ct}[\mathsf{vote}], \mathsf{ct}[\mathsf{crd}, \hat{a}] = (\mathsf{cp}_1^*, \ldots, \mathsf{cp}_k^*), \pi_{\mathsf{ballot}}\big).$$

The Extract algorithm has these steps:

1. check the proof π_{ballot} and returns $\perp$ if $\mathsf{Verify}(\mathsf{ballot}) \mapsto \mathsf{reject}$,
2. using the election's secret key, $\mathsf{SK}_{\mathsf{election}}$ it runs the decryption algorithm to get the polynomial evaluation $\mathsf{poly}_{\mathbf{pin}}(\hat{a}, \mathsf{crd})$,
3. if the polynomial evaluation is not equal to zero, it outputs $\perp$,
4. execute the decryption algorithm

$$\mathsf{Dec}(\mathsf{MSK}, \mathsf{ct}[\mathsf{vote}]) \mapsto \mathsf{vote}^*, \mathsf{Dec}(\mathsf{MSK}, \mathsf{ct}[\mathsf{crd}]) \mapsto \mathsf{crd}$$

and returns $(\mathsf{crd}, \mathsf{vote}^*)$.

[9]In [43], they consider the Sigma-Protocol with a strong form of the Fiat-Shamir transformation which is Zero-Knowledge and simulation sound extractable with respect to expected polynomial-time adversaries.

FIGURE 7.1: This figure depicts the non-malleable experiment where adversary $\mathcal{A}^*$ acts as the bPRIV challenger, $\mathcal{C}^{\mathsf{bPRIV}}$, versus adversary $\mathcal{A}^{\mathsf{bPRIV}}$. In fact, $\mathcal{A}^*$ take advantage of the $\mathcal{A}^{\mathsf{bPRIV}}$ ability to win the non-malleable game while interacting with challenger $\mathcal{C}^{\mathsf{non-mall}}$.

$\mathcal{C}^{\mathsf{non-mall}}$	$\mathcal{A}^{\mathsf{non-mall}}$/ $\mathcal{A}^*$ /$\mathcal{C}^{\mathsf{bPRIV}}$	$\mathcal{A}^{\mathsf{bPRIV}}$
$(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Kgen}(1^\ell)$ $\xrightarrow{\mathsf{pk}}$		
	$\mathsf{pp}_{\mathsf{election}} \leftarrow \mathcal{A}^*(\mathsf{pk})$	
	$\xrightarrow{(\mathsf{pk}_{\mathsf{election}}=\mathsf{pk},\mathsf{pp}_{\mathsf{election}})}$	
		For $j = 1,\dots i-1$ $(\mathtt{id}_j, \mathsf{vote}^0_j, \mathsf{vote}^1_j) \leftarrow \mathcal{A}$
	$\xleftarrow{(\mathtt{id}_j,\mathsf{vote}^0_j,\mathsf{vote}^1_j)}$ $\mathsf{ballot}_{\mathtt{id}_j} = (\mathsf{ct}_{\mathsf{vote}} = \mathsf{Enc}(\mathsf{pk}, \mathtt{id}_j, \mathsf{vote}^1_j), *)$	
		For $j = i$ $(\mathtt{id}_j, \mathsf{vote}^0_j, \mathsf{vote}^1_j) \leftarrow \mathcal{A}$
	$\xleftarrow{(m_0=\mathsf{vote}^0_j, m_1=\mathsf{vote}^1_j)}$	
$\beta \xleftarrow{\$} \{0,1\}$ $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_\beta)$ $\xrightarrow{(\mathsf{ct}^*,\pi)}$		
	$\mathsf{ballot}_{\mathtt{id}_i} = (\mathsf{ct}^*, , \pi_{\mathsf{ballot}}, *)$	
		For $j = i+1,\dots n$ $(\mathtt{id}_j, \mathsf{vote}^0_j, \mathsf{vote}^1_j) \leftarrow \mathcal{A}$
	$\xleftarrow{(\mathtt{id}_j,\mathsf{vote}^0_j,\mathsf{vote}^1_j)}$ $\mathsf{ballot}_{\mathtt{id}_j} = (\mathsf{ct}^*_{\mathsf{vote}} = \mathsf{Enc}(\mathsf{pk}, \mathtt{id}_j, \mathsf{vote}^1_j), *)$	
	For $j \in [n], j \neq i$: $\xleftarrow{(\mathsf{ballot}_j)} \mathcal{A}^*$	
$\xrightarrow{(\mathsf{vote}_j \text{ OR } \perp)} \mathcal{A}^*$		
	$f_{\mathsf{res}}((\mathsf{vote}^1_1, \dots, \mathsf{vote}^1_{i-1}, \mathsf{vote}^0_i), \dots \mathsf{vote}^0_n)$ $\pi' \leftarrow \mathsf{Sim}_{\mathsf{Proof}}(\mathsf{res})$ $\xrightarrow{(\mathsf{res},\pi')} \mathcal{A}$	$\beta' \leftarrow \mathcal{A}(\mathsf{res},\pi')$
	$\beta' \leftarrow \mathcal{A}^*$	

Individual Validation Algorithm. $\mathsf{ValidInd}(\mathsf{ballot})$ checks the validity of the ballot by running the verification algorithms, which takes as input all Zero-Knowledge proof systems generated in the casting phase. The ballot is considered valid if all proofs are accepted and if any proof fails, returns $\perp$.

To specify the re-vote policy, recall that we employ a multi-party computing protocol in Paillier instantiation of the **PIN-JCJ** protocol to choose the ballot with the minimum polynomial evaluation value from all ballots submitted by a single voter with identifier $\mathtt{id}$. If the minimal value appears in more than one ciphertext, MPC_{min} picks a random one. This results in the following re-vote policy:

re-Vote Policy: If the voter $\mathsf{v}_{\mathtt{id}}$ casts several ballots using a valid credential (all of which result in a polynomial evaluation of zero), the tally phase considers a random valid ballot.

Now assume a voter with an identification $\mathtt{id}$ casts q valid ballots:

$$\mathsf{ballot}_1 = [\mathsf{vote}_1], \ldots, \mathsf{ballot}_q = [\mathsf{vote}_q]$$

We show her submitted votes by the set

$$\mathsf{voteList}_{\mathtt{id}} = \{\mathsf{vote}_1^{\mathtt{id}}, \ldots, \mathsf{vote}_q^{\mathtt{id}}\}.$$

According to the above re-vote policy, we formally define the correctness of the tally results in the following definition.

Definition 48. *In e-voting protocol, $\Pi^{\mathsf{pinJCJ}} = \langle n_{\mathsf{v}}, \mathsf{Agent}, \Pi^{\mathsf{h}}, \mathsf{cList}, \Gamma^{\mathsf{elc}}, f_{\mathsf{res}} \rangle$, if the following holds true, we say that the tally result is valid with respect to the tally function f_{res} if there exist $j_1, \ldots, j_{n_{\mathsf{v}}}$ such that:*

$$\mathsf{res} = f_{\mathsf{res}}(\mathsf{vote}_{id_1, j_1}, \ldots, \mathsf{vote}_{id_{n_{\mathsf{v}}}, j_{n_{\mathsf{v}}}}) : \forall i = 1, \ldots, n_{\mathsf{v}} : \mathsf{vote}_{id_i, j_i} \in \mathsf{voteList}_{id_i}$$

Strong Consistency Assumptions We consider the following assumptions to prove the strong consistency of **PIN-JCJ** protocol:

A.1 The underlying public-key encryption scheme, Paillier cryptosystem in this case (BGN and IPE in other instantiations) is correct.

A.2 The signature scheme $\mathfrak{s}$ is EUF-CMA-secure 15.

A.3 The polynomial evaluation process is correct with respect to the error-list. Namely, the zero set of polynomial only contains the error-list, described in the election policy document.

A.4 The MPC protocol, which determines the validity of the polynomial evaluation, is correct.[10]

A.5 $\pi_{\mathsf{Kgen}}, \pi_{\mathsf{Shuffle}}, \pi_{\mathsf{Dec}}, \pi^{\perp}_{\mathsf{Dec}}$ are sound systems and π_{Enc} are a NIZKPoK.

A.6 The scheduler $\mathfrak{s}$, the bulletin board $\mathfrak{BB}$ are honest, the election trustee and all the registrars are honest.[11]

$$\varphi = \mathsf{hon}(\mathfrak{s}) \wedge \mathsf{hon}(\mathfrak{T}) \wedge \mathsf{hon}(\mathfrak{R}) \wedge \mathsf{hon}(\mathfrak{BB})$$

[10]This assumption particularity is needed for the Paillier instantiation. For other instantiation we only need the correctness of the public key cryptosystem..

[11]Simplified version: Assuming the single party setting rather than the distributed setting.

Theorem 7.9.2. *Under the assumptions stated above,* **[A.1-6]** *(See 7.9.3) and the two algorithms* Extract *and* ValidInd*, the protocol* $\Pi^{\mathsf{pinJCJ}} = \langle n_{\mathsf{v}}, \mathsf{Agent}, \Pi^{\mathsf{h}}, \mathsf{cList}, \Gamma^{\mathsf{elc}}, f_{\mathsf{res}} \rangle$ *is strongly consistent.*

Proof. The first condition of strong consistency is that honestly created ballots are correctly extracted, which is concluded by the correctness property of the building blocks: correctness property of the public-key cryptosystem [**A.1**], completeness of the Zero-Knowledge proof system for the relation π_{ballot} [**A.5**] and the correctness of the polynomial evaluation [**A.3**]. Therefore the first and the second condition of definition 45 are met in our protocol.

The third condition says that no adversary can produce a ballot box on which the tally algorithm succeeds but the result is incorrect according to definition 48. But first let us review the tally phase of our protocol.

Assume the ballot box (in our protocol the bulletin board) contains q valid ballots all from a single voter (for simplicity we put $\mathtt{id} = 1$)

$$\mathsf{ballot}_1 = [\mathsf{vote}_1, \mathsf{crd}_1, \hat{a}_1], \ldots, \mathsf{ballot}_q = [\mathsf{vote}_1, \mathsf{crd}_1, \hat{a}_q].$$

In the tally phase we have the following steps; first we obtain the ciphertext with minimum value of $\mathsf{poly}_a(\hat{a} + \mathsf{crd})$ by running the MPC_{min} protocol for each voter. We keep this ballot and remove all the other ballots for voter $\mathsf{v}_{\mathtt{id}}$. Since $\mathsf{poly}_{\mathtt{id},a}$ has the range in non-negative integers if there is any ballot with valid credential and PIN, the output of $\mathsf{MPC}_{\mathsf{min}}$ will be a valid ballot.

Then we anonymize the remaining ballots, and at the end we decrypt both $\mathsf{CT}[\mathsf{poly}_{\mathsf{a}}(\mathsf{crd} + \hat{\mathsf{a}})]$ and $\mathsf{CT}_{\mathsf{vote}}$ and accept all the vote with $\mathsf{poly}_a(\hat{a} + \mathsf{crd}) = 0$ and discard others.

As a consequence, in our protocol, the tally result is incorrect if one of the following failures occurs:

1. A ballot with an invalid credential plus pin is considered valid. This would only happen if one of the two events happen:

   (a) The polynomial evaluation yielded a zero rather than a non-zero number,

   (b) TheMPC_{min} protocol output a ciphertext that does not contain the minimum value.

   Due to the assumptions [**A.1,2**], both of the above events happen with negligible probability.

2. A ballot with a valid long credential and valid PIN is removed from the bulletin board. This may happen if the weeding step in the tally phase fails, which can happen in two ways:

   (a) The polynomial evaluation wrongly outputs a non-zero value instead of zero, which does not occur due to the correctness of the polynomial evaluation and the correctness of the decryption algorithm.

   (b) The ballot was eliminated because it was deemed a duplicate ballot cast by an adversary (not the voter) using the same credential and valid pin. This situation similarly has a negligible probability since the only potential scenario is an adversary successfully guessing the voter credential.

As a result of the above argument plus considering the soundness of the mix-nets leads to the theorem 7.9.2 with respect to assumptions [**A.1-6**] (See 7.9.3). □

7.9.4 Strong Correctness Property

Recall the validation algorithm ValidInd (See 45) that takes as input the public key of the election pk, a ballot ballot, and outputs accept or reject. The definition of strong correctness (See 6.4.3) to show that our protocol achieves the strong correctness property, we need to prove that no polynomial adversary can generate a bulletin board $\mathfrak{BB}^{\mathcal{A}}$, such that a fresh honestly created ballot for an honest voter who has not voted previously is rejected.

We emphasize two points:

i. Because the **PIN-JCJ** protocol allows for re-voting, a valid ballot cast by the voter will be accepted even if the adversary casts the ballot in advance on the voter's behalf. However, it should be noted that this may not be true for all protocols, such as the DeVoS protocol presented in 8 and extended Helios presented in [180], because in these protocols, the fresh vote is dependent on the previously submitted ballot. As a result, an adversary may cast a vote that invalidates the subsequent one.

ii. There is a possibility that the adversary casts a valid ballot on the voter's behalf, which will result in the removal of the original honest ballot as a duplicate of the adversary ballot during the tally phase. We prove that the probability of this event occurring for an adversary without knowing the voter credential is negligible due to the following assumptions.

Strong Correctness Assumptions: To prove the strong correct property of **PIN-JCJ**, we make the following assumptions about the primitives we use (See 7.6 and 7.9):

A.1 The underlying public key encryption scheme, Paillier cryptosystem in this case (BGN and IPEin other instantiations), are IND-CPA-secure.

A.2 The signature scheme is EUF-CMA-secure.

A.3 The $\mathsf{MPC}_{\mathsf{min}}$ protocol is correct.

A.4 All the proofs, $\pi_{\mathsf{Kgen}}, \pi_{\mathsf{Shuffle}}, \pi_{\mathsf{Dec}}$ and π_{Enc} are Zero-Knowledge proof system.

A.5 The scheduler $\mathfrak{s}$, the bulletin board $\mathfrak{BB}$ are honest, the election trustee and all the registrars are honest.

$$\varphi = \mathsf{hon}(\mathfrak{s}) \wedge \mathsf{hon}(\mathfrak{T}) \wedge \mathsf{hon}(\mathfrak{R}) \wedge \mathsf{hon}(\mathfrak{BB})$$

A.6 The polynomial evaluation process is correct respect to the error-list. Namely, the zero set of polynomial only contains the error list, described in the election policy document.

As a result of the preceding argument leads to the theorem 7.9.3 with respect to above assumptions, **A[1-6]**:

Theorem 7.9.3. *Under the assumptions stated above, **A.1-6** the protocol* $\Pi^{\mathsf{pin-JCJ}}$ *described by the tuple* $\langle n_{\mathsf{v}}, \mathsf{Agent}, \Pi^{\mathsf{h}}, \mathsf{cList}, \Gamma^{\mathsf{elc}}, f_{\mathsf{res}} \rangle$ *is strongly correct in the presence of a computational-bounded adversary.*

7.9.5 Verifiability

In this section, we formally evaluate the **PIN-JCJ** protocol's verifiability. To this end we use the general KTV computational model and adapt the verifiability definition with the goal $\gamma(\varphi)$ proposed in [85].

Our primary reason for choosing this model for the verification analysis, apart from its expressiveness that it is particularly suitable for JCJ analysis, as there are no explicit assumptions regarding the result function and re-voting policy. Additionally, it is applicable to $\mathsf{MPC}_{\mathsf{min}}$ protocols.

In our model judge, $\mathfrak{J}$, an honest agent, is in charge of the verification procedure, and he executes the honest program $\hat{\pi}_{\mathfrak{J}}$ whenever triggered by the scheduler $\mathfrak{s}$. At the end of the verification procedure, $\mathfrak{J}$ outputs accept or reject through his own channel.

In a nutshell, in a **PIN-JCJ** protocol run, judge $\mathfrak{J}$ takes as input only public information (e.g., the Zero-Knowledge proofs in **PIN-JCJ** published on the bulletin board) and then performs certain checks. If all checks succeed, the judge accepts the protocol run, or otherwise rejects it. Precisely judge $\mathfrak{J}$ conduct the program $\hat{\pi}_{\mathfrak{J}}$ as defined in Figure 7.2.

Verifiability Assumption. We prove the verifiability property for **PIN-JCJ** for the goal $\gamma(k,\varphi)$ as defined in 42 and under the following assumptions:

A.1 The underlying public-key cryptosystem (such as BGN, Paillier or IPE) is correct.

A.2 The probability that an adversary retrieves (such as blindly guessing or stealing the voter credential) the voter's PIN and long credential (crd) is negligible because $\mathsf{crd} \xleftarrow{\$} \{0,1\}^{\ell}$ is a long string and it is chosen uniformly at random. Therefore, we let $\mathsf{Pr}_{\mathsf{pin}}$ and $\mathsf{Pr}_{\mathsf{crd}}$ represent the probability that an adversary get the voter's PIN and long credential (crd), respectively. It it worth mentioning that, the two credentials, long credential (crd) and PIN, are chosen randomly and independently in the registration phase. Hence the probability of recovering both of them, namely the full credential, is:

$$\mathsf{Pr}_{\mathsf{full.crd}} = \mathsf{Pr}_{\mathsf{pin}} \times \mathsf{Pr}_{\mathsf{crd}}.$$

A.3 The polynomial evaluation process is correct respect to the error-list. Namely, the zero set of polynomial only contains the error list, described in the election policy document.

A.4 The MPC protocol $\mathsf{MPC}_{\mathsf{min}}$ is $\big(\gamma(0,\varphi),0\big)$-verifiable, meaning that if the output of $\mathsf{MPC}_{\mathsf{min}}$ does not correspond to its input, then this can always be detected publicly.

A.5 $\pi_{\mathsf{Kgen}}, \pi_{\mathsf{Shuffle}}, \pi_{\mathsf{Dec}}$ are NIZK systems and π_{Enc} is a NIZK Proof of Knowledge.

A.6 The scheduler $\mathfrak{s}$, the bulletin board $\mathfrak{BB}$, and the judge $\mathfrak{J}$ are honest:

$$\varphi = \mathsf{hon}(\mathfrak{s}) \wedge \mathsf{hon}(\mathfrak{BB}) \wedge \mathsf{hon}(\mathfrak{J}) \wedge \mathsf{hon}(\mathsf{MPC}_{\mathsf{min}}).$$

Additional Note. For verifiability to hold, a minority of election trustee, the mix server $\mathfrak{mix.s}$, as well as an arbitrary number of voters may be controlled by the adversary. In particular, we do not need to introduce any additional trust assumption compared to basic secure e-voting protocols (without coercion-resistance), such as JCJ [83].

According to [182], a goal γ is *verifiable* by the judge, $\mathfrak{J}$, in a protocol's run if and only if $\mathfrak{J}$ accepts a run r of **PIN-JCJ** in which the goal γ is violated (i.e., $r \notin \gamma$) with at most negligible probability (in the security parameter). To formally capture this notion by

$$\Pr[(\hat{\pi}_{\mathsf{P}} \| \pi_{\mathcal{A}})^{(\ell)} \mapsto \neg\gamma, (\mathfrak{J}: \mathsf{accept})]$$

we denote the probability that a run of the protocol along with an adversary $\pi_{\mathcal{A}}$ (and a security parameter ℓ) produces a run that is not in γ but in which $\mathfrak{J}$ (nevertheless) returns accept.

This probability should be negligible in terms of the security parameter. Hence intuitively, to prove the verifiability we need to demonstrate that the probability that in a run of **PIN-JCJ** more than k votes of honest voters have been manipulated but the judge $\mathfrak{J}$ nevertheless accepts the run is bounded:

Theorem 7.9.4 (Verifiability)**.** *Under the assumptions* **[A.1-6]** *stated above, the goal* $\gamma(\varphi)$ *is verifiable in the protocol* $\Pi^{\mathsf{pinJCJ}} = \langle n_{\mathsf{V}}, \mathsf{Agent}, \Pi^{\mathsf{h}}, \mathsf{cList}, \Gamma^{\mathsf{elc}}, f_{\mathsf{res}} \rangle$ *by the judge* $\mathfrak{J}$ *which run the honest program* $\hat{\pi}_{\mathfrak{J}}$ *as described in Figure 7.2.*

Proof. Assume that assumptions [**A.1-6**], as specified above hold true. Then to prove Theorem 7.9.4, we need to show the following implication.

If the judge $\mathfrak{J}$ outputs accept in a given protocol's run of **PIN-JCJ** (in which **A.[1-6]** are satisfied), then there exist (valid) dishonest choices $(\mathsf{c}_i)_{i \in I_d}$ such that the election result equals $f_{\mathsf{res}}(\mathsf{c}_{\sigma(i)})_{i \in I_h \cup I_d}$, where $(\mathsf{c}_i)_{i \in I_h}$ are the honest voters' choices and σ is some permutation. This means that, due to the specification of $\mathfrak{J}$ Figure 7.2, each NIZKP published on $\mathfrak{BB}$ is valid.

Let $\mathsf{v}_{\mathtt{id}}$ be an arbitrary *honest* voter who submitted q valid ballots, each containing voter choice $\mathsf{vote}_{\mathtt{id},i}$:

$$\begin{aligned} \mathsf{ballotList}_{\mathtt{id}} &:= \{\mathsf{ballot}_{\mathtt{id},1}[\mathsf{vote}_1], \ldots, \mathsf{ballot}_{\mathtt{id},q}[\mathsf{vote}_q]\} \\ \mathsf{voteList}_{\mathtt{id}} &:= \{\mathsf{vote}_{\mathtt{id},1}, \ldots, \mathsf{vote}\mathtt{id}, q\}. \end{aligned} \tag{7.21}$$

Since $\mathfrak{J} \mapsto$ accept and according to the completeness property of the π_{ballot}(7.11), all the ballots $\mathsf{ballot}_{\mathtt{id},j}$ would pass the first verification step. We highlight that with overwhelming probability all valid ballots in $\mathsf{ballotList}_{\mathtt{id}}$(7.21) are cast by the voter. Because the adversary can submit a ballot on behalf of the voter $\mathsf{v}_{\mathtt{id}}$, but due to the assumption **A.2**, the ballot would be valid with a negligible probability.

In the next step, all these ballots go through the multi-party computation algorithm, $\mathsf{MPC}_{\min}$. Because of step 6, in figure 7.2, and the soundness property of the π_{MPC}, the output of the $\mathsf{MPC}_{\min}$ would be some index t such that:

$$\forall k = 1, \ldots, q : \ 0 \le \mathsf{poly}_{\mathtt{id},a}(\hat{a}_m, \mathsf{crd}) \le \mathsf{poly}_{\mathtt{id},a}(\hat{a}_k, \mathsf{crd}) \tag{7.22}$$

and since the honest voter cast at least one valid ballot (there exist some index j such that:

$$\exists\, j \in [q] \ : \ \mathsf{poly}_{\mathtt{id},a}(\hat{a}_j, \mathsf{crd}) = 0$$

The above inequality (7.22) implies that $\mathsf{poly}_{\mathtt{id},a}(\hat{a}_m, \mathsf{crd}) = 0$. We now show that $\mathsf{ballot}_{\mathtt{id},m}[\mathsf{vote}_{\mathtt{id},m}]$ is counted in the tally phase with overwhelming probability.

In the next step, the tabulation teller, randomize the polynomial evaluation and then applies the mix-net to the list $\mathbf{L}_1$ and obtain $\mathbf{L}_2$:

$$\begin{aligned} \mathbf{L}_1 &= \Big[\big(\mathsf{ct}_{\mathtt{id}} = \mathsf{ct}[\mathsf{poly}_{\mathtt{id},a}(\hat{a}_m + \mathsf{crd}_{\mathtt{id}})], \mathsf{ct}'_{\mathtt{id}} = \mathsf{ct}[\mathsf{vote}_{\mathtt{id},m}]\big)\Big]_{\mathtt{id}\in\mathsf{idSet}} \\ \mathbf{L}_1 &\xrightarrow{\pi_{\mathsf{randomization}},\pi_{\mathsf{Shuffle}}} \mathbf{L}_2 \\ \mathbf{L}_2 &= \Big[(\mathsf{ct}_{\sigma(\mathtt{id})}, \mathsf{ct}'_{\sigma(\mathtt{id})}])\Big]_{\mathtt{id}\in\mathsf{idSet}} \end{aligned} \tag{7.23}$$

Since the judge accepted the protocol's run, $\pi_{\mathsf{randomization}}$ and the mix server's NIZKP for $\mathrm{R}_{\mathsf{shuffle}}$ 8.2 are valid. Due to the soundness of π_{Shuffle} (**A.5**), it follows that, with overwhelming probability, there exists some permutation σ and some index $j \in [n_{\mathsf{v}}]$ such that for each $\mathtt{id} \in \mathsf{idSet}$, we have that:

$$\mathsf{ct}'_{\mathtt{id}} \in \mathsf{ReEnc}(\mathsf{pk}, \mathsf{ct}_{\sigma(j)}).$$

Due to the re-encryption property of Π^{pke} (**A.4**) and the bijective property of σ, we can deduce that for each honest voter, there exists ct_j in the output list $\mathbf{L}_2$ of mix server such that

$$(\mathsf{ct}'_i, \mathsf{ct}'_j = \mathsf{ct}[\mathsf{vote}_j]) \in \mathsf{ReEnc}(\mathbf{L}_1)$$

where $\mathsf{vote}_j \in \mathsf{voteList}_j$, which is one of the voter's original choices, and analogously for the dishonest voters. More precisely, there exists some index j that $\sigma(j) = \mathtt{id}$, which implies the ballot containing $\mathsf{vote}_{\mathtt{id},x}$ is now entering the decryption step.

Since the judge accepted the protocol run, the decryption trustee's NIZKPs, π_{Dec}, is valid. Due to the soundness of π_{Dec} (**A.5**), it follows that, with overwhelming probability, there exists sk such that $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Kgen}$ such that:

$$(\mathsf{poly}_{\mathtt{id},a}(\hat{a}_m + \mathsf{crd})^{\mathsf{random}}, \mathsf{vote}_{\mathtt{id},m}) = \mathsf{Dec}(\mathsf{sk}, (\mathsf{ct}_{\sigma(j)}, \mathsf{ct}'_{\sigma(j)})).$$

Because the encryption scheme is correct (**A.1**), and the soundness of property of π_{dec} the output of the decryption algorithm on input with index $j : \sigma(j) = \mathtt{id}$ would be $(0, \mathsf{vote}_{\mathtt{id},m})$.

We can make the same argument as above for each honest voter; this leads to the following conclusion:

$$\mathsf{res} = f_{\mathsf{res}}(\mathsf{vote}_1, \ldots, \mathsf{vote}_{n_{\mathsf{v}}}) : \forall i = 1, \ldots, n_{\mathsf{v}} : \mathsf{vote}_i \in \mathsf{voteList}_i$$

which establish the verifiability property of our protocol.

□

FIGURE 7.2: Verification Procedure

Inputs: This procedure takes as input all public data of the election, namely the public bulletin board, $\mathfrak{BB}$,

Output: accept/reject

1. Run the verification algorithm for the relation $\mathsf{R}_{\mathsf{KeyGen}}$ (see 7.3) with inputs the public key of the election and π_{Kgen}:

$$\begin{cases} \textbf{If } [\mathsf{Verify}(\mathsf{PK}_{\mathsf{election}}, \mathsf{R}_{\mathsf{KeyGen}}, \pi_{\mathsf{Kgen}}) \to 0] \textbf{ Then } (\mathfrak{J} \mapsto \mathsf{reject}), \\ \textbf{If } [\mathsf{Verify}(\mathsf{PK}_{\mathsf{election}}, \mathsf{R}_{\mathsf{KeyGen}}, \pi_{\mathsf{Kgen}}) \to 1] \textbf{ Then } \text{ Go to the next step,} \end{cases}$$

2. Run the verification algorithm for the relation $\mathsf{R}_{\mathsf{poly}}$ (see 7.9) for all registered voters with inputs $\mathtt{v}_{\mathtt{id}}, \mathsf{CP}, \mathsf{Enc}(\mathsf{crd})$

$$\begin{cases} \textbf{If } [\mathsf{Verify}(\mathsf{pp}, \mathsf{R}_{\mathsf{poly}}, \mathtt{v}_{\mathtt{id}}, \mathsf{CP}, \mathsf{Enc}(\mathsf{crd}), \pi_{\mathsf{poly}}) \to 0] \textbf{ Then } (\mathfrak{J} \mapsto \mathsf{reject}), \\ \textbf{If } [\mathsf{Verify}(\mathsf{pp}, \mathsf{R}_{\mathsf{poly}}, \mathtt{v}_{\mathtt{id}}, \mathsf{CP}, \mathsf{Enc}(\mathsf{crd}), \pi_{\mathsf{poly}}) \to 1] \textbf{ Then } \text{ Go to the next step,} \end{cases}$$

3. Run the verification algorithm for the relation (see 7.11) to check the validity of the proofs, π_{ballot}. Same as the previous step, the judge reject the election in case
   - there is a ballot with a valid proof π_{ballot} which is withdrawn,
   - there is a ballot with an invalid proof counted as a legitimate ballot.
4. Run the verification algorithm for proof $\pi_{\mathsf{randomization}}$ with respect to relation $\mathsf{R}_{\mathsf{randomization}}$ 7.16, reject if the output of the algorithm is 0. Otherwise go to the next step.
5. Run the verification algorithm for proof π_{Shuffle} with respect to relation $\mathsf{R}_{\mathsf{shuffle}}$ 8.2, reject if the output of the algorithm is 0. Otherwise go to the next step.
6. In case the protocol is instantiated with multi-party computation then during the execution of $\mathsf{MPC}_{\mathsf{min}}$ the judge runs the judging procedure $\mathsf{Verify}_{\mathsf{mpc}}$ of $\mathsf{MPC}_{\mathsf{min}}$, and $\mathfrak{J}$ outputs reject if $\mathsf{Verify}_{\mathsf{mpc}}$ rejects the computation. If the protocol run the polynomial evaluation, $\mathfrak{J}$ needs to run the verification algorithm for the polynomial evaluation.
7. Run the verification algorithm for $\mathsf{R}_{\mathsf{dec}}$ to verify the decryption procedure and output reject in case any decryption proof fails.
8. If none of these situations occur, the judge $\mathfrak{J}$ outputs accept on a distinct tape.

7.10 Conclusion

In this paper we have presented attacks and repairs on the NV12 scheme, especially, we have also presented protocols which are resilient to human errors in the form of PIN typos. It is interesting to notice that the digitally stored key could be combined or replaced with a key derived from biometric data. An important future direction is to make the error correction here so efficient that we can allow using noisy biometric data without fuzzy extraction.

For the Paillier-based system that we have presented it would be natural to add the tally system from Ordinos [184] since this is also based on Paillier encryption. Ordinos will only reveal the winner or the ranking of the candidates in the election, and will thus help for coercion-resistance in the case where there are candidates which expected to only get few or no votes. Another method that could used in both protocols is the risk-limiting tally method described in [164] which gives plausible deniability for the voter.

Finally, some socio-tehcnical research questions are:

1. Which type of PIN errors do voters do when the are in a vote setting and do not get any feedback on the correctness of the PIN.
2. Related to this, what it the optimal PIN policy that corrects as many PIN typos while still keeping the entropy of the PIN space sufficiently high.
3. If we do not use a smart card, or use both a smart card and key storage: how well can voters be trained to handle, fake and hide secret keys.

Chapter 8

A New Technique for Deniable Vote Updating

This chapter proposes a new e-voting system that enables voters with an intuitive mechanism to update their possibly coerced vote in a deniable way. What is more, our e-voting system does not introduce any additional trust assumptions for end-to-end verifiability and vote privacy besides the standards. Moreover, we demonstrate that our e-voting system can be instantiated efficiently for practical elections. With these properties, our e-voting system has the potential to close the gap between theory and practice in coercion-resistant e-voting.

Contents

8.1 Introduction

Designing e-voting systems for practical elections that satisfy at least a basic level of security is a demanding task. On top of this, to make the system resistant against malicious influencers who want to swing an election by coercing voters is even more challenging. Numerous e-voting systems have been proposed to reduce the risk of coercion, many of them achieve this by sacrificing efficiency, usability, or basic security, which are all required for real-world elections.

In order to overcome this unsatisfying state-of-affairs, we propose a new e-voting system which enables voters with an intuitive mechanism to update their possibly coerced vote in a deniable way. What is more, our e-voting system does not introduce any additional trust assumptions for end-to-end verifiability and vote privacy besides the standards. Moreover, we demonstrate that our e-voting system can be instantiated efficiently for practical elections. With these properties, our e-voting system has the potential to close the gap between theory and practice in coercion-resistant e-voting.

In what follows, we review the related work and state-of-affairs in Section 8.2. Then in Section 8.3, we describe the main idea of DeVoS, explain why it guarantees the features claimed, and discuss the coercion threat model we consider. Next, the complete DeVoS protocol is presented in Section 8.4 with full technical details. In Section 8.5, we instantiate our protocol with two different cryptosystems. Finally, we formally analyze the security of DeVoS in Section 8.6.

8.2 Related work

Let us first recap the basic idea of coercion-resistant e-voting systems. Instead of obeying the coercer, each coerced voter can run some *counter-strategy* in such systems. As a result, the coerced voter can achieve her own goal (e.g., voting for her favorite candidate). At the same time, due to some technical mechanisms, the coercer should not be able to distinguish whether the coerced voter followed his instructions (e.g., voted for the coercer's favorite candidate) or ran the counter-strategy. From a technical perspective, three different approaches in the literature implement this concept: *fake credentials*, *masking*, and *deniable vote updating*. We will briefly explain these different approaches next.

Fake credentials are used, for example, in [168, 80, 17, 79, 97, 232], and they work as follows. Each voter is provided with a unique and secret credential $\hat{c}$. A voter uses $\hat{c}$ to submit her vote when she is not under coercion. Otherwise, if a voter is under coercion, she can create a so-called *fake credential* c to submit her coerced vote. Since the voter's fake credential is invalid, the voting authorities will secretly remove the vote. At the same time, the fake credential c and the real one $\hat{c}$ are indistinguishable from a coercer's perspective.

Masking choices is employed, for example, in [21, 259]. Its idea is the following one. Each voter is provided with a unique and secret mask $\hat{m}$. A voter uses $\hat{m}$ to blind her actual vote $\hat{v}$ when she is not under coercion. Otherwise, if a voter is being coerced to vote for a different choice v, then she computes a fake mask m such that the resulting blinded vote still remains a vote for her actual choice $\hat{v}$.

Unfortunately, in both the fake credential and masking approach, the ceremonies that voters have to run appear to be too complex for real human voters because they typically require capabilities that are difficult or impossible to attain (e.g., memorizing long, randomly-looking credentials or inputting these credentials without errors). We refer the interested reader to [205, 179] for more details on the usability issues of fake credentials and masking choices. Due to such drawbacks, these two concepts are likely rendered completely ineffective to protect against coercion in real practical elections. Achieving coercion-resistance via deniable vote updating, as described next, is more promising.

Deniable vote updating enables each voter to overwrite her previously submitted ballot, that she may have cast under coercion, such that no one else, including a possible coercer, can see whether or not the voter has subsequently updated her vote. Among others, this technique is employed in the hybrid e-voting system used for national elections in Estonia [154] and formerly in Norway. Here, voters can overwrite electronically cast ballots by submitting a physical ballot at the polling station. There are also exist solutions to deniable vote updating for completely remote e-voting systems, most notably [180, 197].

In an e-voting system with deniable vote updating, "the vote casting process is no different from simpler voting systems that do not ensure coercion resistance" and "even in case of coercion, the concept of voting again to overwrite the vote cast under coercion would most probably fit into the mental models of the voters" [179]. Despite these significant advantages, existing e-voting systems with deniable vote updating have fundamental restrictions. We will elaborate on this observation in what follows.

The deniable vote updating techniques proposed in [9, 196] require that the voters' submitted ballots are secretly compared pairwise, leading to a quadratic complexity in the number of voters. This property is undesirable for elections with medium-size or larger electorates. More specifically, as demonstrated in [197], in an election with 180,000 voters, the solution by [9] would require more than one core *year* to make the comparisons.

The only *scalable* e-voting systems with deniable vote updating are [180, 197]. These techniques have in common that several indistinguishable dummy ballots hide the voters' re-voting pattern. Unfortunately, even though [180] follows the concept of deniable vote updating, the counter-strategy proposed in [180] is cumbersome for human voters. If a voter in [180] wants to update a choice v that she submitted under coercion, then she needs to memorize v, invert it, multiply the result with her truly favorite choice $\hat{v}$, and submit a ballot for $v^{-1} \cdot \hat{v}$. The procedure becomes even more complex if the coercer asks the voter to submit several choices, each one updating the one submitted before. It is questionable whether human voters can execute this complex counter-strategy and thus whether [180] provides a sufficient level of coercion-resistance in practice. In contrast to [180], the counter-strategy voters have to run in [197] is as easy as it could possibly be. In fact, if a voter was coerced to submit a vote for v (or even a sequence of votes), she can simply, at any later point of the submission phase, cast a vote for her truly favorite choice $\hat{v}$ (without having to memorize any previously cast a vote). On the downside, [146] demonstrated that [197] does not provide a reasonable level of security because there exists a single voting authority in [197] which needs to be trusted for all security properties, i.e., verifiability, privacy, and coercion-resistance. According to [146], this security issue is intrinsic to the approach taken in [197] and could thus, if possible, only be resolved by fundamental modifications.

Altogether, we can conclude that, to date, there does not exist a practically efficient e-voting system in the literature which is provably secure and provides human voters with a simple counter-strategy to defeat coercion.

8.2.1 Our Contributions

In order to overcome the unsatisfying state-of-affairs described in Sec. 8.2, we propose DeVoS, the first remote e-voting system that satisfies all of the following properties:

- Voters can deniably intuitively update their votes.
- End-to-end verifiability and vote privacy are provably guaranteed without any additional trust assumptions besides the standards.
- Large-scale real-world elections can be realized efficiently.

With the unique combination of these properties, DeVoS has the potential to close the gap between coercion-resistant e-voting in theory and practice.

8.3 Overview of DeVoS

We describe the main idea of DeVoS, explain why it guarantees the features claimed, and discuss the coercion threat model we consider.

8.3.1 Main Idea

In a nutshell, DeVoS works as follows; we provide full technical details in Sec. 8.4. For each voter v_i, there exists an initially empty vector list_i on the public bulletin board $\mathfrak{BB}$ to which ciphertext/proof pairs $(\mathsf{ct}_i^j, \pi_i^j)$ can be appended.[1] The idea is that ct_i^j encrypts a candidate/choice and that π_i^j is a zero-knowledge proof (ZKP) that ct_i^j is valid in the following sense:

1. Voter v_i herself, but no one else, is permitted to append a "fresh" ciphertext ct_i^j to list_i, i.e., a ciphertext which is completely unrelated to the previous ciphertexts $\mathsf{ct}_i^0, \ldots, \mathsf{ct}_i^{j-1}$ in her vector list_i.
2. Each other participant is only permitted to submit a ciphertext ct_i^j which contains the same plaintext as the last ciphertext ct_i^{j-1} in list_i. Technically speaking, each participant can only append a re-randomization of ct_i^{j-1} to list_i.

Once the submission phase has closed, the last ciphertext in v_i's vector list_i, denoted by ct_i, is v_i's input to the subsequent publicly verifiable mixing phase (that is standard). Due to the soundness of the ZKP π_i^j, it is ensured that ct_i indeed contains the last vote submitted by v_i, which is necessary for end-to-end verifiability.

To achieve deniable vote-updating (and thus coercion-resistance), we exploit the zero-knowledge property of π_i^j (which hides whether or not ct_i^j is a re-randomization of ct_i^{j-1}), as explained next. We employ a *posting trustee* PT whose

[1] On the contrary, in basic secure e-voting systems (e.g., Belenios [83]), each voter v_i is assigned a *single* pair (ct_i, π_i) instead of a vector list_i of pairs.

role is to add indistinguishable *noise ballots* to the voters' ballot vectors list_i at certain times. The posting trustee, PT is the authority who collects all incoming ballots and updates the current status of all voters' ballot vectors list_i periodically. Now, instead of publicly updating list_i on the bulletin board $\mathfrak{BB}$ immediately after a new ballot has arrived, we introduce some (short) delay. After some specified time (e.g., 1 min) has passed, the posting trustee PT adds noise ballots for (some of) those voters v_i who did not submit a ballot since the last update. Technically, the posting trustee PT constructs a noise ballot $(\mathsf{ct}_i^j, \pi_i^j)$ as follows: re-randomize the currently last ciphertext ct_i^{j-1} in list_i and generate a ZKP π_i^j that ct_i^j is a re-randomization of ct_i^{j-1}. In this way, a coercer is not able to distinguish whether v_i updated her choice (after she submitted a coerced ballot) or the posting trustee added a noise ballot because (1) "freshly" generated ciphertexts and re-randomized ciphertext are computationally indistinguishable, and (2) the ZKP π_i^j does not reveal whether or not ct_i^j is a noise ciphertext. This property ensures that if voter v_i was coerced to submit some ciphertext ct, she may, at any later point of the submission phase, submit a new ciphertext ct' which contains her favorite choice and update list_i accordingly.

8.4 Protocol Description; Participants, Primitives and Framework

We present the DeVoS e-voting protocol with full technical details.

8.4.1 Protocol Participants

The DeVoS protocol is run among the following participants: voting authority $\mathfrak{Auth}$, bulletin board $\mathfrak{BB}$, mix server $\mathfrak{mix.s}$, decryption trustee $\mathfrak{DT}$, posting trustee PT, and voters $\mathsf{v}_1, \ldots, \mathsf{v}_{n_\mathsf{v}}$. Observe that all participants except for the posting trustee PT are standard in modern secure e-voting protocols (where ballots are mixed in the tallying phase). The role of the mix server $\mathfrak{mix.s}$, decryption trustee $\mathfrak{DT}$, and posting trustee PT can easily be distributed; to simplify the presentation of DeVoS, we assume that a single participant runs the respective program.

We assume all parties have authenticated channels to and from the bulletin board, $\mathfrak{BB}$. This (standard) assumption ensures that all parties have the same view on the bulletin board, and that the bulletin board can authenticate the senders of incoming messages. To guarantee deniable vote-updating, it is also necessary to assume that the channels between individual voter's $\mathsf{v}_{\mathtt{id}}$ and the posting trustee PT are authenticated and untappable.

8.4.2 Cryptographic Primitives

We start with the cryptographic primitives used in DeVoS:[2]

- **An IND-CPA-secure public-key encryption scheme:** $\Pi^{\mathsf{pke}} = (\mathsf{Kgen}, \mathsf{Enc}, \mathsf{Dec})$. We assume that Π^{pke} allows for *re-encryption*, i.e., there exists a ppt algorithm ReEnc which takes as input public key pk and ciphertext $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m; \mathsf{r})$ and outputs ciphertext ct' such that $\mathsf{ct}' = \mathsf{Enc}(\mathsf{pk}, m; \mathsf{r}')$ for some (fresh) random r'.

- A **one-way function** $f\colon \{0,1\}^* \to \{0,1\}^*$ (see 2.5).

[2]Instead of relying on specific primitives, the security of DeVoS (8.6) can be guaranteed under certain assumptions these primitives have to satisfy. We will demonstrate in Sec. 8.5 how to instantiate the generic DeVoS e-voting protocol with highly efficient cryptographic primitives.

- **Non-interactive zero-knowledge proof:** For the protocol's verifiability, we use a non-interactive zero-knowledge proof system to prove correctness and plaintext knowledge for all encrypted data, such as "Proof of Plaintext Knowledge" and "Proof of Decryption Validity." We may utilize either a zero-knowledge proof system or a witness-indistinguishable proof system for this aim. Formally we will use the following proofs in our protocol:

  1. A *proof of correct key generation*, i.e., a *non-interactive zero-knowledge proof* (NIZKP) π_{Kgen} for proving the correctness of a public key pk w.r.t. Π^{pke}.
     The underlying relation $\mathrm{R}_{\mathsf{KeyGen}}$ is

     $$\big(x = \mathsf{pk}, w = (\mathsf{random}, \mathsf{sk})\big) \in \mathrm{R}_{\mathsf{KeyGen}} \Leftrightarrow (\mathsf{pk}, \mathsf{sk}) = \mathsf{Kgen}(\mathsf{random}). \tag{8.1}$$

  2. A *proof of correct encryption* π_{Enc}, i.e., a NIZKP of knowledge (NIZKPoK) for proving correctness of a ciphertext ct' w.r.t. public key pk, public verification key vk, and (previous) ciphertext ct. The underlying relation $\mathrm{R}_{\mathsf{enc}}$ is

     $$\begin{aligned}\big(x = (\mathsf{pk}, \mathsf{vk}, \mathsf{ct}, \mathsf{ct}'), w\big) \in \mathrm{R}_{\mathsf{enc}} \Leftrightarrow \big(w = \mathsf{r} : \mathsf{ct}' = \mathsf{ReEnc}(\mathsf{pk}, \mathsf{ct}; \mathsf{r})\big) \vee \\ \big(w = (\mathsf{ssk}, m, \mathsf{r}) : \mathsf{vk} = f(\mathsf{ssk}) \wedge \mathsf{ct}' = \mathsf{Enc}(\mathsf{pk}, m; \mathsf{r})\big).\end{aligned}$$

     The relation $\mathrm{R}_{\mathsf{enc}}$ is a disjunction of two statements:
     (a) ct' is a re-encryption of ct, or
     (b) ct' is a "fresh" encryption of some message m *and* the prover knows a valid secret signing key ssk for public verification key vk.

  3. A *proof of shuffle*, π_{Shuffle} concerning the cryptosystem Π^{pke}, i.e., NIZK proof of knowledge for the following relation $\mathrm{R}_{\mathsf{shuffle}}$:

     $$\begin{gathered}\big(x = ((\mathsf{ct}_i)_{i=1}^n, (\mathsf{ct}'_i)_{i=1}^n), w = ((\mathsf{r}_i)_{i=1}^n, \sigma)\big) \in \mathrm{R}_{\mathsf{shuffle}} \\ \Updownarrow \\ \big(\sigma : [n] \to [n] \text{ bijective}\big) \wedge \big(\forall i \in [n] : \mathsf{ct}'_i = \mathsf{ReEnc}(\mathsf{pk}, \mathsf{ct}_{\sigma(i)}; \mathsf{r}_i)\big).\end{gathered} \tag{8.2}$$

     In other words, the public statement of a proof of shuffle consists of two ciphertext vectors $(\mathsf{ct}_i)_{i=1}^n$ and $(\mathsf{ct}'_i)_{i=1}^n$ of the same size, and if the prover's output is valid, then this implies that the prover knows random coins $(\mathsf{r}_i)_{i=1}^n$ and a permutation σ over $[n]$ such that ct'_i is a re-encryption of $\mathsf{ct}_{\sigma(i)}$ (using randomness r_i).

  4. A *proof of correct decryption* π_{Dec} concerning Π^{pke}, i.e., a NIZKP for the following relation $\mathrm{R}_{\mathsf{dec}}$:

     $$\big(x = (\mathsf{pk}, m, \mathsf{ct}), w = (\mathsf{r}, \mathsf{sk})\big) \in \mathrm{R}_{\mathsf{dec}} \Leftrightarrow \big(m = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})\big) \wedge \big(\mathsf{pk}, (\mathsf{r}, \mathsf{sk})\big) \in \mathrm{R}_{\mathsf{KeyGen}}.$$

     Additionally, we also use a NIZKP $\pi_{\mathsf{Dec}}^{\perp}$ for proving that a ciphertext ct does not decrypt to a message $m \in \mathsf{cList}$, where cList is a certain set of plaintexts (the set of valid choices), without revealing the actual plaintext. The respective relation $\mathrm{R}_{\mathsf{dec}}^{\perp}$ is

     $$\big(x = (\mathsf{pk}, \mathsf{ct}), w = (\mathsf{r}, \mathsf{sk}, m)\big) \in \mathrm{R}_{\mathsf{dec}}^{\perp} \Leftrightarrow \big((\mathsf{pk}, m, \mathsf{ct}), (\mathsf{r}, \mathsf{sk})\big) \in \mathrm{R}_{\mathsf{dec}} \wedge m \notin \mathsf{cList}.$$

     We note that, unlike the other cryptographic primitives, $\pi_{\mathsf{Dec}}^{\perp}$ is most likely not used in a real election; see the decryption phase for details.

For all the above relations we may utilize either a zero-knowledge proof system or a witness-indistinguishable proof system for this aim.

- **EUF-CMA-secure signature scheme:** We assume that each message encrypted by a protocol participant contains some election parameters such as the election identifier, and to avoid cumbersome notation, we use the following convention: Signing a message m implies that the signature is computed on the tuple $(m; \mathsf{pp})$ where the second components are public parameters of the election, including an election identifier.

Observe that all of these primitives are standard in modern secure e-voting, except for the disjunctive NIZKP π_{Enc} (and the conjunctive NIZKP $\pi^{\perp}_{\mathsf{Dec}}$) which we demonstrate to construct using established cryptographic techniques in Sec. 8.5.

8.4.3 Protocol Framework

A protocol run consists of the following phases which we will explain in more detail further below:

- *Setup phase*: The voting authority $\mathfrak{Auth}$ generates basic parameters. All protocol participants generate their key material and publish the respective public parts. Each voter $\mathsf{v}_{\mathtt{id}}$ is assigned an initially empty vector $\mathsf{list}_{\mathtt{id}}$ of ciphertext/proof tuples.
- *Submission phase*: Voter $\mathsf{v}_{\mathtt{id}}$ pick her choice, encrypt them, prove correctness, and submit the resulting ciphertext/proof pair as a ballot to the posting trustee PT. Voters can re-vote to update their previously submitted choices. The posting trustee PT collects incoming ballots and publishes them on the bulletin board $\mathfrak{BB}$ periodically, together with several indistinguishable "noise" ballots for those voters who did not submit a ballot in the current period. Every voter $\mathsf{v}_{\mathtt{id}}$ who submitted a ballot to PT can verify whether her ballot appears in the $\mathsf{list}_{\mathtt{id}}$ published on the bulletin board $\mathfrak{BB}$.
- *Tallying phase (standard)*: The mix server $\mathfrak{mix.s}$ takes as input the last ciphertexts $\mathsf{ct}_{\mathtt{id}}$ from each ciphertext vector $\mathsf{list}_{\mathtt{id}}$, re-encrypts and shuffles all of them uniformly at random. Finally the decryption trustee $\mathfrak{DT}$ decrypts the mix server's outcome.
- *Verification phase (standard)*: Everyone can verify the correctness of the public material (ZKPs, complaints, etc.) published on the bulletin board, $\mathfrak{BB}$.
- **Setup Phase.** The election authority $\mathfrak{Auth}$ determines all election parameters and posts them on the bulletin board $\mathfrak{BB}$:
  - Security parameter 1^{ℓ},
  - List $\vec{\mathtt{id}}$ of eligible voters,
  - *Micro submission periods* $t_0, t_1, \ldots, t_e$ (t_0: starting time of submission phase, t_e: closing time of submission phase),
  - Election ID $\mathtt{id}_{\mathsf{election}}$,
  - The set of valid choices cList.

The decryption trustee $\mathfrak{DT}$ runs the key generation algorithm of the public-key encryption scheme Π^{pke} to generate its public/private (encryption/decryption) key pair $(\mathsf{pk},\mathsf{sk})$. Additionally, $\mathfrak{DT}$ creates a NIZKP π_{Kgen} to prove the validity of pk and posts $(\mathsf{pk},\pi_{\mathsf{Kgen}})$ on the bulletin board $\mathfrak{BB}$.

Each voter $\mathsf{v}_{\mathtt{id}}$ chooses a secret "signing" key $\mathsf{sk}_{\mathtt{id}} \leftarrow \{0,1\}^{\ell}$ uniformly at random and computes her public verification key as $\mathsf{pk}_{\mathtt{id}} \leftarrow f(\mathsf{sk}_{\mathtt{id}})$, where f is the one-way function mentioned above. The voter sends $\mathsf{pk}_{\mathtt{id}}$ to the bulletin board verifying whether $\mathsf{v}_{\mathtt{id}} \in \vec{\mathtt{id}}$. If this is the case, then $\mathfrak{BB}$ links $\mathsf{v}_{\mathtt{id}}$'s vector $\mathsf{list}_{\mathtt{id}}$ to $\mathsf{pk}_{\mathtt{id}}$ and initializes it as

$$\mathsf{list}_{\mathtt{id}} = \left[\mathsf{list}^{0}_{\mathtt{id}} = \mathsf{Enc}(\mathsf{pk},0;0),\epsilon\right].$$

- **Voting Phase.** We now describe the program run by an honest voter $\mathsf{v}_{\mathtt{id}}$ in the voting phase. We start with the program that $\mathsf{v}_{\mathtt{id}}$ runs when she is not under coercion. In this case, the $\mathsf{v}_{\mathtt{id}}$ picks her favorite choice $m \in \mathsf{cList}$ and encrypts it under pk to obtain

  $$\mathsf{ct}' \leftarrow \mathsf{Enc}(\mathsf{pk},m;r).$$

  Next, $\mathsf{v}_{\mathtt{id}}$ reads the latest status of her vector $\mathsf{list}_{\mathtt{id}}$, including the currently last ciphertext ct in $\mathsf{list}_{\mathtt{id}}$, from the bulletin board $\mathfrak{BB}$ and uses $\mathsf{sk}_{\mathtt{id}}$, message m, and randomness r to create a NIZKPoK π_{Enc} for $\mathrm{R}_{\mathsf{ballot}}(x,w)$ detailed in follows:

  $$\begin{aligned}
  &\mathrm{R}_{\mathsf{ballot}}(x,w) = \mathsf{True} \iff (\mathsf{P}_1(x,w_1) = \mathsf{True}) \lor (\mathsf{P}_1(x,w_2) = \mathsf{True})\\
  &x = \left[\mathsf{list}^{\mathsf{pk}_{\mathtt{id}}} = [\mathsf{pk}_{\mathtt{id}},\mathsf{ct}_0,(\mathsf{ct}_1,\pi_1),\ldots,(\mathsf{ct}_j,\pi_j)],\mathsf{ct}'\right],\\
  &w_1 = (m,\mathsf{r}_1,\mathsf{sk}_{\mathtt{id}}),\ w_2 = (\mathsf{r}_2)\\
  &\mathsf{P}_1(x,w_1) = \mathsf{True} \iff (\mathsf{ct}' = \mathsf{Enc}_{\mathsf{PK}}(m;\mathsf{r}_1)) \land (\mathsf{vote} \in \mathsf{cList}) \land \mathsf{PoK}(\mathsf{sk}_{\mathtt{id}})\\
  &\mathsf{P}_2(x,w_2) = \mathsf{True} \iff \left(\mathsf{ct}' = \mathsf{Re.Enc}_{\mathsf{PK}}(\mathsf{ct}_j;\mathsf{r}_2)\right)
  \end{aligned}$$

  Observe that $\mathsf{v}_{\mathtt{id}}$ actually proves the right term of the disjunctive relation $\mathrm{R}_{\mathsf{enc}}$ (see above), i.e., that ct' is a "fresh" encryption of some message m *and* the voter (prover) knows a valid secret signing key $\mathsf{sk}_{\mathtt{id}}$ for public verification key $\mathsf{pk}_{\mathtt{id}}$. Afterwards, $\mathsf{v}_{\mathtt{id}}$ sends $(\mathsf{ct}',\pi_{\mathsf{Enc}})$ to the posting trustee PT. The voter can then verify whether $\mathsf{list}_{\mathtt{id}}$ published by $\mathfrak{BB}$ contains her latest submission $(\mathsf{ct}',\pi_{\mathsf{Enc}})$ after the current micro submission period has passed.

  Let us now describe an honest voter's program, i.e., the voter's counter-strategy, when she is under coercion. This counter-strategy is straightforward in DeVoS: wait until the coercer has left and then execute the same program that you run when you are not under coercion (see above), or vote again even more concisely.

  We now describe the program of the posting trustee PT. Assume that we are in the micro submission period t_l. For each incoming ballot $(\mathsf{ct}',\pi_{\mathsf{Enc}})$ by some authenticated voter $\mathsf{v}_{\mathtt{id}}$, the posting trustee, verifies the correctness of $(\mathsf{ct}',\pi_{\mathsf{Enc}})$ w.r.t. the latest status of $\mathsf{list}_{\mathtt{id}}$ published on $\mathfrak{BB}$. If the ballot is not correct or $\mathsf{v}_{\mathtt{id}}$ has already sent a ballot in the current period, t_l, then PT ignores the ballot. Otherwise, PT internally adds $(\mathsf{ct}',\pi_{\mathsf{Enc}})$ to $\mathsf{list}_{\mathtt{id}}$, i.e., updates $\mathsf{list}_{\mathtt{id}}$ accordingly. Once period t_l has closed, the posting trustee PT internally adds a "noise" ballot, i.e., a re-encryption ct' of the last ciphertext ct plus a NIZKP π_{Enc}, to the ballot vector $\mathsf{list}_{\mathtt{id}}$ of each voter $\mathsf{v}_{\mathtt{id}}$ who did not submit a (valid) ballot during t_l. The posting trustee PT then sends the latest status of all $\mathsf{list}_{\mathtt{id}}$ to the bulletin board $\mathfrak{BB}$, which updates its public content accordingly.

- **Tally Phase.** This part is essentially standard: the final ciphertexts are first shuffled and then decrypted, both in a publicly verifiable way. More precisely, after the submission phase has closed, mix server $\mathfrak{mix.s}$ reads all vectors $\mathsf{list}_{\mathtt{id}}$ from the bulletin board and verifies their correctness (ZKPs etc.). If one of these vectors is not correct, then $\mathfrak{mix.s}$ aborts. Otherwise, $\mathfrak{mix.s}$ chooses a permutation σ uniformly at random over $[n_{\mathsf{v}}]$, extracts the last ciphertext $\mathsf{ct}_{\mathtt{id}}$ from each vector $\mathsf{list}_{\mathtt{id}}$, then shuffles and re-encrypts $\mathsf{ct}_{\mathtt{id}}$ into

  $$\mathsf{ct}'_{\mathtt{id}} \leftarrow \mathsf{ReEnc}(\mathsf{pk}, \mathsf{ct}_{\sigma(i)})$$

  Eventually, $\mathfrak{mix.s}$ creates a NIZKP π_{Shuffle} for proving that the resulting shuffled and re-encrypted ciphertext vector list' was generated correctly (without revealing the actual links between input and output ciphertexts) and sends $(\mathsf{list}', \pi_{\mathsf{Shuffle}})$ to the bulletin board $\mathfrak{BB}$.

  Afterwards, the decryption trustee $\mathfrak{DT}$ reads $(\mathsf{list}', \pi_{\mathsf{Shuffle}})$ from the bulletin board and verifies its correctness. If π_{Shuffle} is not correct, then $\mathfrak{DT}$ aborts. Otherwise, $\mathfrak{DT}$ uses its secret key sk to decrypt each ciphertext $\mathsf{ct}' \in \mathsf{list}'$ into m'. If m' is a valid choice, i.e., $m' \in \mathsf{cList}$, then $\mathfrak{DT}$ creates a NIZKP π_{Dec} for proving that it decrypted ct' correctly (in practice, these individual proofs would be batched for efficiency reasons), and sends the result to the bulletin board $\mathfrak{BB}$. Otherwise, if $m' \notin \mathsf{cList}$, then $\mathfrak{DT}$ creates a NIZKP $\pi^{\perp}_{\mathsf{Dec}}$ for proving that ct' decrypted to an invalid message, and publishes $\pi^{\perp}_{\mathsf{Dec}}$, without revealing plaintext m'. The reason why $\mathfrak{DT}$ must not publish invalid messages is to protect against *forced abstention*. Otherwise, a coercer could instruct a voter to submit a "vote" for some random invalid m'. In this way, the coerced voter would effectively abstain from voting and the coercer could check whether the voter obeyed by verifying whether m' appears in the final result. An alternative protection against forced abstention to the one chosen in DeVoS could be to extend π_{Enc} such that it also proves that the encrypted message belongs to cList. We did not follow this approach because the efficiency of the submission phase would then depend on the complexity of cList, which would be particularly undesirable in DeVoS where π_{Enc} is computed often. We expect that in real election generating $\pi^{\perp}_{\mathsf{Dec}}$ will not be necessary; the mere existence of $\pi^{\perp}_{\mathsf{Dec}}$ prevents a possible coercer from executing the aforementioned forced abstention attack. The final list of plaintexts $\vec{m}'$ is supposed to contain all voters' choices $\vec{m}$ in random order.

- **Public Verification Phase.** In this phase, every participant, including the voters or external observers, can verify the correctness of the previous phases, particularly the correctness of all NIZKPs published during the setup, voting, and tallying phase.

8.5 Protocol Instantiation

This section provides two instantiations of the DeVoSprotocol with concrete cryptographic primitives. The ElGamal cryptosystem is used in both protocols as underlying public key encryption scheme, although the proof systems are different.

We stress that we explain only the basic building blocks and their algorithm and suppress some details about ballot integrity and non-malleability from the zero-knowledge proofs, e.g. the inclusion of election identifiers and the correct form of the Fiat-Shamir transformations.

8.5.1 Instantiation with Bilinear Groups

The first instantiation relies on the standard threshold variant of ElGamal cryptosystem over bilinear map (See 1) where its security is based on the hardness of the SXDH assumption 8 and the Groth-Sahai NIWI-proof system [143].

The main point of using those primitives in this instantiation is that, ElGamal is a homomorphic encryption scheme that can be efficiently implemented in a bilinear group. A bilinear map allows us to do the disjunction proof with the efficient NIWI-proof system. We stress that instead of the standard ElGamal encryption Scheme 2.7.2.1 we use the following version:

Definition 49 (ElGamal Encryption Scheme in Bilinear Setting). *Considering the group generator*

$$\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \mathbf{e}, \mathbb{G}_T) \leftarrow \mathsf{GroupGen}(1^\lambda),$$

we define ElGamal encryption scheme, $\Pi^{\mathsf{ElGamal}} = \langle \mathsf{Kgen}, \mathsf{Enc}, \mathsf{Dec} \rangle$ *for message space* $\mathcal{M} = \mathbb{Z}_p$*, ciphertext space,* $\mathcal{C} = \mathbb{G}_1^2$ *as follow:*

- ***Key Generation:***
  1. *Run* $\mathsf{GroupGen}(1^\ell)$ *to generate* $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \mathbf{e}, \mathbb{G}_T)$.
  2. *Select a random integer* x *from* $\mathbb{Z}_p$.
  3. *Set* $h_1 = g_1^x$.
  4. *Set* $\mathsf{pk} = (\mathbb{G}_1, p, g_1, h_1)$ *and* $\mathsf{sk}_{ElGamal} = x$.
  5. *Return* $\mathsf{Key} = (\mathsf{pk}, \mathsf{sk})$.
- ***Encryption:*** *For message* $m \in \mathbb{Z}_p$*:*
  1. *Select a random number* $\mathsf{random} \in \mathbb{Z}_p$.
  2. *Set the ciphertext* $\mathsf{CT} = (g_1^{\mathsf{random}}, h_1^{\mathsf{random}} \cdot g_1^m)$
  3. *Return* CT.
- ***Decryption:*** *For ciphertext* $\mathsf{CT} = (u, v) \in \mathbb{G} \times \mathbb{G}$*:*
  1. *Compute the discrete logarithm of* $g^{m'} = \log_{g_1}(u^{-\mathsf{sk}} \cdot v)$.
  2. *Return* m'.

Theorem 8.5.1. *The SXDH assumption. 8 results in the semantic security of the above version of ElGamal encryption.*

As we show in 2.7.2.1 the ElGamal cryptosystem allows re-encryption.

- IND-CPA-secure PKE scheme Π^{pke}: We instantiate this central primitive with ElGamal over bilinear setting $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \mathbf{e}, \mathbb{G}_T) \leftarrow \mathsf{GroupGen}(1^\ell)$ as defined above.

- NIZKP of correct key generation π_{Kgen}:

$$\mathrm{R}_{\mathsf{KeyGen}} = \{(x, w) : x = (g_1, h_1), w, h_1 = g_1^w\}$$

Since g_1 is a generator of group $\mathbb{G}_1$, for any $h_1 \in \mathbb{G}_1$ there exist some u such that $(g, h) \in \mathrm{R}_{\mathsf{KeyGen}}$. In particular, we do not need a NIZKP because we can simply check whether g_1 and h_1 are group members and g_1 is a generator of $\mathbb{G}_1$.

- One-way function f: We choose modular exponentiation in $\mathbb{G}_2$ concerning g_2, i.e., $f\colon \mathbb{Z}_p \mapsto \mathbb{G}_2$ with $x \mapsto g_2^x$.

- NIZKPoK of correct encryption π_{Enc}: The specific relation to be proven by the voter $\mathtt{v_{id}}$ with key pair $(\mathsf{pk_v}, \mathsf{sk_v}) = (h_\mathsf{v} = g_2^{x_\mathsf{v}}, x_\mathsf{v})$ is:

$$\mathrm{R}_{\mathsf{ballot}} = \Big\{(x, w) = \Big(x = \big(\mathsf{ct} = (u, v), \mathsf{ct}' = (u', v'), \mathcal{G}, h_1, h_v\big), w = (\mathsf{r}, m, x_v)\Big) :$$
$$(h_v = g_2^{x_v}, u' = g^{\mathsf{r}}, v' = h_1^{\mathsf{r}} \cdot g_1^m) \vee (u' = u \cdot g_1^r, v' = v \cdot h_1^r)\Big\}.$$

- NIZKPoK of correct shuffling π_{Shuffle}: Since there exist different options in the literature to instantiate this primitive for ElGamal PKE (e.g., [246, 145]), we do not single any of them out.

- NIZKP of correct decryption π_{Dec} and NIZKP of encryption to invalid message $\pi_{\mathsf{Dec}}^{\perp}$: We describe specific NIZKPs in App. 8.5.2.2.

8.5.1.1 NIWI Proof Systems for DeVoS

We recall that in the voting phase the voter or posting trustee must provide the proof related to the $\mathrm{R}_{\mathsf{ballot}}$ 8.4.3. Now considering the Groth-Sahai proof system for relation R_{GS} 8.5.1.1:

$$\begin{aligned} \mathrm{R}_{GS} = \Big\{&(x, w) : \\ &x = \mathsf{Eq} \in \mathcal{GS}^{\mathsf{Eq}}, \\ &w = (\{u_i\} : u_i \in (\mathbb{G}_1 \cup \mathbb{G}_2, \mathbb{G}_T) \\ &\mathsf{Eq}[w] = \mathsf{Eq}[\{g_i\}] = \mathsf{True} \\ &\Big\} \end{aligned}$$

the relation $\mathrm{R}_{\mathsf{ballot}}$ has the following form:

$$\begin{aligned} \mathrm{R}_{\mathsf{ballot}} = \Big\{&(x, (w_1, w_1)) : \\ &x = \big(\mathsf{pk}_i, \mathsf{list}^{\mathsf{pk}_i} = [\mathsf{ballot}_0, (\mathsf{ballot}_1, \pi_1), \ldots, (\mathsf{ballot}_{j-1}, \pi_{j-1})], \mathsf{ballot}_j\big), \\ &w_1 = (m, \mathsf{r}, \mathsf{sk}_i), w_2 = \mathsf{r}_2, \\ &\Big(\mathsf{Eq}_{\mathsf{vote.validity}}[w_1]) \wedge \mathsf{Eq}_{\mathsf{freshVote}}[w_1]\Big) \vee \mathsf{Eq}_{\mathsf{reEncVote}}[w_2]\Big) = \mathsf{True}; \\ \Big\}& \end{aligned} \tag{8.3}$$

We now describe π_{ballot} for fresh-vote, re-Encrypt vote and the vote's validity (the vote is in the list of the candidates) by describing each equation.

Equation Description.

1. $\mathsf{Eq}_{\mathsf{freshVote}}$: Validity of a fresh vote from the voter,

$$\mathsf{v}_{\mathtt{id}} = (\mathsf{pk}_i d = h_{\mathtt{id}}, \mathsf{sk}_{\mathtt{id}} = x_{\mathtt{id}})$$

as describe in 8.3. The proof for this relation can be generated by R_{GS} concerning the following equation:

$$\text{Variables:} \quad \mathcal{X} = g_1^{\mathsf{r}_1}, \mathcal{Y} = g_2^{\mathsf{r}_1}, \mathcal{H}_{\mathtt{id}} = h_{\mathtt{id}}^{\mathsf{r}_1}, \mathcal{H}^*_{\mathtt{id}} = g_2^{x_{\mathtt{id}}}, \mathcal{M}_1 = g_1^m$$

$$\mathsf{Eq}_{\mathsf{freshVote}}: \begin{cases} \mathbf{e}(u, g_2) = \mathbf{e}(g_1, \mathcal{Y}) = \mathbf{e}(\mathcal{X}, g_2) \\ \mathbf{e}(v, g_2) = \mathbf{e}(\mathcal{H}_{\mathtt{id}}, g_2) \cdot \mathbf{e}(\mathcal{M}_1, g_2) = \mathbf{e}(h_{\mathtt{id}}, \mathcal{Y}) \cdot (\mathcal{M}_1, g_2) \\ \mathbf{e}(h_{\mathtt{id}}, g_2) = \mathbf{e}(g_1, \mathcal{H}^*) \end{cases} \tag{8.4}$$

2. Validity of a re-encrypt vote as described in 8.3. The proof for this relation can be generated by R_{GS} concerning the following equation:

$$\text{Variables:} \quad \mathcal{X}^* = g_1^{r^*}, \mathcal{Y}^* = g_2^{r^*}, \mathcal{H}^*_{\mathtt{id}} = h_{\mathtt{id}}^{r^*}$$

$$\mathsf{Eq}_{\mathsf{reEncVote}}: \begin{cases} \mathbf{e}(u_{i-1}, g_2) \cdot \mathbf{e}(\mathcal{X}^*, g_2) = e(u_i, g_2) \\ \mathbf{e}(v_{i-1}, g_2) \cdot \mathbf{e}(\mathcal{H}^*_{\mathtt{id}}, g_2) = e(v_i, g_2) \\ \mathbf{e}(\mathcal{X}^*, g_2) = \mathbf{e}(g_1, \mathcal{Y}^*) \\ \mathbf{e}(\mathcal{H}^*_{\mathtt{id}}, g_2) = \mathbf{e}(h_{\mathtt{id}}, \mathcal{Y}^*) \end{cases} \tag{8.5}$$

3. Validity of voter choice: the encrypted vote is in the candidate list. The proof for this relation can be generated by R_{GS} with respect to the following equation:

$$\text{Variables:} \quad \mathcal{X} = g_1^{\mathsf{r}_1}, \mathcal{Y} = g_2^{\mathsf{r}_1}, \mathcal{H}_{\mathtt{id}} = h_{\mathtt{id}}^{\mathsf{r}_1}$$
$$\mathcal{M}_i = g_1^{m^i} : i = 1, 2, \ldots, k, \mathcal{M}' = g_2^m$$

$$\mathsf{Eq}_{\mathsf{vote.validity}}: \begin{cases} \mathbf{e}(v_j, g_2) = \mathbf{e}(\mathcal{H}_{\mathtt{id}}, g_2) \cdot \mathbf{e}(\mathcal{M}_1, g_2) = \mathbf{e}(h_2, \mathcal{Y}) \cdot (\mathcal{M}_1, g_2) \\ \mathbf{e}(\mathcal{M}_k, g_2) \cdot \mathbf{e}(\mathcal{M}_{k-1}, g_2)^{p_1} \cdot \ldots \cdot \mathbf{e}(\mathcal{M}_1, g_2)^{p_{k-1}} \cdot \mathbf{e}(g_1, g_2)^k = 1_{\mathbb{G}_T} \\ \forall i = 1, 2, \ldots, k : \mathbf{e}(\mathcal{M}_i, g_2) \cdot \mathbf{e}(\mathcal{M}_{i-1}, \mathcal{M}')^{-1} = 1_{\mathbb{G}_T} \\ \mathbf{e}(\mathcal{M}, g_2) = \mathbf{e}(g_1, \mathcal{M}') \end{cases} \tag{8.6}$$

Correctness of the Equations. Now we show that each equation, is equivalent to the related relation of the system of equation and the relation Ṛ:

1. Validity of the Vote: Let's consider candidate list as a set of integers less than p. Then, to prove that $m \in \mathsf{CandidateList}$ it would be enough if we prove m is the root of the following publicly known polynomial:

   $\mathsf{Poly}_{\mathsf{cList}}(x) = (x - c_1) \cdot (x - c_2) \cdot \ldots (x - c_k) = x^k + p_1 x^{x-1} + \ldots + p_k,$

   Hence, if we define variables $\mathcal{M}_i = g^{m^i}$, the equation 8.6 proves that m is one of the zeros of the polynomial and hence it is a valid choice.

2. Correctness of Fresh-vote and voter signature: A valid proof for the system of equation $\mathsf{Eq}_{\mathsf{freshVote}}$ claims that values y_i, x_i, m_i, h exist, such that $\mathcal{X} = g_1^{x_i}, \mathcal{Y} = g_2^{y_i}$ For

simplicity we consider the equation without index j. A valid proof for the system of equation $\mathsf{Eq}_{\mathsf{freshVote}}$ claims that there exists values r, y, x, m such that

$$u = g_1^{\mathsf{r}}, \mathcal{X} = g_1^{x}, \mathcal{Y} = g_2^{y}, \mathcal{M} = g_1^{m}, \mathcal{H}^* = g_2^{x_{\mathtt{id}}}.$$

Then we have:

$$\begin{aligned}
&1. \quad \mathbf{e}(u, g_2) = \mathbf{e}(g_1, \mathcal{Y}) = \mathbf{e}(\mathcal{X}, g_2): \\
&\quad\ \ \mathbf{e}(g_1^{\mathsf{r}}, g_2) = \mathbf{e}(g_1, g_2^{y}) = \mathbf{e}(g_1^{x}, g_2) \implies \mathsf{r} = x = y \\
&2. \quad \mathbf{e}(v, g_2) = \mathbf{e}(h_{\mathtt{id}}, \mathcal{Y}) \cdot \mathbf{e}(\mathcal{M}_1, g_2), \\
&\quad\ \ \mathbf{e}(v, g_2) = \mathbf{e}(h_{\mathtt{id}}, g_2^{\mathsf{r}}) \cdot \mathbf{e}(g_1^{m}, g_2) \implies v = h_{\mathtt{id}}^{\mathsf{r}} \cdot g_1^{m} \\
&3. \quad \mathbf{e}(h_{\mathtt{id}}, g_2) = \mathbf{e}(g_1, \mathcal{H}^*) \implies \mathcal{H}^* = g_2^{x_{\mathtt{id}}}
\end{aligned} \tag{8.7}$$

The above computation shows that $u = g^{\mathsf{r}}, v = h_{\mathtt{id}}^{\mathsf{r}} \cdot g_1^{m}$ which prove the well-formedness of the encryption and it also shows that the voter knows the value $x_{\mathtt{id}}$ such that $h_{\mathtt{id}} = g_1^{x_{\mathtt{id}}}$:

8.5.2 Instantiation with Exponential ElGamal

We propose the following instantiation of the abstract DeVoS protocol (Sec. 8.4):

- IND-CPA-secure PKE scheme Π^{pke}: We instantiate this central primitive with exponential ElGamal PKE [112]. If the decisional Diffie-Hellman (DDH)(see 5) problem is hard relative to $\mathbb{G}$, then (exponential) ElGamal is IND-CPA-secure. To re-randomize an ElGamal ciphertext (u, v), choose $s \in \mathbb{Z}_p$ uniformly at random and return $(u', v') \leftarrow (u \cdot g^s, v \cdot h^s)$.

- NIZKP of correct key generation π_{Kgen}:

$$\mathrm{R}_{\mathsf{KeyGen}} = \{(x, w) : x = (g, h), w = u, h = g^u\}$$

  Since g is a generator of group $\mathbb{G}$, for any $h \in \mathbb{G}$ there exist some u such that $(g, h) \in \mathrm{R}_{\mathsf{KeyGen}}$. In particular, we do not need a NIZKP because we can simply check whether g and h are group members and g is a generator of $\mathbb{G}$.

- One-way function f: We choose modular exponentiation in $\mathbb{G}$ with respect to g, i.e., $f\colon \mathbb{Z}_p \to \mathbb{G}$ with $x \mapsto g^x$.

- NIZKPoK of correct encryption π_{Enc}: The specific relation to be proven by the voter v with key pair $(\mathsf{pk}_v, \mathsf{sk}_v) = (h_v = g^{u_v}, u_v)$ is:

$$\begin{aligned}
\mathrm{R}_{\mathsf{enc}} = \Big\{(x, w) = \Big(x = \big(\mathsf{ct} = (u, v), \mathsf{ct}' = (u', v'), g, h, h_v\big), w = (r, m, u_v)\Big):& \\
(h_v = g^{u_v}, u' = g^r, v' = h^r \cdot g^m) \vee (u' = u \cdot g^r, v' = v \cdot h^r)\Big\}.&
\end{aligned}$$

  Our proof system is based on Schnorr's signature [233] and existing techniques for proving relations between group elements [77, 68] implemented in bilinear groups of size 256 bits.

- NIZKPoK of correct shuffling π_{Shuffle}: Since there exist different options in the literature to instantiate this primitive for ElGamal PKE (e.g., [246, 145]), we do not single one of them out.

- NIZKP of correct decryption π_{Dec} and NIZKP of encryption to invalid message $\pi^{\perp}_{\mathsf{Dec}}$: We describe specific NIZKPs in App. 8.5.2.2.

8.5.2.1 Non-Interactive Sigma-Protocol

Considering the hardness of the DDH assumption(see 5) in group $\mathbb{G}$, the following protocol is a zero-knowledge proof of knowledge proof system for R_{enc}:

FIGURE 8.1: Non-Interactive Sigma protocol for R_{enc} DDH-Relation

• Setting:	$\mathcal{G} = (G, g, p) : \mathbb{G} = \langle g \rangle; \vert G \vert = p$, $(x, w) : x = \big(g, h, h_i, \mathsf{ct}_1 = (u_1, v_1), \mathsf{ct}_2 = (u_2, v_2)\big), w = (\alpha_i, r, m)$ $\mathrm{R}_{enc} = \{(x, w) :$ $(h_i = g^{\alpha_i}, u_2 = g^r, v_2 = h^r g^m)$ $\vee (u_2 = u_1 g^r, v_2 = v_1 h^r)\}$
• Protocol:	$\langle \mathsf{Prove}(x, w), \mathsf{Verify}(x) \rangle$
Prover	
	1. Choose $e_2, z, a_1, a_2, a_3 \leftarrow \mathbb{Z}_p$ 2. Compute: $\bar{u} = g^{a_1}, \bar{v} = h^{a_1} g^{a_2}$, $\bar{h}_i = g^{a_3}$, $u^* = g^{z_2}(u_2 \cdot u_1^{-1})^{e_2}$, $v^* = h^{z_2}(v_2 \cdot v_1^{-1})^{e_2}$ Set:
Replaced with Verifier challenge	$e = \mathsf{hash}(\bar{u}, \bar{v}, \bar{h}_i, u^*, v^*; nonce)$, $e_1 = e - e_2; z_1 = a_1 - e_1 r, z_2 = a_2 - e_1 m, z_3 = a_3 - e_1 \alpha$ Prover sends $\pi = (e_1, e_2, z_1, z_2, z_3, z)$ to the verifier $\mathsf{Prove} \xrightarrow{\pi = (e_1, e_2, z_1, z_2, z_3, z)} \mathsf{Verify}$
Verification	
	1. Compute: $U_1 = g^{z_1} u_2{}^{e_1} \cdot u_1^{-e_1}, V_1 = h^{z_1} \cdot v_2^{e_1} v_1^{-e_1}$, $U_2 = g^z (u_2 \cdot u_1)^{e_2}, V_2 = h^z (v_2 \cdot v_1)^{e_2}$ 2. Set $e' = \mathsf{hash}(U_1, V_1, h_i, U_2, V_2)$ 3. $\mathsf{Verify}(x, \pi)$ accepts if and only if $e' = e_1 + e_2$

8.5.2.2 Decryption Trustee Proofs

We describe the standard techniques for proving the correct decryption and proving that the encrypted value does not belong to a pre-defined set of values.

Proof of correct decryption, π_{Dec}. The goal is to prove that the ciphertext $\mathsf{ct} = (u, v)$ can be decrypted to $m \in \mathsf{cList}$. If this is the case, the decryption trustee $\mathfrak{DT}$ computes $\bar{v} = u^{\mathsf{sk}}$ and publishes $\bar{v}$ along with a NIZKPoK proof π_{Dec_1} for the following relation:

$$\mathrm{R}_{\mathsf{Dec1}} = \{(x = (g, \mathsf{pk}, (u, \bar{v})), w = \mathsf{sk}) : (pk, \bar{v}) = (g^{\mathsf{sk}}, u^{\mathsf{sk}})\} \tag{8.8}$$

The verifier verifies π_{Dec_1} and checks whether $v(\bar{v})^{-1} =^? g^m$.

Note that $\mathrm{R}_{\mathsf{KeyGen}}$ in ElGamal cryptosystems can be easily checked. As long as the public key is a member of the underlying group other than the identity element it implies the correctness of the key.

Proof of invalid plaintext, $\pi^{\perp}_{\mathsf{Dec}}$. The goal is to prove that the ciphertext $\mathsf{ct} = (u, v)$ cannot be decrypted to a message in cList, i.e., it encrypts some $m' \not\in \mathsf{cList}$. We note that the technique for proving the correct decryption presented above cannot be used here, since it reveals the plaintext (in the form of g^m). The verifier should only learn that the ciphertext cannot be decrypted to any value from the set cList and nothing else. We first show a technique for proving that the ciphertext cannot be decrypted to some m' and then extend it to the whole set cList. The technique is a slight modification of *Plaintext Equivalence Test* (PET) [162], the latter is used in a distributed setting and allows parties to prove that a pair of ElGamal ciphertexts encrypt the same plaintext as long as at least one party is honest.

To prove that (u, v) cannot be decrypted to m', the prover does the following:

1. Pick a random value $r \leftarrow \mathbb{Z}_p^*$ and compute

   $$\bar{v} = v \cdot g^{-m'}, \; u^* = u^r, v^* = \bar{v}^r.$$

   Then set $x_1 = ((u, \bar{v}), (u^*, v^*))$, $w_1 = r$ and create a NIZKPoK proof $\pi^{\sigma} = \pi^{\sigma}(x_1, w_1)$.

2. Compute $\bar{v}^* = (u^*)^{\mathsf{sk}}$, then set

   $$x_2 = \big(g, \mathsf{pk}, (u^*, \bar{v}^*)\big), w_2 = \mathsf{sk}$$

   and create a decryption proof $\pi_{\mathsf{Dec}_1} = \pi_{\mathsf{Dec}_1}(x_2, w_2)$ for the relation $\mathrm{R}_{\mathsf{Dec1}}$ (Equation (8.8)).

3. Send $(m', u^*, v^*, \bar{v}^*)$, along with π_{Dec_1} and π^{σ} to the verifier.

The verifier recomputes $\bar{v} = v/g^{m'}$, verifies the proofs π^{σ} and π_{Dec_1}, then accepts the proof of invalid plaintext if $v^* \neq \bar{v}^*$.

Let us analyze the above proof system for two cases: 1) the ciphertext decrypts to m', and 2) the ciphertext does not decrypt to m'. If (u, v) decrypts to m', then we have that $(v/g^{m'})^r$ cancels out the message term and (u^*, v^*) is encryption of the identity. It follows that $v^* = \bar{v}^*$. If (u, v) does not decrypt to m', then $(u, v/g^{m'})$ will be an encryption of some message $m'' = m - m' \neq 0$. Raising it to r will yield an encryption (u^*, v^*) of $m^* = m'' \cdot r$. It follows that $v^* \neq \bar{v}$, and the verifier will learn $g^{m^*} = g^{m'' \cdot r}$. Since r is fresh and kept secret, it effectively masks the original plaintext encrypted in (u, v).

To prove that a ciphertext (u, v) does not decrypt to any message from cList, the decryption trustee $\mathfrak{DT}$ proves individually for each $m' \in \mathsf{cList}$ that the ciphertext does not decrypt to m' using the technique above.

8.5.2.3 Practical Efficiency

In order to evaluate DeVoS's efficiency in handling large-scale real elections, we implemented the system and here we briefly discuss the protocol's efficiency.[3]

Since all phases of DeVoS except for the submission phase are essentially standard, we focus on the latter phase to demonstrate DeVoS' practical efficiency. To this end, we implemented the ZKP that both the voters and the posting trustee compute to prove that a submitted ballot is valid. Our benchmarks show that this ZKP has a small size and can be computed quickly, which is important because the posting trustee PT computes such a ZKP for each noise ballot that it publishes.

Interestingly, we can further optimize the efficiency of PT and using techniques from the field of differential privacy (DP), we prove that DeVoS still achieves a reasonable level of deniable vote updating when PT adds dummy ballots for only a fraction of those voters who did not submit a ballot in a given micro submission phase.

8.6 DeVoS; Security Properties

In this part, we conduct a security analysis on our scheme by providing the security model, stating the security criteria we wish to guarantee, and determining the security assumptions needed to support them.

8.6.1 Security Model

In our protocol, we assume the following trust assumptions:

1. The adversary is computationally bounded.
2. In case of using the threshold cryptosystem we require that the majority of the election trustee and also the majority of the supervised registrar are trustworthy. Namely, the adversary cannot corrupt a threshold set of election trustee. In case of single party, we assume the election trustee and the registrar are honest.
3. At least one mix server is honest.
4. There is a point in the voting phase, where the adversary cannot control the voter.
5. Voter supporting device, vsd cannot be tampered with or compromised.
6. Voter's vsd does not leak the voter's private credentials to the adversary.
7. The adversary cannot control the voter's computer or voter's device. considering the Benaloh challenges in the protocol this assumption will reduce to: The adversary cannot control the voting and the verification environments simultaneously [206].
8. The channel to the ballot boxes is anonymous.
9. The voter's credential is not leaked to the adversary without voter's knowledge.

[3]Due to the fact that the author of this thesis was not the primary contributor to this part, we only refer to the subject and will not give the details in this manuscript

To evaluate the security properties, i.e., public end-to-end verifiability and vote privacy that our scheme provides, we perform an evaluation. We refer to 6.2.1 for the formal verifiability theorem, which is proof based on the KTV framework. Here we briefly elaborate on the privacy.[4]

Recall that the purpose of PT is to add some noise to the set of submitted ballots which hides the voters' re-voting pattern. This means that we trust PT for coercion-resistance. We stress that, unlike other coercion-resistant e-voting protocols (e.g., VoteAgain [197]), DeVoS does not introduce any additional trust assumptions for end-to-end verifiability and vote privacy.

To see why this important feature of DeVoS holds true, recall that any noise ciphertext ct_i^j contains a vote for the same choice as ct_i^{j-1} does because, due to the soundness of π_i^j, only v_i herself can submit a "fresh" ciphertext ct_i^j that is not a re-randomization of ct_i^{j-1}. Particularly, even if the posting trustee PT is malicious, the vote contained in ct_i^{j-1} cannot be tampered with undetectably. Therefore, DeVoS does not introduce any additional trust assumptions for end-to-end verifiability (Theorem 8.6.2) compared to basic modern secure e-voting systems (e.g., Helios [10]). The same holds true for vote privacy, essentially because the program of PT does not need any secret inputs and could, in principle, be executed by any external party.

8.6.2 Privacy

We analyze the vote privacy of DeVoS based on the KTV framework(see 6.2.1). We show that the privacy level of DeVoS is ideal under the following (minimal) assumptions:

A.1 The PKE scheme with re-randomization is IND-CPA-secure.

A.2 The function $f\colon \{0,1\}^* \to \{0,1\}^*$ is one-way.

A.3 $\pi_{\mathsf{Kgen}}, \pi_{\mathsf{Shuffle}}, \pi_{\mathsf{Dec}}, \pi_{\mathsf{Dec}}^{f}, \pi_{\mathsf{Dec}}^{\perp}$ are NIZKs and π_{Enc} is a NIZKPoK.

A.4 The scheduler $\mathfrak{s}$, the bulletin board $\mathfrak{BB}$, the mix server $\mathfrak{mix.s}$, the decryption trustee $\mathfrak{DT}$, and at least $n_{\mathsf{v}}^{\mathsf{honest}}$ voters are honest.

and we prove the following theorem:

Theorem 8.6.1 (Privacy)**.** *Under the assumptions **A.1** to **A.4** stated above, the voting protocol* $\langle n_{\mathsf{v}}, \mathsf{Agent}, \Pi^{\mathsf{h}}, \mathsf{cList}, \Gamma^{\mathsf{elc}}, f_{\mathsf{res}} \rangle$ $\mathsf{P}_{\mathsf{DeVoS}} = \langle n_{\mathsf{v}}, \mathsf{Agent}, \Pi^{\mathsf{h}}, \mathsf{cList}, \Gamma^{\mathsf{elc}}, f_{\mathsf{res}} \rangle$ *achieves a privacy level of* $\delta^{\mathsf{ideal}}_{(n_{\mathsf{v}}^{\mathsf{honest}},\mathsf{C},\mu)}$.[4]

8.6.3 Intuitive Counter-Strategy

As described in Sec. 8.1, in a coercion-resistant e-voting system, each coerced voter has the option to run some counter-strategy instead of obeying the coercer. Similarly to VoteAgain [197], the counter-strategy voters have to run in DeVoS is as simple as possible: wait until the coercer has left and then execute the same program that you run when you are not under coercion, i.e., submit a ballot for your favorite choice. Or, even more concisely: vote again.

An extensive usability study of this counter-strategy, which would have been clearly beyond the scope of this cryptographic paper, will be interesting future work.

[4]Due to the fact that the author of this thesis was not the primary contributor to this part, we only refer to the subject and will not give the details in this manuscript

8.6.4 Coercion Threat Model

We consider an "over-the-shoulder" coercer [79] who can be physically present at a voter's location at some point of the submission phase. The coercer can instruct the voter to submit a vote for a particular candidate using the voting program; the coercer can check on-site whether the voter is following his instructions.

In order to secretly overwrite the coerced vote, we assume that the voter can submit a new ballot after the coercer has left the voter's place before the submission phase closes. Although the latter assumption may not hold true for *all* coerced voters, it is realistic to assume that *most* coerced voters can submit a new vote after coercion. In this way, the malicious impact of a coercer on the election result remains limited.

We note that all e-voting systems with deniable vote updating (e.g., [180, 197]) assume that voters have the opportunity to submit an uncoerced vote at some point of the submission phase.

We do *not* consider a coercer who can make a voter run a malicious script on the voting device because (we conjecture that) coercion-resistance is impossible in such cases. Related to this restriction, we assume that a coerced voter does not reveal her private credentials to the coercer; if the coercer could learn these credentials, then they could easily vote on behalf of the voter.

In order to realize this assumption in practice, one could generate and store the credential in a trusted execution environment (TEE) so that the voter cannot read her credential, even if she wanted to. Then of course, the coercer could take the device on which the credential is stored with him when he leaves the voter's place. However, this strategy would leave some physical evidence which poses a significant risk from a potential coercer's point of view, particularly when facing severe penal consequences. Furthermore, collecting secret credentials physically requires much time or many resources. We therefore conjecture that the risk of being caught collecting sufficiently many voting devices to significantly change the election result would not be worth the effect.

We assume that a possible coercer is not able to corrupt the posting trustee PT because he could otherwise easily see whether or not a voter updated her coerced vote. Furthermore, we make the minimal assumption that all parties who we trust for vote privacy (namely, the mixing authorities) do not collude with a possible coercer; otherwise, the coercer could easily link a coerced voter's submitted ballot with her decrypted vote in the final result.

8.6.5 Verifiability

In this section, we formally evaluate the DeVoS protocol's verifiability. To this end we use the general KTV computational model and adopt the verifiability definition with the goal $\gamma(\varphi)$ proposed in [85] presented in Section 6.3.1.

Our primary reason for choosing this model for the verification analysis is that it is suitable for our protocol. There are no explicit assumptions regarding the result function and re-voting policy.

In our model judge, $\mathfrak{J}$, an honest agent, is in charge of the verification procedure, and he executes the honest program $\hat{\pi}_{\mathfrak{J}}$ whenever triggered by the scheduler $\mathfrak{s}$. At the end of the verification procedure, $\mathfrak{J}$ outputs accept or reject through his channel.

In a nutshell, in a DeVoS protocol run, judge $\mathfrak{J}$ takes as input solely public information (e.g., the Zero-Knowledge proofs in DeVoS published on the bulletin board) and then performs certain checks. If all checks succeed, the judge accepts the protocol run, and rejects it otherwise. Precisely judge $\mathfrak{J}$ conducts the program $\hat{\pi}_{\mathfrak{J}}$ as defined in Figure 8.2.

Verifiability Assumption. We prove the verifiability of DeVoS under the following assumptions:

A.1 The PKE scheme $\mathcal{E}$ with re-randomization is correct (for verifiability, IND-CPA-security is not needed).

A.2 The function $f\colon \{0,1\}^* \to \{0,1\}^*$ is one-way.

A.3 $\pi_{\mathsf{Kgen}}, \pi_{\mathsf{Shuffle}}, \pi_{\mathsf{Dec}}, \pi^{\perp}_{\mathsf{Dec}}$ are NIP systems (for verifiability, ZK is not needed) and π_{Enc} is a NIZKP.

A.4 The scheduler $\mathfrak{s}$, the bulletin board $\mathfrak{BB}$, and judge $\mathfrak{J}$ are honest:

A.5 $\pi_{\mathsf{Kgen}}, \pi_{\mathsf{Shuffle}}, \pi_{\mathsf{Dec}}$ are NIZK systems and π_{Enc} is a NIZK Proof of knowledge.

A.6 The scheduler $\mathfrak{s}$, the bulletin board $\mathfrak{BB}$, and judge $\mathfrak{J}$ are honest:

$$\varphi = \mathsf{hon}(\mathfrak{s}) \wedge \mathsf{hon}(\mathfrak{BB}) \wedge \mathsf{hon}(\mathfrak{J}).$$

Additional Note. Note that, for verifiability to hold, the posting trustee PT, the mix server $\mathfrak{mix}.\mathfrak{s}$, the decryption trustee $\mathfrak{DT}$, as well as an arbitrary number of voters may be controlled by the adversary. In particular, we do not need to introduce any additional trust assumption compared to basic secure e-voting protocols (without coercion-resistance), such as Belenios [83].

The verification procedure $\mathfrak{J}$ of DeVoS essentially involves checking the NIZKPs published on the bulletin board $\mathfrak{BB}$: if one of these checks fails, the protocol run and hence the result are rejected. Now, the following theorem states that the probability that in a run of DeVoS an honest voter's vote has been dropped or manipulated if φ holds true (i.e., $\gamma(\varphi)$ is broken) but the protocol run is nevertheless accepted by $\mathfrak{J}$ is negligible.

Theorem 8.6.2 (Verifiability)**.** *Under the assumptions (**A.1-6**) stated above, the goal $\gamma(\varphi)$ is verifiable in the protocol $\mathsf{P}_{\mathsf{DeVoS}}(n_{\mathsf{v}}, \mathsf{cList}, \mu)$ by the judge $\mathfrak{J}$.*

FIGURE 8.2: Verification Procedure

Inputs: This procedure takes as input all public data of the election, namely the public bulletin board, $\mathfrak{BB}$,

Output: accept/reject

---

1. Run the verification algorithm for the relation $\mathsf{R}_{\mathsf{KeyGen}}$(see 7.3) with inputs the public key of the election and π_{Kgen}:

$$\begin{cases} \textbf{If } \left[\mathsf{Verify}(\mathsf{PK}_{\mathsf{election}}, \mathsf{R}_{\mathsf{KeyGen}}, \pi_{\mathsf{Kgen}}) \to 0\right] \textbf{ Then } (\mathfrak{J} \mapsto \mathsf{reject}), \\ \textbf{If } \left[\mathsf{Verify}(\mathsf{PK}_{\mathsf{election}}, \mathsf{R}_{\mathsf{KeyGen}}, \pi_{\mathsf{Kgen}}) \to 1\right] \textbf{ Then } \text{ Go to the next step,} \end{cases}$$

2. Check the validity of the signature for each ballot. If one of the following two events occurs during the tally phase, the judge will reject the election:
   - a ballot with a valid signature is withdrawn,
   - a ballot with an invalid signature is counted as a legitimate ballot.
3. Run the verification algorithm for the relation $\mathsf{R}_{\mathsf{ballot}}$(see 8.4.3) to check the validity of the proofs, π_{ballot}. As in the previous step, the judge rejects the election in case
   - there is a ballot with a valid proof π_{ballot} is withdrawn,
   - there is a ballot with invalid proof counted as a legitimate ballot.
4. Run the verification algorithm for proof π_{Shuffle} with respect to relation $\mathsf{R}_{\mathsf{shuffle}}$ 8.2, reject if the algorithm's output is 0. Otherwise go to the next step.
5. Run the verification algorithm for $\mathsf{R}_{\mathsf{dec}}$ to verify the decryption procedure and output reject in case any decryption proof fails.
6. If none of these situations occurs, judge $\mathfrak{J}$ outputs accept on a distinct tape.

According to [182], a goal γ is *verifiable* by the judge, $\mathfrak{J}$, in a protocol's run if and only if $\mathfrak{J}$ accepts a run r of DeVoS in which the goal γ is violated (i.e., $r \notin \gamma$) with at most negligible probability (in the security parameter). To formally capture this notion by

$$\Pr[(\hat{\pi}_{\mathsf{P}} \| \pi_{\mathcal{A}})^{(\ell)} \mapsto \neg\gamma, (\mathfrak{J} : \mathsf{accept})]$$

we denote the probability that a run of the protocol along with an adversary $\pi_{\mathcal{A}}$ (and a security parameter ℓ) produces a run that is not in γ but in which $\mathfrak{J}$ (nevertheless) returns accept.

This probability should be negligible in terms of the security parameter. Hence intuitively, to prove the verifiability, we need to demonstrate that the probability that in a run of DeVoS , more than k votes of honest voters have been manipulated, but judge $\mathfrak{J}$, nevertheless, accepts the run is bounded:

Proof. Assume that assumptions [**A.1-4**], as specified in Sec. 6.3.1, hold true. Then to prove Theorem 8.6.2, we need to show the following implication.

If the judge $\mathfrak{J}$ outputs accept in a given protocol run of DeVoS (in which[**A.1-4**]) are satisfied), then there exist (valid) dishonest choices $(\mathsf{vote}_{\mathtt{id}})_{i \in I_d}$ such that the election result equals $(\mathsf{c}_{\sigma(i)})_{i \in I_h \cup I_d}$, where $(\mathsf{vote}_{\mathtt{id}})_{i \in I_h}$ are the honest voters' choices and σ is some permutation.

This means that, due to the specification of $\mathfrak{J}$ Figure 8.2, each NIZKP published on $\mathfrak{BB}$ is valid.

Assume that we are in a run r of DeVoS in which the $\mathfrak{J}$ outputs accept in procedure 8.2. Then, due to the specification of $\mathfrak{J}$, each NIZKP published on $\mathfrak{BB}$ is valid.

Let $\mathsf{v}_{\mathtt{id}}$ be an arbitrary *honest* voter who chose $\mathsf{vote}_{\mathtt{id}}$ and thus submitted ballot $(\mathsf{ct}', \pi_{\mathsf{Enc}})$ where $\mathsf{ct}' \in \mathsf{Enc}(\mathsf{pk}, \mathsf{vote}_{\mathtt{id}})$. Let ct'' be ciphertext appended by the posting trustee PT to $\mathsf{v}_{\mathtt{id}}$'s vector list_i behind ct' (if any). Since each NIZKP is valid, due to the soundness of π_{Enc} (**A.3**), it follows that

(1) $\mathsf{ct}'' \in \mathsf{Enc}(\mathsf{pk}, \mathsf{c}^*)$ for some c^*, or

(2) $\mathsf{ct}'' \in \mathsf{ReEnc}(\mathsf{pk}, \mathsf{ct}')$

holds true. In case (1), due to the knowledge soundness property of π_{Enc} and the fact that PT posted a valid proof for ct'', it follows that PT knows a witness $(\mathsf{ssk}_{\mathtt{id}}, \mathsf{c}, r)$ for the relation

$$(\mathsf{vk}_i = f(\mathsf{ssk}_{\mathtt{id}})) \wedge (\mathsf{ct}'' = \mathsf{Enc}(\mathsf{pk}, \mathsf{c}^*; r)) \wedge (\mathsf{c}^* \in \mathsf{cList}).$$

Since $\mathsf{v}_{\mathtt{id}}$ is honest, there exist two possibilities for PT to learn $\mathsf{ssk}_{\mathtt{id}}$:

(1) extracting $\mathsf{ssk}_{\mathtt{id}}$ from $\mathsf{vk}_i = f(\mathsf{ssk}_{\mathtt{id}})$, or

(2) extracting $\mathsf{ssk}_{\mathtt{id}}$ from $\mathsf{v}_{\mathtt{id}}$'s NIZKP π_{Enc} for ct'.

Due to the one-way property of f (**A.2**) as well as the ZK property of π_{Enc} (assumption (V3)), we can deduce that case (1) can occur in at most a negligible set of protocol runs of DeVoS. Therefore, with overwhelming probability in the security parameter ℓ, we find that case (2) occurs. By the re-randomization property of Π^{pke} (assumption (V1)), it therefore follows via induction that the last ciphertext ct_i in list_i, which is $\mathsf{v}_{\mathtt{id}}$'s input to the subsequent tallying phase, encrypts $\mathsf{v}_{\mathtt{id}}$'s choice, i.e., $\mathsf{ct}_i \in \mathsf{Enc}(\mathsf{pk}, \mathsf{vote}_{\mathtt{id}})$.

Let $\mathsf{v}_{\mathtt{id}}$ be an arbitrary *dishonest* voter and let ct_i be the last ciphertext in list_i. Due to the soundness of π_{Enc} (assumption (V3)), there exist two possible cases:

(1) ct_i is a re-encryption of the previous ciphertext, or

(2) ct_i is a fresh encryption of some $\mathsf{vote}_{\mathtt{id}} \in \mathsf{cList}$.

In case (1), we can again distinguish between the same cases for the previous ciphertext, and so on. Due to the re-encryption property of Π^{pke} (assumption **A.1**), it therefore follows that $\mathsf{ct}_i \in \mathsf{Enc}(\mathsf{pk}, \mathsf{vote}_{\mathtt{id}})$ for some $\mathsf{vote}_{\mathtt{id}} \in \mathsf{cList}$.

From what we have shown above, we can deduce that (with overwhelming probability) the input to the tallying phase consists of ciphertexts $(\mathsf{ct}_i)_{i \in I_h \cup I_d}$, where for each $i \in I_h$ the respective ciphertext ct_i encrypts $\mathsf{v}_{\mathtt{id}}$'s intended choice $\mathsf{vote}_{\mathtt{id}}$, and where for each $i \in I_d$ the respective ciphertext ct_i encrypts some $\mathsf{vote}_{\mathtt{id}}$.

In the next step in the tally phase, mix-nets shuffle the list $\mathbf{L}_1$ contains valid ballots on the bulletin board and outputs the second list $\mathbf{L}_2$ along with the proof of shuffle, π_{Shuffle}. Since the judge accepted the protocol run, the mix server's NIZKP π_{Shuffle}, (step 4 in Figure 8.2) is valid. Due to the soundness of π_{Shuffle} (**A.3**), it follows that (with overwhelming probability) there exists some permutation σ such that for each $i \in [n_{\mathsf{v}}]$, we have that $\mathsf{ct}'_i \in \mathsf{ReEnc}(\mathsf{pk}, \mathsf{ct}_{\sigma(i)})$. Due to the re-encryption property of Π^{pke} (**A.1**) and the bijective property of σ, we can deduce that for each honest voter $\mathsf{v}_{\mathtt{id}}$, there exists ct'_j in the output $\mathbf{L}_2$ of mix server $\mathfrak{mix}.\mathfrak{s}$ such that $\mathsf{ct}_j \in \mathsf{Enc}(\mathsf{pk}, \mathsf{vote}_{\mathtt{id}})$, where $\mathsf{vote}_{\mathtt{id}}$ is $\mathsf{v}_{\mathtt{id}}$'s original choice; analogously for the dishonest voters.

Since the judge accepted the protocol run, the decryption trustee's NIZKPs π_{Dec} and π^f_{Dec} (if any) are valid (Step 5 in 8.2). Due to the soundness of π_{Dec} and π^f_{Dec} (**A.3**), it follows that (with overwhelming probability) there exists sk such that $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Kgen}$ and $[\mathsf{m}_j]_{j \in [n_{\mathsf{v}}]} = \mathsf{Dec}(\mathsf{sk}, \mathsf{L}_2)$ and $m_j \in \mathsf{cList}$ for each index j in the list L_2. Because the encryption scheme is correct (**A.1**), we can eventually conclude that there exist valid choices $\mathsf{vote}_{\mathtt{id}}$ ($i \in I'_d \subseteq I_d$) and a permutation σ such that $\mathsf{res} = (\mathsf{c}_{\sigma(i)})_{i \in I_h \cup I'_d}$,

□

Chapter 9

Risk-Limiting Tallies

Consider an election in where some candidates receive few or no votes, as frequently occurs in real elections. As an example, consider a first-past-the-post *Student Rep Election* with four candidates $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$, $\mathcal{D}$, with popularity ratings of $49.9\%, 49.9\%, 0.1\%$ and 0.1%, respectively. $\mathcal{C}$ and $\mathcal{D}$ have a negligible chance of winning and $\mathcal{A}$ and $\mathcal{B}$ have a high but equal chance of the other. Consider the following scenario: your boss, or someone with the power to penalize you, would ask you to vote for $\mathcal{D}$, the candidate that no one has likely voted for, but your preferred candidate is $\mathcal{A}$. What would you do? You have two choices:

i. The first choice would be to give up on your preference and vote $\mathcal{D}$. We may believe that voting for the low-support candidate D is unlikely to result in $\mathcal{D}$'s win. However the situation is more difficult than it appears at first sight. Because, by taking away votes from $\mathcal{A}$, the primary opponent of $\mathcal{B}$ (the coercer's preferred candida), the coercer is performing an abstention attack. Simply put, the coerced voter is forced to support $\mathcal{B}$ by voting for $\mathcal{D}$.

ii. You can vote for $\mathcal{A}$ and face the chance that if no one votes for $\mathcal{D}$, you will be punished, which has a high possibility. (There is a high probability that you will be punished if no one votes for it.)

As the example shows, there are some situations in which even the most powerful and secure e-voting protocol cannot protect the voter from coercion. In this type of situation, *Risk-Limiting Tallies* (RLT) are a mechanism that safeguards' voters. The concept is simple: only a random subset of ballots is revealed during the tally phase, while the remaining remain masked. Then coerced voters can always claim that they followed all of the instructions, but their ballots were masked. In other words, they have plausible deniability. This chapter, presents our research on a new method in risk-limiting tally for voting systems involving complex ballot forms.

Contents

Risk-Limiting Tallies (RLT), which reveal only a random sample of ballots, have previously been proposed as a method of mitigating certain coercion threats [164]. While masking some ballots provides plausible deniability for coerced voters, risk-limiting techniques ensure that the required level of confidence in the election result is achieved. RLV extended this approach by masking a random subset of receipts or trackers.

This chapter shows how the RLT approach can be generalised and made more flexible and practical by masking at a finer granularity: at the level of ballot components.

We focus on elections with complex ballots, in which RLT may be vulnerable to pattern-based vote-buying. We propose several measures of verifiability and resistance to coercion and investigate the performance of several sampling/masking strategies against these measures. Additionally, we define new quantitative measures for the level of coercion resistance in the lack of plausible deniability and vote-buying resistance in the absence of "free lunch" vote sellers. Additionally, we will discuss the privacy breached by such masked ballots, including their coercion-resistance and receipt-free nature. These results and the various ballot masking strategies, are relevant for all elections that publish ballots for auditing, verification, or transparency purposes.

Outline. The outline of this chapter is as follows; After an introduction in Section 9.1 Section 9.2 explain the idea of partially masking ballots. Section 9.3 describes how RLT can be used in masked RLT and RLV. Section 9.4 present our result for in distinguishing distance between randomly masked ballots. Section 9.5 considers quantitative game-based notions of privacy, coercion-resistance, and receipt-freeness. Section 9.6 concludes. We would like to emphasize that the research presented in this chapter was previously published in (E-Vote-ID-2021)[1] [227].

9.1 Introduction

Some voting systems, including many E2E verifiable systems and some conventional elections, such as some Australian elections, publish ballots (plaintext). If these ballots are suitably anonymised, by for example verifiable mixes published on a bulletin board, then this is typically quite safe. But in some contexts, revealing such information may be problematic: certain corner cases, such as unanimous votes or absence of any votes for a candidate and coercion threats, such as signature attacks.

[1]The International Conference for Electronic Voting

In [164] the idea of Risk-Limiting Tallies (RLT) and Risk-Limiting Verification (RLV) was proposed to mitigate such threats. The idea is to shroud a proportion of the (anonymised) votes so voters can plausibly claim to have complied with the coercer, even though no votes appear for the candidate demanded by the coercer or no ballot with the pattern demanded by the coercer shows up in the tally. The proportion left shrouded can be adjusted using risk-limiting techniques to ensure that the confidence in the announced outcome achieves the required threshold, e.g. 99%. The idea extends to the verification aspects: shrouding some proportion of receipts or trackers. This proves particularly effective in for example the Selene scheme to counter the "sting in the tail": the coercer claiming that the voter's fake tracker is his own.

In this paper, we note that despite the pleasing features of the constructions of [164], there are still some drawbacks, in particular if the ballots are rather complex. Moreover, while RLT may disincentivize *coercion*, there may still be an incentive for *vote-buying*: the voter might still cast the required pattern vote in the hope that it will be revealed.

Further, it has been suggested that RLT is arguably undemocratic in that some voters' ballots do not contribute to the final tally.

The second objection can be countered by arguing that every vote has an equal probability of being included in the count. The outcome will be a correct reflection of all votes cast with whatever confidence level is required. Nonetheless, it is an aspect that some people find troubling and thus is worth addressing. A pleasing side effect of our construction is that all ballots are treated equally.

These observations suggest exploring different ways to apply RLT and RLV when ballots are complex: rather than shrouding entire ballots at random, we shroud, at random, some preferences on each ballot. In effect we are filtering the tally horizontally rather than vertically. This hits both of the issues above: the chance any given pattern remains identifiable after the filtering is reduced, and every ballot contributes to the outcome, albeit not necessarily to every contest. In the *full tally* construction below, every ballot contributes fully to the announced outcome, but we shroud the link between the tracker and some components of the ballots. For tracker-based schemes, the voters can verify some but not all of their selections. This paper seeks to quantify these effects and explore trade-offs among them.

Our techniques allow us to state and prove bounds on the number of voters an adversary can attack using pattern-based or "signature" attacks. firs, note that assigning the same or similar, complex ballot pattern to many voters is counterproductive for the adversary. If even a few voters comply, the rest can point to the signature ballots that already appear and claim compliance. Thus, an adversary who wants to influence many voters with a signature attack must be able to produce many distinguishable ballot patterns. This observation motivates us to prove lower and upper bounds on the number of distinguishable patterns an adversary can construct. We prove these bounds using a connection to a well-studied problem in the theory of error-correcting codes.

This ballot-masking method and its privacy implications are interesting not only for RLT and RLV but for all schemes where all or some ballots are published for auditing, verification, or transparency. For example, Colorado is currently redacting cast-vote records (CVRs) by removing entire CVRs, e.g., for rare ballot styles; partial masking has been considered an alternative. We note, however, that masking parts of the ballot might make it hard to detect ill-formed, e.g., over-votes etc.

We also note that this idea has similarities to the SOBA constructions for Risk-Limiting-Audits (RLAs), [38], which also publishes each audited ballot "disassembled" into different contests, In contrast, the auditors will see the intact ballot. The VAULT approach [37] also uses homomorphic encryption of the cast-vote records to achieve the SOBA goals more easily. (VAULT was used for the first time in a risk-limiting audit in Inyo County, California, in 2020.) The purpose and underlying cryptographic constructions are quite different, but our analysis also applies to these cases applies.

We can separate ballots into atomic parts for some tally algorithms and reveal them independently after anonymising them, effectively counting signature attacks. However, that reduces public transparency and may reduce public confidence in the election result.

For Selene, where voters verify their votes via trackers, this separation provides a method to verify without revealing individual ballots: we simply assign a distinct tracker to each ballot element. Voters can then verify some or all components of their ballot using those trackers. A coerced voter could use the Selene tracker-faking mechanism to assemble a ballot that matches the coercer's instructions. Technically this is straightforward but from usability, standpoint seems problematic. Moreover, even if the voter was prepared to concoct such a fake ballot, the necessary ingredients might not be available, so coercion threats remain. The probability that one of the atomic trackers is the same as the coercer's increases. Thus it makes sense to look for alternatives.

Below, we present the main ideas and analyse differences in privacy, coercion-resistance, and receipt-freeness for the different methods.

9.2 Masking Complex Ballots

Many elections use simple plurality voting: the voter selects at most one candidate from a set, in the simplest case, a referendum, choosing between "yes" and "no." The $k \geq 1$ candidates who get the most votes win, or the candidate(s) win at least a threshold fraction of the valid votes. The next level of complexity is single-winner plurality, aka "first past the post." More complex social choice functions and correspondingly more complex ballots are common. Perhaps the next level in complexity is *approval voting*. The voter can cast votes for several candidates for a single office, and multi-winner plurality, in which a voter can vote for up to k candidates for k offices. In some cases, voters may have a quota of votes and are allowed to cast more than one vote for a given candidate, up to some limit. Some methods allow voters to give a preference ranking to the candidates, which can then be translated into "points", which are then totaled, and these rankings are later converted to points, k points for the first preference, $k-1$ for the second and so on. In all these cases, the votes for each candidate are added up, and the one with the most votes wins.

All the above are what we call *separable* voting methods; that, is, how the components of the ballots are grouped into the individual ballots does not affect the outcome. Consequently we can separate the components and forget any information about to how they were originally grouped, at least if we have some method assuring that the ballots were well-formed. Some voting methods are not separable in this sense; prime examples are Single Transferable Votes (STV) and Instant Runoff.

Common to all of these social choice functions, if the ballots are published, the number of ways a ballot can be filled grows exponentially, meaning that they are vulnerable to signature attacks (also known as "Italian" attacks),*i.e.,* a coercer chooses a particular, unlikely, pattern, instructs the victim to mark a ballot with that pattern and checks whether a ballot with that pattern appears in the tally.

Let us assume that the ballots are of the following form:

$$(v_1, v_2, \ldots, v_k) : v_i \in \mathsf{cList}$$

with k the number of candidates and v_i taking values from a specified set of the candidate list. cList might, for example, just be $\{0,1\}$ or a set of integers plus a blank: $\{1, \ldots., s\}$ $\bigcup$ $\{\text{blank}\}$.

In many types of elections, these ballot-level selections, or subsets thereof, will reappear as part of the tally procedure (e.g. in electronic mixnet tallies), as part of an audit trail or for transparency (electronic scans of paper ballots), in Risk-Limiting Audits using samples of votes, or verification procedures (e.g. in tracker-based schemes such as Selene). However, the mapping between the published votes and the voter is normally anonymised to preserve privacy.

As mentioned above, revealing these ballots may endanger the receipt-freeness of the election. With *Masked Tallies*, introduced here, only parts of each ballot are revealed:

$$\left(\mathsf{mask}_{i1}(v_1^{(i)}), \mathsf{mask}_{i2}(v_2^{(i)}), \ldots, \mathsf{mask}_{ik}(v_k^{(i)})\right) \quad \text{for } i = 1, \ldots, n_{\mathsf{v}}.$$

The functions mask_{ij} are either the identity, displaying the component of the vote, or a constant, .e.g., $* \notin \mathsf{cList}$, masking the component and n is the number of ballots cast.

Risk-Limiting Tallies [164] involved unmasking as many randomly selected ballots as needed to determine the election result with a chosen risk limit. The remaining ballots were kept completely masked. Here we suggest a generalization, allowing partial masking of the ballots, and we will discuss the impact on risk limits, privacy, coercion-resistance, and resistance to vote-buying.

9.3 Partially Masked RLTs and RLVs

Before extending these to general masks, we reprise risk-limiting tallies and verification, RLT and RLV [164]. But, first, we recapitulate the idea of tracker-based verification in terms of Selene.

9.3.1 Selene

Selene($\Sigma\varepsilon\lambda\acute{\eta}\nu\eta$)[2], introduced by Peter Ryan*et al.* [225], is an e-voting protocol that puts voters' choices in the clear on the $\mathfrak{BB}$ while providing a unique private tracking number that allows voters to verify their vote. Furthermore, establishing a level of coercion-resistance ensures that the voters only learn their tracking numbers after the votes have been posted. The resulting scheme provides receipt-freeness, a good level of coercion-resistance while also providing a more immediately understandable form of verifiability. We briefly mention the protocol here, and for more details on the subject we refer to [225].

Cryptographic primitives Selene relies on the following primitives: signature scheme 2.7.4, ElGamal encryption scheme 2.7.2.1 and verifiable re-encryption mix protocols, the Fiat-Shamir heuristic in RO model 3.8.4 and Pederson Commitment 2.7.1.

Protocol participants. Selene is run among the following participants: voting authority $\mathfrak{Auth}$, bulletin board $\mathfrak{BB}$, mix server $\mathfrak{mix.s}$, tellers $\mathfrak{T}$, and voters $\mathsf{v}_1, \ldots, \mathsf{v}_{n_\mathsf{v}}$.

[2]In Greek mythology, $\Sigma\varepsilon\lambda\acute{\eta}\nu\eta$ means moon

Protocol description. Selene has the following phases:

Set up. The election authority runs the key generation algorithm of ElGamal encryption scheme, to generate the pair of keys for the election and publish them on the bulletin board along with the election parameter:

$$\mathsf{key}_{\mathsf{election}} = (\mathsf{PK}, \mathsf{SK}); \mathsf{PK} = \left(\mathbb{G}, g, h, \mathsf{pp}_{\mathsf{election}}\right), \mathsf{SK} = (x)$$

and then in the next step for all voters, $\mathsf{v}_1, \ldots, \mathsf{v}_{n_\mathsf{v}}$ a pair of key is assigned:

$$\mathsf{v}_i : \left(h_i, \mathsf{sk}_i; h_i = g^{\mathsf{sk}_i}\right)$$

Generating the tracking number.

1. The election trustee publicly creates a set of tracker numbers $\mathsf{tr.n}_j$ for $j = 1\ldots, n_\mathsf{v}$, computes $g^{\mathsf{tr.n}_j}$ and generate a list, $\mathsf{L}^* = \{\{g^{n_j}\}_{\mathsf{pk}}\}_j$ where $\left\{g^{n_j}\right\}_{\mathsf{pk}} = \mathsf{ElGamal.Enc}_{\mathsf{pk}}(g^{n_j})$.

$$\mathsf{L}^* = \left\{n_j, g^{n_i}, \{g^{n_j}\}_{\mathsf{pk}}\right\} \tag{9.1}$$

2. Then the list L is put through a sequence of verifiable re-encryption and permutation mixnet to obtain the new list $\mathsf{L} = \{g^{n_{\sigma i}}\}_{\mathsf{pk}}$.

3. Each component from the list L is assigned to each voter public key h_i

$$\left(\mathsf{v}_i : (h_i, \mathsf{sk}_i), \{g^{\mathsf{tr.n}_{\sigma i}}\}\right) \tag{9.2}$$

Generating the commitment:

1. The tellers $\mathfrak{T}_1, \ldots, \mathfrak{T}_t$, distributively, generate the trapdoor commitments:

2. For $j = 1, \ldots, t$ the teller $\mathfrak{T}_j$ choose n_v random numbers r_{ij} for each voter v_j and encrypt $h_i^{\mathsf{r}_i}$. At the end of this step, for each voter v_i, t, ciphertexts are generated:

$$(\mathsf{v}_i, h_i) : \left[\{h_i^{\mathsf{r}_{i1}}\}, \{h_i^{\mathsf{r}_{i2}}\}, \ldots, \{h_i^{\mathsf{r}_{it}}\}\right]$$

by exploiting the multiplicative homomorphic property of ElGamal cryptosystem we obtain:

$$\{h_i^{\mathsf{r}_i}\} = \prod_{j=1}^{t} \{h_i^{\mathsf{r}_{ij}}\} = \{h_i^{\mathsf{r}_i = \sum_{j=1}^{t} \mathsf{r}_{ij}}\}$$
$$\{h_i^{\mathsf{r}_i}\} \cdot \{g^{n_{\sigma i}}\} = \{h_i^{\mathsf{r}_i} . g^{\mathsf{tr.n}_{\sigma i}}\}$$

Note that the tellers must keep their $g^{\mathsf{r}_{ij}}$.

3. Next the teller by decrypting the term $\{h_i^{\mathsf{r}_i} . g^{\mathsf{tr.n}_{\sigma i}}\}$, retrieves $\mathsf{Com}_i = \mathsf{r}_i \cdot \mathsf{tr.n}_{\sigma i}$. We consider Com_i as a commitment of voter's tracking number which in fact is Pedersen commitment to value n_i.

4. Now they publish the following terms for each voter:

$$\mathsf{v}_i : (h_i, \{g^{\mathsf{tr.n}_i}\}, \mathsf{Com}_i \cdot g^{\mathsf{tr.n}_{\sigma i}}, .)$$

All these steps have been done in a verifiable way, meaning that each step is provided with zero-knowledge proofs, (for simplicity, we will not mention details.) The last entry is left blank for the vote.

Voting. In this step, every voter v_i encrypt their choice, $\{\mathsf{vote}_i\}$, sign it (to avoid ballot stuffing[12]) and casts their vote in the form:

$$\big(\mathsf{sign}(\{\mathsf{vote}_i\}_{\mathsf{pk}_T}), \pi_i\big)$$

where π_i is a non-interactive proof of knowledge of the plaintext to ensure ballot independence [16, 35,13] and prevent an attacker from copying and re-encrypting a previously cast a vote as their own. The voter sends their ballot to the bulletin board. The bulletin board checks the signature and the proof before publishing the ballot :

$$\big(\mathsf{v}_i, h_i, \{g^{\mathsf{tr.n}_i}\}, (h_i^{\mathsf{r}_i} \cdot g^{\mathsf{tr.n}_{\sigma i}}), (\{\mathsf{vote}_i\}_{\mathsf{pk}_T}, \pi_i)\big)$$

The second and fourth components are taken out and put through verifiable re-encryption mix-nets and threshold decrypted for each voter. We then obtain the list of pairs:

$$(\mathsf{vote}_i, n_{\sigma i})$$

Revealing the trackers. After the trackers and votes have been published on $\mathfrak{BB}$, each teller $\mathfrak{T}_j$ sends $g^{\mathsf{r}_{ij}}$ to the voter via a private channel for suitable amount of time. Each voter combine the randomness to obtain,

$$\prod_{j=1}^{t} g^{\mathsf{r}i,j} = g^{\mathsf{r}_i}$$

which is the α-term of the ElGamal encryption under the voter's public key. Now the voter has both components:

$$\begin{aligned}
\mathsf{Enc}_{\mathsf{pk}_i}(g^{n_i}; \mathsf{r}_i) = (\alpha_i, \beta_i)& \\
\alpha_i &= \prod_{j=1}^{t} g^{\mathsf{r}i,j} = g^{\mathsf{r}_i} \\
\beta_i &= \mathsf{Com}_i = h_i^{\mathsf{r}_i} \cdot g^{n_{\sigma j}} \\
&\Rightarrow \mathsf{Dec}_{\mathsf{sk}_i}(\alpha_i, \beta_i) \mapsto g^{\sigma i}
\end{aligned}$$

And by having the value $g^{n_{\sigma i}}$ voter, she can find her number $n_{\sigma i}$ from the list 9.1 and verify her vote.

We reprise risk-limiting tallies and verification, RLT and RLV [164], before extending these to general masks. Now we recapitulate the idea of tracker-based verification in terms of Selene.

According to the above description, assume that votes are encrypted componentwise; at, the end of the mixing we will have encrypted votes and trackers that have been mixed verifiably in parallel on the bulletin board:

$$(\{\mathsf{tr.n}_i\}_{\mathsf{pk}}, (\{v_1^{(i)}\}_{\mathsf{pk}}, \{v_2^{(i)}\}_{\mathsf{pk}}, \ldots\ldots \{v_k^{(i)}\}_{\mathsf{pk}}))$$

These ballots can now be verifiably decrypted to reveal the vote/tracker pairs that the voters can check, and anyone can compute the tally directly on the plaintext votes.

9.3.2 RLTs and Verification with Partially Masked Ballots

In the original approach to RLT (where ballots are without trackers) and RLV (with trackers for individual verification), see [164], the idea was only to decrypt a random subset of the ballots. The number decrypted is controlled by a risk-limit that bounds the probability that the announced election result will be wrong.

In the new masked RLV and RLT approach, we reveal randomly selected ballots' components (and RLV trackers). If there is more than one contest on the ballot, the contests can be treated independently. How much we reveal will again be governed by a specified risk limit, as in [164]. A natural choice is to first decrypt m of each ballot's k entries at random and increase m if necessary to meet the risk limit. This is the simplest and will be used in the analysis below. In practice, it may make sense to change the rate of openings per candidate dynamically, e.g. if a candidate is popular, we might be able to decrease the rate of the unmasking of votes for that candidate, maintaining the risk limit while improving coercion-resistance.

Using this masked approach for RLV with tracker verification masking means that only parts of the ballot can be verified. Still, unlike to the original RLV, every voter can verify something. We will quantify how much.

9.3.2.1 Full Tally with Partial Verification (FTPV)

A social choice function is *separable* if, for tallying purposes, each vote's components can be considered separately. For example, plurality, approval, and Borda count are separable; instant-runoff voting and single transferrable vote are not. For separable social choice functions, it is possible to compute the full tally, i.e. achieve 100% confidence in the outcome while partially masking selections. For each ballot, we randomly select some components. All selected components for all ballots are gathered in another part of the BB and subjected to a full, component-wise shuffling before decryption. Then, their positions in the original ballots are replaced by $*$. Thus, the way these selected components appeared in the original ballots is lost.

The FTPV approach above might still hit corner cases, for instance, if no vote was cast for a particular candidate. This suggests using a hybrid approach in which we use the approach above but reveal a random subset of the components separated from the ballots. Thus, we reveal enough of each ballot linked to the tracker to make verification meaningful while mitigating coercion threats. A larger portion of the ballots is revealed without a link to the trackers to attain the required risk limit tally

9.3.2.2 Masked Audited Tallies

An interesting option for efficient computation of complex social choice functions is to use the unmasking to challenge the correctness of the tally. At the same time, partial homomorphic encryption is efficient for calculating the sum of votes, while more complex functions are less efficient. However, randomly revealed (previously mixed) plaintexts could be used to audit the correctness of the computation of the election result on encrypted data. One example is the possibility of encrypting the data with both multiplicative ElGamal and additive Paillier for efficient computations. Normally, it is hard to prove that the plaintexts are equal, but masking can be directly checked for the revealed plaintexts. This gives guarantees of the integrity of the tally against risk-adverse adversaries.

9.4 Distinguishing Distance and Applications to Signature Attacks and Individual Verifiability

In [227] we define a metric on the set of complex ballots that characterizes how well pairs of strings can be distinguished under random masking. We then observe that this metric is a monotone transformation of the Hamming distance used in coding theory in some cases. We also precisely characterize the cases when this occurs. Next, we use the connection to coding theory to answer the following question: how many simultaneous signature attacks can a coercer and/or vote-buyer launch result in Theorem 9.4.1. Finally, we give another application of the distinguishing distance: we use it to quantify the effect of a masking strategy on individual verifiability (See 9.3).

Here we briefly describe our result related to *distinguishing distance*, and for more details on the concept and proof of the theorem, we refer to [227].[3] Therefore, we prove the following theorem to establish our bounds on the number of simultaneous signature attacks under a pairwise distinguishability constraint.

Theorem 9.4.1. *For every finite set* cList*, for every* $k \in \mathbb{N}$*, for every probability distribution* p_S *on subsets of* $\{1,\ldots,k\}$ *satisfying*

$$\exists (r(0),\ldots,r(k))\ \forall s,\ p_S(s) = \frac{r(|s|)}{\binom{k}{|s|}},$$

for every $q \in [0, 1 - p_S(\emptyset)]$*, let* $r_{max}(\mathcal{V},k,p_S,q)$ *denote the size of the largest collection* $\{x_1,\ldots x_r\}$ *with the property* $\forall i \neq j, d_{p_S}(x_i,x_j) \geq q$*. Then*

$$\frac{|\mathcal{V}|^k}{\sum_{j=0}^{g_{p_S}(q)-1}\binom{k}{j}(|\mathcal{V}|-1)^j} \leq r_{max}(\mathcal{V},k,p_S,q) \leq \frac{|\mathcal{V}|^k}{\sum_{j=0}^{\lfloor (g_{p_S}(q)-1)/2\rfloor}\binom{k}{j}(|\mathcal{V}|-1)^j}$$

In the above theorem, $\mathcal{V}$ = cList refers to the possible votes $\mathcal{V}$ = cList, and we consider the complex ballots with k components taken from the set $\mathcal{V}$; thus, the set of possible ballots is $\mathcal{V}^k$. We ignore here any constraints on what constitute valid ballots. For $x \in \mathcal{V}^k$ and $S \subset \{1,\ldots,k\}$, we denote by x_S the substring of x on the positions in S.

Quantifying the Effect of Masking on Individual Verifiability. We also quantify the effect of a particular masking strategy, specified by the probability distribution p_S, on individual verifiability. We propose the following quantity:[3]

$$IV(p_S) = \inf_{x \neq y \in \mathcal{V}^k} d_{p_S}(x,y) \tag{9.3}$$

This quantity takes values between 0 and 1, where $IV(p_S) = 1$ means that the masking strategy p_S leaves the individual verifiability of the underlying voting protocol invariant, while $IV(p_S) = 0$ means that the masking strategy p_S destroys any individual verifiability that was present in the underlying voting protocol.

One attractive feature of this setup is that an individual voter does not need to know the distribution p_S.

[3]Due to the fact that the author of this thesis was not the primary contributor to this part, we only refer to the subject and will not give the details in this manuscript.

For distributions p_S that satisfy $\exists(r(0),\dots,r(k))\forall s, p_S(s) = \frac{r(|s|)}{\binom{k}{|s|}}$, we propose a simple formula for $IV(p_S)$:

$$IV(p_S) = \sum_{j=0}^{k-1} \frac{\binom{k-1}{j} r(j+1)}{\binom{k}{j+1}} = \sum_{l=1}^{k} \frac{l}{k} r(l).$$

9.5 Quantitative Privacy-Type Properties

We now want to measure and compare privacy properties for different masked tally methods. When computing concrete values we will consider approval voting with k candidates only 0 or 1 is allowed for each candidate, without any overall constraint, $(v_1,\dots,v_k) \in \{0,1\}^k$. For the n honest voters we assume for simplicity that the probability to vote $v_i = 1$ is p_i and these probabilities are independent. As a special concrete case we consider a student election with $n = 1001$ voters (one voter is under observation), $k = 5$ candidates with probabilities $(0.6, 0.4, 0.01, 0.01, 0.01)$, i.e. two popular candidates and three unpopular.

9.5.1 Privacy

We first consider the quantitative δ-privacy definition from [183] (See 6.2.3). The main other quantitative privacy definition is [45], but it is less suited considering signature attacks. We mention that in definition 40, the value δ will depend on the chosen vote distribution, and we see that it is especially relevant to penalize signature attacks: if we assume that there is a vote choice $v^* = (v_1^*,\dots,v_k^*)$ which rarely gets selected and has a probability close to zero, then an unmasked tally which reveals all cast plaintext ballots, even in anonymised form, will have $\delta = 1$ (when the adversary simply checks if v^* appears).

Full Ballot Disclosure. When we reveal all ballots, we can consider the case where the observer tries to distinguish a voter casting the most unpopular vote vs the most popular vote, as in a signature attack. That is, in the definition we let $v_0^O = (v_1,\dots,v_k)$ with $v_i = 1$ if $p_i \le 1/2$ and $v_i = 0$ if $p_i > 1/2$, and we have $v_1^O = (1-v_1,\dots,1-v_k)$. Denote the corresponding probability p_{min}. Now a good strategy is simply to check if at least one $(v_1,\dots,v_k)$ appears in the disclosed ballots, and the algorithm then outputs "1". This means

$$\Pr[(\pi_{\mathsf{O}}||\pi_{\mathsf{V}_{obs}}(v_0^O)||\pi_v)^{(l)} \to 1] = 1,$$

but $(\pi_{\mathsf{O}}||\pi_{\mathsf{V}_{obs}}(v_1^O)||\pi_v)$ will also output "1" if another voter chooses v_0^O. This happens with probability $1-(1-p_{min})^{n_h}$. We conclude that $\delta \ge (1-p_{min})^{n_h}$. For the case of the student election we have that $v_0^O = (0,1,1,1,1)$ with $p_{min} = 0.4^2 \cdot 0.01^3 = 1.6 \cdot 10^{-7}$. Thus for $n_h = 1000$ we have $\delta \ge (1-p_{min})^{n_h} \approx 0.99984$, i.e. close to 1.

Result Only. We now consider the case where we only reveal the overall result

$$\mathsf{res} = (r_1,\dots,r_k).$$

In this case we can follow an analysis close to [183, 184] for calculating δ. For every possible result r we calculate the probability that the result happened if the observed voter cast v_0^O or v_1^O. The algorithm will then output one if the former probability is larger. We get

$$\delta = \sum_{r \in M^*_{v_0^O, v_1^O}} (A_r^{v_0^O} - A_r^{v_1^O})$$

where

$$M^*_{v_0^O, v_1^O} = \{r \in \mathcal{R} : A_r^{v_1^O} \leq A_r^{v_0^O}\},$$

and $\mathcal{R}$ is the set of all possible results of the election. at the same time, A_r^v denotes the probability that the choices of the honest voters yield the result r given that v_{obs}'s choice is v. These probabilities can explicitly be calculated since each candidate count from the honest voters, X_i, is binomially distributed, $X_i \sim BD(n_h, p_i)$. We thus have

$$A_r^v = \mathbb{P}(X_1 = r_1 - v_1) \cdots \mathbb{P}(X_k = r_k - v_k) = \prod_{i=1}^{k} \binom{n-1}{r_i - v_i} p_i^{r_i - v_i} (1-p_i)^{n - r_i + v_i - 1}.$$

RLT. In the original RLT method, we keep a certain fraction, f_{blind}, of the ballots hidden, that is $(1 - f_{\mathsf{blind}})n$ ballots are published. Let's consider the optimal algorithm from the full ballot disclosure and the corresponding δ_{full}. We see that $\delta = (1 - f_{\mathsf{blind}})\delta_{\mathsf{full}}$ since the probability that the observed voter's ballot is hidden is $(1 - f_{\mathsf{blind}})$.

Masked RLT. We now consider the case of masked RLTs where we release all ballots but with only m out of k components unmasked. A good strategy to lower bound δ is to count the number N_b of colliding ballots v which satisfy

$$\mathsf{mask}_v v = \mathsf{mask}_v v_b^O \text{ for } b = 0, 1.$$

We choose v_0^O as the most unlikely ballot, as above and take v_1^O as the opposite ballot to discriminate optimally between the two counts. The main distinguishing power comes from N_0, and we let the distinguishing algorithm output "1" if the probability of the honest voters casting $N_0 - 1$ colliding votes is higher than getting N_0 collisions. The probability for each honest voter to have a collision is

$$p_{\mathsf{col}} = \binom{k}{m}^{-1} \cdot \sum_{1 \leq i_1 < i_2 < \ldots < i_m \leq k} p_{i_1} \cdots p_{i_m}$$

and $N_0 \sim BD(n_h, p)$, where p_i is the probability of a match in the ith candidate. In [227] we analyse the above probability formula for the student election example[4]. The algorithm above will then simply give the probability at the mode of the binomial distribution with p_{col}. For $m = 3$ we find $\delta \geq 0.6$ for the student election.

9.5.2 Coercion-Resistance and No Deniability

We now analyse the coercion-resistance level, as defined in *Coercion-Resistance* [181] (See 39) each technique provides. However, since plausible deniability is an essential factor for the usability of coercion-resistance mechanisms, we consider the following definition to measure this aspect.

[4]Due to the fact that the author of this thesis was not the primary contributor to this part, we only refer to the subject and will not give the details in this manuscript.

The level of the plausibility of a voter claiming to have followed the coercer while actually following the counter-strategy, relates to the probability of false positives when the coercer tries to determine if the voter disregarded the instructions. In the following we assume that the coercer outputs 1 when blaming the voter without loss of generality. We now want to define the maximal probability of getting caught without any deniability, i.e. we consider the case where $\Pr[(\pi_c || \pi_{v_{co}} || \pi_v)^{(l)} \mapsto 1] = 0$ or negligible, i.e. the coercer only uses strategies where he never blames an honest voter.

Definition 50. *An e-voting protocol S achieves $\delta^{cr,no-d}$-coercion-resistance if for all dictated coerced strategies $\pi_{v_{co}} \in V_S$ there exists a counter-strategy $\tilde{\pi}_{v_{co}} \in V_S$ s.t. for all coercer programs $\pi_c \in C_S$:*

- $\Pr[(\pi_c || \tilde{\pi}_{v_{co}} || \pi_v)^{(l)} \mapsto \gamma]$ *is overwhelming.*
- $\Pr[(\pi_c || \tilde{\pi}_{v_{co}} || \pi_v)^{(l)} \mapsto 1]$ *is $\delta^{cr,no-d}$-bounded and* $\Pr[(\pi_c || \pi_{v_{co}} || \pi_v)^{(l)} \mapsto 1]$ *is negligible.*

The coercer's optimal strategy to obtain this $\delta^{cr,no-d}$ and the voter's strategy might differ from those in Definition 39 but $\delta^{cr,no-d} \le \delta^{cr}$.

The no deniability probability clearly separates the RLT approaches. However, the original RLT always has plausible deniability if we choose to keep some ratio of ballots shrouded, and the voter can claim her ballot was not revealed. This is e.g. important for RLV giving deniability against an attack where the coercer provides a ciphertext to cast and asks for its decrypted vote.

In the case of masked ballots, there can be a chance of getting caught undeniably. This will depend strongly on the number of revealed ballot components m, the vote distribution and the voter's goal. For the student election analysed above, the worst case when the goal of the voter is to cast $(1,0,0,0,0)$. The coercer's optimal strategy is then to demand a vote for $(0,1,1,1,1)$. The coercer will blame the voter if there is no matching masked ballot, i.e. if no honest voters produce a collision which happens with probability $(1-p_{col})^{n_h+1}$. The probability of no deniability is then $p = 8 \cdot 10^{-9}$ for $m = 2$ but jumps abruptly to $p = 0.6$ for $m = 3$.

An interesting case is when the voter has a simple goal to cast a signature part or not, and when the vote distribution has some ballots strictly zero probability. for example let us consider a three candidate 0/1 election with 1-vote probabilities $(1/2, 1/2, 0)$. The voter's goal is to cast a 1 for the first candidate. The coercer's optimal strategy is to demand a signature ballot $(0,0,1)$. The voter has two counter-strategies: 1) casting a vote $(1,0,0)$ without the signature part or 2) casting a vote $(1,0,1)$ with the signature part. For (1) there is no deniability if no other voter casts a matching ballot and the coerced voter's ballot does not match either.

For $m = 1$, this happens with $p = (2/3)^{n_h+1}$ and for $m = 2$ with $p = (11/12)^{n_h}$, both are small if we have many voters. For 2) there will always be a matching vote if the first part of the coerced voter's ballot is masked. However, if the last part is revealed, the coercer can deduce this ballot comes from the coerced voter since this candidate had a probability of 0. If the vote in the first part is revealed as well, then the voter is caught with no deniability. Thus is no deniability with probability $1/3(2/3)^{n_h}$ for $m = 1$ and $1/3 + 1/3(11/12)^{n_h}$ for $m = 2$. Thus for $m = 1$, strategy 2) is always better, but for $m = 2$ strategy 1) is better when we have more than 13 voters. In some cases the voter strategy thus depends on m, which might not be known beforehand.

Finally, it is also natural to define the level of plausibility we can provide. The average plausibility that a voter has, e.g. in Definition 39, is useful. Still, it would be more useful to guarantee that the voter always has a certain level of coercion-resistance. We leave a precise definition for future work.

9.5.3 Receipt-Freeness

Following [181], definition 39 also covers receipt-freeness. However, we again argue that modelling some variants is useful. For example the following definition is based on a swap of $\pi_{\mathsf{v}_{co}}$ and $\tilde{\pi}_{\mathsf{v}_{co}}$ in Definition 50, and models vote buyers who do not want to pay a "free lunch" to vote sellers who follow their own goal. The voter goal γ can here be to cast a specified vote or set of votes.

Definition 51 (Weak Vote Buying Resistance)**.** *For a given small p_{fl}, S achieves δ^{wvb}-coercion-resistance if for all dictated coerced strategies $\pi_{\mathsf{v}_{co}} \in V_S$ there exists a counter-strategy $\tilde{\pi}_{\mathsf{v}_{co}} \in V_S$ s.t. for all coercer programs $\pi_c \in C_S$:*

- $\Pr[(\pi_c || \tilde{\pi}_{\mathsf{v}_{co}} || \pi_v)^{(l)} \mapsto \gamma]$ *is overwhelming.*
- $\Pr[(\pi_c || \pi_{\mathsf{v}_{co}} || \pi_v)^{(l)} \mapsto 1] - \Pr[(\pi_c || \tilde{\pi}_{\mathsf{v}_{co}} || \pi_v)^{(l)} \mapsto 1]$ *is δ^{wvb}-bounded and* $\Pr[(\pi_c || \tilde{\pi}_{\mathsf{v}_{co}} || \pi_v)^{(l)} \mapsto 1]$ *is p_{fl}-bounded.*

We interpret outputting "1" as paying the vote seller, and this definition bounds how often an instruction-following vote seller gets paid by a vote-buyer (by $\delta^{wvb} + p_{\text{fl}}$), but under the condition that a voter who casts another vote is only paid with a (very) small probability p_{fl}. This is a weakened vote-buyer model but interesting since a vote buyer should avoid vote sellers going for a "free lunch". If the probability of an honest vote seller getting paid is low, it will help curb vote-selling (even though the vote buyer could increase the price and create a "vote selling lottery"). This definition, makes sense to drop the quantification over the coercer's strategies to see the resistance of vote to buy for different vote choices.

RLT. In the original RLT, a signature ballot will be revealed with probability $1 - f_{blind}$. If the vote buyer sees this, they can pay the vote seller. They will only pay the voter seller wrongly with a small probability p_{fl} equal to the probability that one of the honest voters cast the signature ballot, i.e. $\delta^{vb} \simeq 1 - f_{blind}$ which can be rather high and protects badly against vote-buying.

Masked RLT. For the masked ballots, we can, however, choose m such that several ballots will have the same masking as the signature ballot and makes it hard for the vote buyer to assess if the signature ballot was cast. For the student election, the number of matches with the optimal signature ballot $(0,1,1,1,1)$ is binomially distributed with an expectation value of 18.4 colliding ballots and a standard deviation of around 4.

For a more precise example, we can consider the three-candidate election with probabilities $(1/2, 1/2, 0)$ as above and assume that the goal of the voter is to cast 0 for candidate 1 and $p_{\text{fl}} = 0$. For $m = 1$, we will have $\delta^{vb} = 0$, but for $m = 2$, the vote-buyer can demand a vote for candidates 1 and 3 and payout if he sees $(1, *, 1)$. Any counter-strategy with 0 for candidate 1 gives $\delta^{vb} = 1/3$.

We note that the new quantitative definitions for no deniability coercion-resistance (Def. 50), the weak vote-buying resistance (Def. 51) and the original δ^{cr}-coercion-resistance (Def. 39) are considering different aspects of coercion-resistance and stating the three different δ-values gives a more nuanced description of the security of a given voting protocol. Also, note that the δ values are calculated using potentially different strategies for the coercer and voter. Finding unified strategies optimising the parameters is an interesting line of future work. Finally, there are natural, more fine-grained, definitions extending these which should also be considered in the future.

9.6 Conclusion

We have shown that the idea of risk-limiting tallies and risk-limiting verification can be applied effectively to complex ballots. Furthermore, we gain far greater flexibility in masking strategies by partially masking each ballot rather than simply masking a subset of the ballots as in the original RLT and RLV. This will be explored further to optimise the trade-offs between the various measures defined in future works.

The approach is more robust against any claims of being undemocratic: all ballots are counted, and indeed in the full tally/partial verification option, all are counted fully. The only compromise then is some reduction in the level of verifiability, but this can be adjusted and is probably acceptable. If we compare this with ThreeBallot, the chance of detecting a manipulated ballot is $1/3$, assuming that the attacker does not learn which ballot was retained by the voter. In our case, we can achieve a good level of coercion mitigation with, say, a shrouding of $\pm 1/2$ of each ballot

Finally, we did a preliminary analysis of the quantitative privacy for the different tally methods, and the coercion-resistance, particularly the probability a coerced voter gets undeniably caught. However, the new masked tallies are more appropriate for receipt-freeness, particularly with upper bounds on the number of vote sellers. In contrast, the old RLT provides good plausible deniability to coerced voters. This suggests combining both methods when possible, but future work is needed to define the precise level of vote-buying resistance.

Bibliography

[1] Michel Abdalla et al. "Better Security for Functional Encryption for Inner Product Evaluations". In: *IACR Cryptol. ePrint Arch.* (2016), p. 11. URL: http://eprint.iacr.org/2016/011.

[2] Michel Abdalla et al. "Decentralizing Inner-Product Functional Encryption". In: *Public-Key Cryptography - PKC 2019 - 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part II.* Ed. by Dongdai Lin and Kazue Sako. Vol. 11443. Lecture Notes in Computer Science. Springer, 2019, pp. 128–157. DOI: 10.1007/978-3-030-17259-6_5. URL: https://doi.org/10.1007/978-3-030-17259-6_5.

[3] Michel Abdalla et al. "Multi-Client Inner-Product Functional Encryption in the Random-Oracle Model". In: *Security and Cryptography for Networks - 12th International Conference, SCN 2020, Amalfi, Italy, September 14-16, 2020, Proceedings.* Ed. by Clemente Galdi and Vladimir Kolesnikov. Vol. 12238. Lecture Notes in Computer Science. Springer, 2020, pp. 525–545. DOI: 10.1007/978-3-030-57990-6_26. URL: https://doi.org/10.1007/978-3-030-57990-6_26.

[4] Michel Abdalla et al. "Multi-Input Functional Encryption for Inner Products: Function-Hiding Realizations and Constructions Without Pairings". In: *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I.* Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10991. Lecture Notes in Computer Science. Springer, 2018, pp. 597–627. DOI: 10.1007/978-3-319-96884-1_20. URL: https://doi.org/10.1007/978-3-319-96884-1_20.

[5] Michel Abdalla et al. "Multi-input Inner-Product Functional Encryption from Pairings". In: *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I.* Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10210. Lecture Notes in Computer Science. 2017, pp. 601–626. DOI: 10.1007/978-3-319-56620-7_21. URL: https://doi.org/10.1007/978-3-319-56620-7_21.

[6] Michel Abdalla et al. "Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions". In: *J. Cryptol.* 21.3 (2008), pp. 350–391. DOI: 10.1007/s00145-007-9006-6. URL: https://doi.org/10.1007/s00145-007-9006-6.

[7] Michel Abdalla et al. "Simple Functional Encryption Schemes for Inner Products". In: *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings.* Ed. by Jonathan Katz. Vol. 9020. Lecture Notes in Computer Science. Springer, 2015, pp. 733–751. DOI: 10.1007/978-3-662-46447-2_33. URL: https://doi.org/10.1007/978-3-662-46447-2_33.

[8] Claudia Z. Acemyan et al. "Usability of Voter Verifiable, End-to-end Voting Systems: Baseline Data for Helios, Prêt à Voter, and Scantegrity II". In: *2014 Electronic Voting Technology*

Workshop/Workshop on Trustworthy Elections, EVT/WOTE '14, San Diego, CA, USA, August 18-19, 2014. USENIX Association, 2014. URL: https://www.usenix.org/conference/evtwote14/workshop-program/presentation/acemyan.

[9] Dirk Achenbach et al. "Improved Coercion-Resistant Electronic Elections through Deniable Re-Voting". In: *USENIX Journal of Election Technology and Systems (JETS)* (Aug. 2015). URL: https://www.usenix.org/conference/jets15/workshop-program/presentation/achenbach.

[10] Ben Adida. "Helios: Web-based Open-Audit Voting". In: *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*. Ed. by Paul C. van Oorschot. USENIX Association, 2008, pp. 335–348. URL: http://www.usenix.org/events/sec08/tech/full_papers/adida/adida.pdf.

[11] Shweta Agrawal, Benoît Libert, and Damien Stehlé. "Fully Secure Functional Encryption for Inner Products, from Standard Assumptions". In: *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9816. Lecture Notes in Computer Science. Springer, 2016, pp. 333–362. DOI: 10.1007/978-3-662-53015-3_12. URL: https://doi.org/10.1007/978-3-662-53015-3_12.

[12] Shweta Agrawal et al. "Adaptive Simulation Security for Inner Product Functional Encryption". In: *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part I*. Ed. by Aggelos Kiayias et al. Vol. 12110. Lecture Notes in Computer Science. Springer, 2020, pp. 34–64. DOI: 10.1007/978-3-030-45374-9_2. URL: https://doi.org/10.1007/978-3-030-45374-9_2.

[13] Shweta Agrawal et al. "Efficient Public Trace and Revoke from Standard Assumptions: Extended Abstract". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. Ed. by Bhavani M. Thuraisingham et al. ACM, 2017, pp. 2277–2293. DOI: 10.1145/3133956.3134041. URL: https://doi.org/10.1145/3133956.3134041.

[14] Shweta Agrawal et al. "Functional Encryption: New Perspectives and Lower Bounds". In: *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. Lecture Notes in Computer Science. Springer, 2013, pp. 500–518. DOI: 10.1007/978-3-642-40084-1_28. URL: https://doi.org/10.1007/978-3-642-40084-1_28.

[15] Prabhanjan Ananth and Abhishek Jain. "Indistinguishability Obfuscation from Compact Functional Encryption". In: *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*. Ed. by Rosario Gennaro and Matthew Robshaw. Vol. 9215. Lecture Notes in Computer Science. Springer, 2015, pp. 308–326. DOI: 10.1007/978-3-662-47989-6_15. URL: https://doi.org/10.1007/978-3-662-47989-6_15.

[16] Prabhanjan Ananth et al. "From Selective to Adaptive Security in Functional Encryption". In: *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*. Ed. by Rosario Gennaro and Matthew Robshaw. Vol. 9216. Lecture Notes in Computer Science. Springer, 2015, pp. 657–677. DOI: 10.1007/978-3-662-48000-7_32. URL: https://doi.org/10.1007/978-3-662-48000-7_32.

[17] Roberto Araújo et al. "Remote Electronic Voting Can Be Efficient, Verifiable and Coercion-Resistant". In: *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*. Ed. by Jeremy Clark et al. Vol. 9604. Lecture Notes in Computer Science. Springer, 2016, pp. 224–232. DOI: 10.1007/978-3-662-53357-4_15. URL: https://doi.org/10.1007/978-3-662-53357-4_15.

[18] Nuttapong Attrapadung and Benoît Libert. "Functional encryption for public-attribute inner products: Achieving constant-size ciphertexts with adaptive security or support for negation". In: *J. Math. Cryptol.* 5.2 (2012), pp. 115–158. DOI: 10.1515/jmc.2011.009. URL: https://doi.org/10.1515/jmc.2011.009.

[19] Nuttapong Attrapadung et al. "Attribute-based encryption schemes with constant-size ciphertexts". In: *Theor. Comput. Sci.* 422 (2012), pp. 15–38. DOI: 10.1016/j.tcs.2011.12.004. URL: https://doi.org/10.1016/j.tcs.2011.12.004.

[20] L Babai. "Trading Group Theory for Randomness". In: *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*. STOC '85. Providence, Rhode Island, USA: Association for Computing Machinery, 1985, 421–429. ISBN: 0897911512. DOI: 10.1145/22145.22192. URL: https://doi.org/10.1145/22145.22192.

[21] Michael Backes, Martin Gagné, and Malte Skoruppa. "Using mobile device communication to strengthen e-Voting protocols". In: *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, November 4, 2013*. Ed. by Ahmad-Reza Sadeghi and Sara Foresti. ACM, 2013, pp. 237–242. DOI: 10.1145/2517840.2517863. URL: https://doi.org/10.1145/2517840.2517863.

[22] Saikrishna Badrinarayanan et al. "Multi-input Functional Encryption for Unbounded Arity Functions". In: *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9452. Lecture Notes in Computer Science. Springer, 2015, pp. 27–51. DOI: 10.1007/978-3-662-48797-6_2. URL: https://doi.org/10.1007/978-3-662-48797-6_2.

[23] Saikrishna Badrinarayanan et al. "Verifiable Functional Encryption". In: *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10032. Lecture Notes in Computer Science. 2016, pp. 557–587. DOI: 10.1007/978-3-662-53890-6_19. URL: https://doi.org/10.1007/978-3-662-53890-6_19.

[24] Alexandros Bakas and Antonis Michalas. "Multi-Input Functional Encryption: Efficient Applications from Symmetric Primitives". In: *19th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2020, Guangzhou, China, December 29, 2020 - January 1, 2021*. Ed. by Guojun Wang et al. IEEE, 2020, pp. 1105–1112. DOI: 10.1109/TrustCom50675.2020.00146. URL: https://doi.org/10.1109/TrustCom50675.2020.00146.

[25] Boaz Barak. "How to Go Beyond the Black-Box Simulation Barrier". In: *FOCS*. 2001, pp. 106–115. DOI: 10.1109/SFCS.2001.959885.

[26] Boaz Barak and Yehuda Lindell. "Strict Polynomial-Time in Simulation and Extraction". In: *SIAM J. Comput.* 33.4 (2004), pp. 738–818. DOI: 10.1137/S0097539703427975. URL: https://doi.org/10.1137/S0097539703427975.

[27] Razvan Barbulescu. “A Brief History of Pairings”. In: *Arithmetic of Finite Fields - 6th International Workshop, WAIFI 2016, Ghent, Belgium, July 13-15, 2016, Revised Selected Papers*. Ed. by Sylvain Duquesne and Svetla Petkova-Nikova. Vol. 10064. Lecture Notes in Computer Science. 2016, pp. 3–17. DOI: 10.1007/978-3-319-55227-9_1. URL: https://doi.org/10.1007/978-3-319-55227-9_1.

[28] Stephanie Bayer and Jens Groth. “Efficient Zero-Knowledge Argument for Correctness of a Shuffle”. In: *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. Springer, 2012, pp. 263–280.

[29] Stephanie Bayer and Jens Groth. “Zero-Knowledge Argument for Polynomial Evaluation with Application to Blacklists”. In: *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 646–663. DOI: 10.1007/978-3-642-38348-9_38. URL: https://doi.org/10.1007/978-3-642-38348-9_38.

[30] Donald Beaver. “Adaptive Zero Knowledge and Computational Equivocation (Extended Abstract)”. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*. Ed. by Gary L. Miller. ACM, 1996, pp. 629–638. DOI: 10.1145/237814.238014. URL: https://doi.org/10.1145/237814.238014.

[31] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. “Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements”. In: May 2000, pp. 259–274. ISBN: 978-3-540-67517-4. DOI: 10.1007/3-540-45539-6_18.

[32] Mihir Bellare and Phillip Rogaway. “Random Oracles are Practical: A Paradigm for Designing Efficient Protocols”. In: *CCS ’93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*. Ed. by Dorothy E. Denning et al. ACM, 1993, pp. 62–73. DOI: 10.1145/168588.168596. URL: https://doi.org/10.1145/168588.168596.

[33] Mihir Bellare and Moti Yung. “Certifying Permutations: Noninteractive Zero-Knowledge Based on Any Trapdoor Permutation”. In: *J. Cryptol.* 9.3 (1996), pp. 149–166.

[34] Michael Ben-Or et al. “Everything Provable is Provable in Zero-Knowledge”. In: *Advances in Cryptology - CRYPTO ’88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*. Ed. by Shafi Goldwasser. Vol. 403. Lecture Notes in Computer Science. Springer, 1988, pp. 37–56. DOI: 10.1007/0-387-34799-2_4. URL: https://doi.org/10.1007/0-387-34799-2_4.

[35] Josh Benaloh. “Ballot Casting Assurance via Voter-Initiated Poll Station Auditing”. In: *Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology*. EVT’07. Boston, MA: USENIX Association, 2007, p. 14.

[36] Josh Benaloh. “Verifiable Secret-Ballot Elections”. PhD thesis. 1987. URL: https://www.microsoft.com/en-us/research/publication/verifiable-secret-ballot-elections/.

[37] Josh Benaloh, Philip B Stark, and Vanessa Teague. “VAULT: Verifiable Audits Using Limited Transparency”. In: *E-Vote-ID 2019* (2019), p. 69.

[38] Josh Benaloh et al. “SOBA: Secrecy-preserving Observable Ballot-level Audit”. In: *2011 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 11)*. San Francisco, CA: USENIX Association, Aug. 2011. URL: https://www.usenix.org/conference/evtwote-11/soba-secrecy-preserving-observable-ballot-level-audit.

[39] Josh Benaloh et al. “STAR-Vote: A Secure, Transparent, Auditable, and Reliable Voting System”. In: *CoRR* abs/1211.1904 (2012). arXiv: 1211.1904. URL: http://arxiv.org/abs/1211.1904.

[40] Josh Cohen Benaloh and Dwight Tuinstra. “Receipt-free secret-ballot elections (extended abstract)”. In: *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*. Ed. by Frank Thomson Leighton and Michael T. Goodrich. ACM, 1994, pp. 544–553. DOI: 10.1145/195058.195407. URL: https://doi.org/10.1145/195058.195407.

[41] Fabrice Benhamouda, Florian Bourse, and Helger Lipmaa. “CCA-Secure Inner-Product Functional Encryption from Projective Hash Functions”. In: *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part II*. Ed. by Serge Fehr. Vol. 10175. Lecture Notes in Computer Science. Springer, 2017, pp. 36–66. DOI: 10.1007/978-3-662-54388-7_2. URL: https://doi.org/10.1007/978-3-662-54388-7_2.

[42] Fabrice Benhamouda et al. “Implicit Zero-Knowledge Arguments and Applications to the Malicious Setting”. In: *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*. Ed. by Rosario Gennaro and Matthew Robshaw. Vol. 9216. Lecture Notes in Computer Science. Springer, 2015, pp. 107–129. DOI: 10.1007/978-3-662-48000-7_6. URL: https://doi.org/10.1007/978-3-662-48000-7_6.

[43] David Bernhard, Olivier Pereira, and Bogdan Warinschi. “How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios”. In: *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. Lecture Notes in Computer Science. Springer, 2012, pp. 626–643. DOI: 10.1007/978-3-642-34961-4_38. URL: https://doi.org/10.1007/978-3-642-34961-4_38.

[44] David Bernhard et al. “Adapting Helios for Provable Ballot Privacy”. In: *Computer Security - ESORICS 2011 - 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings*. Ed. by Vijay Atluri and Claudia Díaz. Vol. 6879. Lecture Notes in Computer Science. Springer, 2011, pp. 335–354. DOI: 10.1007/978-3-642-23822-2_19. URL: https://doi.org/10.1007/978-3-642-23822-2_19.

[45] David Bernhard et al. “Measuring vote privacy, revisited”. In: *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*. Ed. by Ting Yu, George Danezis, and Virgil D. Gligor. ACM, 2012, pp. 941–952. DOI: 10.1145/2382196.2382295. URL: https://doi.org/10.1145/2382196.2382295.

[46] David Bernhard et al. “SoK: A Comprehensive Analysis of Game-Based Ballot Privacy Definitions”. In: *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*. IEEE Computer Society, 2015, pp. 499–516. DOI: 10.1109/SP.2015.37. URL: https://doi.org/10.1109/SP.2015.37.

[47] John Bethencourt, Amit Sahai, and Brent Waters. "Ciphertext-Policy Attribute-Based Encryption". In: *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*. IEEE Computer Society, 2007, pp. 321–334. DOI: 10.1109/SP.2007.11. URL: https://doi.org/10.1109/SP.2007.11.

[48] Olivier Blazy et al. "Batch Groth-Sahai". In: *Applied Cryptography and Network Security, 8th International Conference, ACNS 2010, Beijing, China, June 22-25, 2010. Proceedings*. Ed. by Jianying Zhou and Moti Yung. Vol. 6123. Lecture Notes in Computer Science. 2010, pp. 218–235. DOI: 10.1007/978-3-642-13708-2_14. URL: https://doi.org/10.1007/978-3-642-13708-2_14.

[49] Manuel Blum. "Coin Flipping by Telephone". In: *Advances in Cryptology: A Report on CRYPTO 81*. 1981, pp. 11–15. URL: /archive/crypto81/11_blum.pdf.

[50] Florian Böhl et al. "Practical Signatures from Standard Assumptions". In: *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 461–485. DOI: 10.1007/978-3-642-38348-9_28. URL: https://doi.org/10.1007/978-3-642-38348-9_28.

[51] Dan Boneh, Xavier Boyen, and Hovav Shacham. "Short Group Signatures". In: *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*. Ed. by Matthew K. Franklin. Vol. 3152. Lecture Notes in Computer Science. Springer, 2004, pp. 41–55. DOI: 10.1007/978-3-540-28628-8_3. URL: https://doi.org/10.1007/978-3-540-28628-8_3.

[52] Dan Boneh and Matthew K. Franklin. "Efficient Generation of Shared RSA Keys (Extended Abstract)". In: *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*. Ed. by Burton S. Kaliski Jr. Vol. 1294. Lecture Notes in Computer Science. Springer, 1997, pp. 425–439. DOI: 10.1007/BFb0052253. URL: https://doi.org/10.1007/BFb0052253.

[53] Dan Boneh and Matthew K. Franklin. "Identity-Based Encryption from the Weil Pairing". In: *SIAM J. Comput.* 32.3 (2003), pp. 586–615. DOI: 10.1137/S0097539701398521. URL: https://doi.org/10.1137/S0097539701398521.

[54] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. "Evaluating 2-DNF Formulas on Ciphertexts". In: *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*. Ed. by Joe Kilian. Vol. 3378. Lecture Notes in Computer Science. Springer, 2005, pp. 325–341. DOI: 10.1007/978-3-540-30576-7_18. URL: https://doi.org/10.1007/978-3-540-30576-7_18.

[55] Dan Boneh, Amit Sahai, and Brent Waters. "Functional Encryption: Definitions and Challenges". In: *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*. Ed. by Yuval Ishai. Vol. 6597. Lecture Notes in Computer Science. Springer, 2011, pp. 253–273. DOI: 10.1007/978-3-642-19571-6_16. URL: https://doi.org/10.1007/978-3-642-19571-6_16.

[56] Dan Boneh and Brent Waters. "Conjunctive, Subset, and Range Queries on Encrypted Data". In: *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*. Ed. by Salil P. Vadhan. Vol. 4392. Lecture Notes in Computer Science. Springer, 2007, pp. 535–554. DOI: 10.1007/978-3-540-70936-7_29. URL: https://doi.org/10.1007/978-3-540-70936-7_29.

[57] Dan Boneh et al. “Chosen-Ciphertext Security from Identity-Based Encryption”. In: *SIAM J. Comput.* 36.5 (2007), pp. 1301–1328. DOI: `10.1137/S009753970544713X`. URL: `https://doi.org/10.1137/S009753970544713X`.

[58] Dan Boneh et al. “On the Impossibility of Basing Identity Based Encryption on Trapdoor Permutations”. In: *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA.* IEEE Computer Society, 2008, pp. 283–292. DOI: `10.1109/FOCS.2008.67`. URL: `https://doi.org/10.1109/FOCS.2008.67`.

[59] Dan Boneh et al. “Public Key Encryption with Keyword Search”. In: *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings.* Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. Lecture Notes in Computer Science. Springer, 2004, pp. 506–522. DOI: `10.1007/978-3-540-24676-3\_30`. URL: `https://doi.org/10.1007/978-3-540-24676-3\_30`.

[60] Dan Boneh et al. “Semantically Secure Order-Revealing Encryption: Multi-input Functional Encryption Without Obfuscation”. In: *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II.* Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. Lecture Notes in Computer Science. Springer, 2015, pp. 563–594. DOI: `10.1007/978-3-662-46803-6\_19`. URL: `https://doi.org/10.1007/978-3-662-46803-6\_19`.

[61] Fabrice Boudot. “Efficient Proofs that a Committed Number Lies in an Interval”. In: *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding.* Ed. by Bart Preneel. Vol. 1807. Lecture Notes in Computer Science. Springer, 2000, pp. 431–444. DOI: `10.1007/3-540-45539-6\_31`. URL: `https://doi.org/10.1007/3-540-45539-6\_31`.

[62] Zvika Brakerski, Ilan Komargodski, and Gil Segev. “Multi-input Functional Encryption in the Private-Key Setting: Stronger Security from Weaker Assumptions”. In: *J. Cryptol.* 31.2 (2018), pp. 434–520. DOI: `10.1007/s00145-017-9261-0`. URL: `https://doi.org/10.1007/s00145-017-9261-0`.

[63] Gilles Brassard, David Chaum, and Claude Crépeau. “Minimum Disclosure Proofs of Knowledge”. In: *J. Comput. Syst. Sci.* 37.2 (1988), pp. 156–189. DOI: `10.1016/0022-0000(88)90005-0`. URL: `https://doi.org/10.1016/0022-0000(88)90005-0`.

[64] Johannes Buchmann, Stephan Düllmann, and Hugh C. Williams. “On the Complexity and Efficiency of a New Key Exchange System”. In: *Advances in Cryptology - EUROCRYPT ’89, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, Proceedings.* Ed. by Jean-Jacques Quisquater and Joos Vandewalle. Vol. 434. Lecture Notes in Computer Science. Springer, 1989, pp. 597–616. DOI: `10.1007/3-540-46885-4\_57`. URL: `https://doi.org/10.1007/3-540-46885-4\_57`.

[65] Johannes Buchmann and Safuat Hamdy. “A survey on IQ cryptography”. In: *Public-Key Cryptography and Computational Number Theory: Proceedings of the International Conference organized by the Stefan Banach International Mathematical Center Warsaw, Poland, September 11-15, 2000.* Ed. by Kazimierz Alster, Jerzy Urbanowicz, and Hugh C. Williams. De Gruyter, 2011, pp. 1–16. DOI: `doi:10.1515/9783110881035.1`. URL: `https://doi.org/10.1515/9783110881035.1`.

[66] Johannes Buchmann and Ulrich Vollmer. *Binary Quadratic Forms - an Algorithmic Approach*. Vol. 20. Algorithms and computation in mathematics. Springer, 2007. ISBN: 978-3-540-46367-2.

[67] Johannes Buchmann and Hugh C. Williams. “A Key-Exchange System Based on Imaginary Quadratic Fields”. In: *J. Cryptol.* 1.2 (1988), pp. 107–118. DOI: 10.1007/BF02351719. URL: https://doi.org/10.1007/BF02351719.

[68] Jan Camenisch and Markus Stadler. “Proof systems for general statements about discrete logarithms”. In: *Technical Report/ETH Zurich, Department of Computer Science* 260 (1997).

[69] Ran Canetti, Oded Goldreich, and Shai Halevi. “The Random Oracle Methodology, Revisited (Preliminary Version)”. In: *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*. Ed. by Jeffrey Scott Vitter. ACM, 1998, pp. 209–218. DOI: 10.1145/276698.276741. URL: https://doi.org/10.1145/276698.276741.

[70] Ran Canetti et al. “Fiat-Shamir: From Practice to Theory”. In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2019. Phoenix, AZ, USA: Association for Computing Machinery, 2019, 1082–1090. ISBN: 9781450367059. DOI: 10.1145/3313276.3316380. URL: https://doi.org/10.1145/3313276.3316380.

[71] Angelo De Caro et al. “On the Achievability of Simulation-Based Security for Functional Encryption”. In: *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. Lecture Notes in Computer Science. Springer, 2013, pp. 519–535. DOI: 10.1007/978-3-642-40084-1_29. URL: https://doi.org/10.1007/978-3-642-40084-1_29.

[72] Guilhem Castagnos and Fabien Laguillaumie. “Linearly Homomorphic Encryption from DDH”. In: *Topics in Cryptology - CT-RSA 2015, The Cryptographer’s Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*. Ed. by Kaisa Nyberg. Vol. 9048. Lecture Notes in Computer Science. Springer, 2015, pp. 487–505. DOI: 10.1007/978-3-319-16715-2_26. URL: https://doi.org/10.1007/978-3-319-16715-2_26.

[73] Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. “Practical Fully Secure Unrestricted Inner Product Functional Encryption Modulo p”. In: *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*. Ed. by Thomas Peyrin and Steven D. Galbraith. Vol. 11273. Lecture Notes in Computer Science. Springer, 2018, pp. 733–764. DOI: 10.1007/978-3-030-03329-3_25. URL: https://doi.org/10.1007/978-3-030-03329-3_25.

[74] Pyrros Chaidos and Geoffroy Couteau. “Efficient Designated-Verifier Non-interactive Zero-Knowledge Proofs of Knowledge”. In: *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10822. Lecture Notes in Computer Science. Springer, 2018, pp. 193–221. DOI: 10.1007/978-3-319-78372-7_7. URL: https://doi.org/10.1007/978-3-319-78372-7_7.

[75] Melissa Chase, Chaya Ganesh, and Payman Mohassel. “Efficient Zero-Knowledge Proof of Algebraic and Non-Algebraic Statements with Applications to Privacy Preserving Credentials”. In: *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*. Ed.

by Matthew Robshaw and Jonathan Katz. Vol. 9816. Lecture Notes in Computer Science. Springer, 2016, pp. 499–530. DOI: 10.1007/978-3-662-53015-3_18. URL: https://doi.org/10.1007/978-3-662-53015-3_18.

[76] David Chaum. "Secret-Ballot Receipts: True Voter-Verifiable Elections". In: *IEEE Secur. Priv.* 2.1 (2004), pp. 38–47. DOI: 10.1109/MSECP.2004.1264852. URL: https://doi.org/10.1109/MSECP.2004.1264852.

[77] David Chaum and Torben P. Pedersen. "Wallet Databases with Observers". In: *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*. Ed. by Ernest F. Brickell. Vol. 740. Lecture Notes in Computer Science. Springer, 1992, pp. 89–105. DOI: 10.1007/3-540-48071-4_7. URL: https://doi.org/10.1007/3-540-48071-4_7.

[78] David Chaum et al. "Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes". In: *2008 USENIX/ACCURATE Electronic Voting Workshop, EVT 2008, July 28-29, 2008, San Jose, CA, USA, Proceedings*. Ed. by David L. Dill and Tadayoshi Kohno. USENIX Association, 2008. URL: http://www.usenix.org/events/evt08/tech/full_papers/chaum/chaum.pdf.

[79] Jeremy Clark and Urs Hengartner. "Selections: Internet Voting with Over-the-Shoulder Coercion-Resistance". In: *Financial Cryptography and Data Security - 15th International Conference, FC 2011, Gros Islet, St. Lucia, February 28 - March 4, 2011, Revised Selected Papers*. Ed. by George Danezis. Vol. 7035. Lecture Notes in Computer Science. Springer, 2011, pp. 47–61. DOI: 10.1007/978-3-642-27576-0_4. URL: https://doi.org/10.1007/978-3-642-27576-0_4.

[80] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. "Civitas: Toward a Secure Voting System". In: *2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA*. IEEE Computer Society, 2008, pp. 354–368. DOI: 10.1109/SP.2008.32. URL: https://doi.org/10.1109/SP.2008.32.

[81] Clifford Cocks. "An identity based encryption scheme based on quadratic residues". In: *IMA International Conference on Cryptography and Coding*. Springer. 2001, pp. 360–363.

[82] U.S. Department of Commerce, National Institute of Standards, and Technology. *Secure Hash Standard - SHS: Federal Information Processing Standards Publication 180-4*. North Charleston, SC, USA: CreateSpace Independent Publishing Platform, 2012. ISBN: 1478178078.

[83] Véronique Cortier, Pierrick Gaudry, and Stéphane Glondu. "Belenios: A Simple Private and Verifiable Electronic Voting System". In: *Foundations of Security, Protocols, and Equational Reasoning - Essays Dedicated to Catherine A. Meadows*. Ed. by Joshua D. Guttman et al. Vol. 11565. Lecture Notes in Computer Science. Springer, 2019, pp. 214–238. DOI: 10.1007/978-3-030-19052-1_14. URL: https://doi.org/10.1007/978-3-030-19052-1_14.

[84] Véronique Cortier et al. "Election Verifiability for Helios under Weaker Trust Assumptions". In: *Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part II*. Ed. by Miroslaw Kutylowski and Jaideep Vaidya. Vol. 8713. Lecture Notes in Computer Science. Springer, 2014, pp. 327–344. DOI: 10.1007/978-3-319-11212-1_19. URL: https://doi.org/10.1007/978-3-319-11212-1_19.

[85] Véronique Cortier et al. "SoK: Verifiability Notions for E-Voting Protocols". In: *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*. IEEE Computer Society, 2016, pp. 779–798. DOI: 10.1109/SP.2016.52. URL: https://doi.org/10.1109/SP.2016.52.

[86] Ronald Cramer and Ivan Damgård. “Linear Zero-Knowledge - A Note on Efficient Zero-Knowledge Proofs and Arguments”. In: *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*. Ed. by Frank Thomson Leighton and Peter W. Shor. ACM, 1997, pp. 436–445. DOI: 10.1145/258533.258635. URL: https://doi.org/10.1145/258533.258635.

[87] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. “Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols”. In: *Advances in Cryptology - CRYPTO ’94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*. Ed. by Yvo Desmedt. Vol. 839. Lecture Notes in Computer Science. Springer, 1994, pp. 174–187. DOI: 10.1007/3-540-48658-5_19. URL: https://doi.org/10.1007/3-540-48658-5_19.

[88] Ivan Damgård. “Commitment Schemes and Zero-Knowledge Protocols”. In: *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998*. Ed. by Ivan Damgård. Vol. 1561. Lecture Notes in Computer Science. Springer, 1998, pp. 63–86. DOI: 10.1007/3-540-48969-X_3. URL: https://doi.org/10.1007/3-540-48969-X_3.

[89] Ivan Damgård. “Efficient Concurrent Zero-Knowledge in the Auxiliary String Model”. In: *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*. Ed. by Bart Preneel. Vol. 1807. Lecture Notes in Computer Science. Springer, 2000, pp. 418–430. DOI: 10.1007/3-540-45539-6_30. URL: https://doi.org/10.1007/3-540-45539-6_30.

[90] Angelo De Caro and Vincenzo Iovino. “On the power of rewinding simulators in functional encryption”. In: *Designs, Codes and Cryptography* (2016), pp. 1–27. ISSN: 1573-7586. DOI: 10.1007/s10623-016-0272-x. URL: http://dx.doi.org/10.1007/s10623-016-0272-x.

[91] Sumit Debnath et al. “Post-Quantum Secure Inner Product Functional Encryption Using Multivariate Public Key Cryptography”. In: *Mediterranean Journal of Mathematics* 18 (Oct. 2021). DOI: 10.1007/s00009-021-01841-2.

[92] S. Delaune, S. Kremer, and M. D. Ryan. “Verifying Privacy-type Properties of Electronic Voting Protocols”. In: *Journal of Computer Security* 17.4 (2009), pp. 435–487.

[93] Apoorvaa Deshpande and Yael Kalai. “Proofs of Ignorance and Applications to 2-Message Witness Hiding”. In: *IACR Cryptol. ePrint Arch.* (2018), p. 896. URL: https://eprint.iacr.org/2018/896.

[94] W. Diffie and M.E. Hellman. “New Directions in Cryptography”. In: *IEEE Transactions on Information Theory* IT-22.6 (1976), pp. 644–654.

[95] Jeremy Epstein. “Weakness in Depth: A Voting Machine’s Demise”. In: *IEEE Secur. Priv.* 13.3 (2015), pp. 55–58. DOI: 10.1109/MSP.2015.46. URL: https://doi.org/10.1109/MSP.2015.46.

[96] Alex Escala and Jens Groth. “Fine-Tuning Groth-Sahai Proofs”. In: *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*. Ed. by Hugo Krawczyk. Vol. 8383. Lecture Notes in Computer Science. Springer, 2014, pp. 630–649. DOI: 10.1007/978-3-642-54631-0_36. URL: https://doi.org/10.1007/978-3-642-54631-0_36.

[97] Aleksander Essex, Jeremy Clark, and Urs Hengartner. "Cobra: Toward Concurrent Ballot Authorization for Internet Voting". In: *2012 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '12, Bellevue, WA, USA, August 6-7, 2012.* Ed. by J. Alex Halderman and Olivier Pereira. USENIX Association, 2012. URL: `https://www.usenix.org/conference/evtwote12/workshop-program/presentation/essex`.

[98] Ehsan Estaji et al. "Revisiting Practical and Usable Coercion-Resistant Remote E-Voting". In: *Electronic Voting - 5th International Joint Conference, E-Vote-ID 2020, Bregenz, Austria, October 6-9, 2020, Proceedings.* Ed. by Robert Krimmer et al. Vol. 12455. Lecture Notes in Computer Science. Springer, 2020, pp. 50–66.

[99] Europarat, ed. *Legal, operational and technical standards for e-voting: Recommendation Rec(2004)11 adopted by the Committee of Ministers of the Council of Europe on 30 September 2004 and explanatory memorandum.* eng. Reprinted, January 2008. Recommendation / Committee of Ministers of the Council of Europe Rec 2004, 11. Strasbourg: Council of Europe Publ, 2008. ISBN: 9789287156358.

[100] C. Feier, S. Neumann, and M. Volkamer. "Coercion-Resistant Internet Voting in Practice". In: *44. Jahrestagung der Gesellschaft für Informatik, Informatik 2014, Big Data - Komplexität meistern, 2014.* Ed. by E. Plödereder et al. Vol. P-232. LNI. GI, 2014, pp. 1401–1414. URL: `https://dl.gi.de/20.500.12116/2749`.

[101] Uriel Feige, Dror Lapidot, and Adi Shamir. "Multiple Non-Interactive Zero Knowledge Proofs Based on a Single Random String (Extended Abstract)". In: *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I.* IEEE Computer Society, 1990, pp. 308–317.

[102] Uriel Feige and Adi Shamir. "Witness Indistinguishable and Witness Hiding Protocols". In: *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA.* Ed. by Harriet Ortiz. ACM, 1990, pp. 416–426. DOI: `10.1145/100216.100272`. URL: `https://doi.org/10.1145/100216.100272`.

[103] Ariel J. Feldman, J. Alex Halderman, and Edward W. Felten. "Security Analysis of the Diebold AccuVote-TS Voting Machine". In: *2007 USENIX/ACCURATE Electronic Voting Technology Workshop, EVT'07, Boston, MA, USA, August 6, 2007.* Ed. by Ray Martinez and David A. Wagner. USENIX Association, 2007. URL: `https://www.usenix.org/conference/evt-07/security-analysis-diebold-accuvote-ts-voting-machine`.

[104] Amos Fiat and Adi Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings.* Ed. by Andrew M. Odlyzko. Vol. 263. Lecture Notes in Computer Science. Springer, 1986, pp. 186–194. DOI: `10.1007/3-540-47721-7\_12`. URL: `https://doi.org/10.1007/3-540-47721-7\_12`.

[105] Lance Fortnow. "The Complexity of Perfect Zero-Knowledge". In: *Adv. Comput. Res.* 5 (1989), pp. 327–343.

[106] Tore Kasper Frederiksen, Jesper Buus Nielsen, and Claudio Orlandi. "Privacy-Free Garbled Circuits with Applications to Efficient Zero-Knowledge". In: *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II.* Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. Lecture Notes in Computer Science. Springer, 2015, pp. 191–219. DOI: `10.1007/978-3-662-46803-6\_7`. URL: `https://doi.org/10.1007/978-3-662-46803-6\_7`.

[107] Gerhard Frey and Hans-Georg Rück. "A Remark Concerning m-Divisibility and the Discrete Logarithm in the Divisor Class Group of Curves". In: *Mathematics of Computation* 62.206 (1994), pp. 865–874. ISSN: 00255718, 10886842. URL: http://www.jstor.org/stable/2153546 (visited on 2022-05-31).

[108] Martin Fürer et al. "On Completeness and Soundness in Interactive Proof Systems". In: *Adv. Comput. Res.* 5 (1989), pp. 429–442.

[109] Jun Furukawa and Kazue Sako. "An Efficient Scheme for Proving a Shuffle". In: *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings.* Ed. by Joe Kilian. Vol. 2139. Lecture Notes in Computer Science. Springer, 2001, pp. 368–387. DOI: 10.1007/3-540-44647-8_22. URL: https://doi.org/10.1007/3-540-44647-8_22.

[110] Steven D. Galbraith. "The Weil pairing on elliptic curves over C". In: *IACR Cryptol. ePrint Arch.* (2005), p. 323. URL: http://eprint.iacr.org/2005/323.

[111] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. "Pairings for cryptographers". In: *Discret. Appl. Math.* 156.16 (2008), pp. 3113–3121. DOI: 10.1016/j.dam.2007.12.010. URL: https://doi.org/10.1016/j.dam.2007.12.010.

[112] Taher El Gamal. "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms". In: *IEEE Trans. Inf. Theory* 31.4 (1985), pp. 469–472. DOI: 10.1109/TIT.1985.1057074. URL: https://doi.org/10.1109/TIT.1985.1057074.

[113] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. "Strengthening Zero-Knowledge Protocols Using Signatures". In: *J. Cryptol.* 19.2 (2006), pp. 169–209. DOI: 10.1007/s00145-005-0307-3. URL: https://doi.org/10.1007/s00145-005-0307-3.

[114] Sanjam Garg, Craig Gentry, and Shai Halevi. "Candidate multilinear maps from ideal lattices". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques.* Springer. 2013, pp. 1–17.

[115] Sanjam Garg et al. "Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits". In: *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA.* IEEE Computer Society, 2013, pp. 40–49. DOI: 10.1109/FOCS.2013.13. URL: https://doi.org/10.1109/FOCS.2013.13.

[116] Sanjam Garg et al. "Functional Encryption Without Obfuscation". In: *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II.* Ed. by Eyal Kushilevitz and Tal Malkin. Vol. 9563. Lecture Notes in Computer Science. Springer, 2016, pp. 480–511. DOI: 10.1007/978-3-662-49099-0_18. URL: https://doi.org/10.1007/978-3-662-49099-0_18.

[117] Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. "Groth-Sahai Proofs Revisited". In: *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings.* Ed. by Phong Q. Nguyen and David Pointcheval. Vol. 6056. Lecture Notes in Computer Science. Springer, 2010, pp. 177–192. DOI: 10.1007/978-3-642-13013-7_11. URL: https://doi.org/10.1007/978-3-642-13013-7_11.

[118] Oded Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness.* Vol. 17. Algorithms and Combinatorics. Springer, 1998. ISBN: 978-3-540-64766-9. DOI: 10.1007/978-3-662-12521-2. URL: https://doi.org/10.1007/978-3-662-12521-2.

[119] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques.* Cambridge University Press, 2001. ISBN: 0-521-79172-3. DOI: 10.1017/CBO9780511546891. URL: http://www.wisdom.weizmann.ac.il/\%7Eoded/foc-vol1.html.

[120] Oded Goldreich. "Zero-Knowledge twenty years after its invention". In: *Electron. Colloquium Comput. Complex.* 063 (2002). URL: https://eccc.weizmann.ac.il/eccc-reports/2002/TR02-063/index.html.

[121] Oded Goldreich and Hugo Krawczyk. "On the composition of zero-knowledge proof systems". In: *Automata, Languages and Programming*. Ed. by Michael S. Paterson. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 268–282. ISBN: 978-3-540-47159-2.

[122] Oded Goldreich, Yishay Mansour, and Michael Sipser. "Interactive proof systems: Provers that never fail and random selection". In: *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*. 1987, pp. 449–461. DOI: 10.1109/SFCS.1987.35.

[123] Oded Goldreich, Silvio Micali, and Avi Wigderson. "How to play any mental game, or a completeness theorem for protocols with honest majority". In: *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*. Ed. by Oded Goldreich. ACM, 2019, pp. 307–328. DOI: 10.1145/3335741.3335755. URL: https://doi.org/10.1145/3335741.3335755.

[124] Oded Goldreich, Silvio Micali, and Avi Wigderson. "Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design (Extended Abstract)". In: *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*. IEEE Computer Society, 1986, pp. 174–187. DOI: 10.1109/SFCS.1986.47. URL: https://doi.org/10.1109/SFCS.1986.47.

[125] Oded Goldreich, Silvio Micali, and Avi Wigderson. "Proofs That Yield Nothing but Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems". In: *J. ACM* 38.3 (July 1991), 690–728. ISSN: 0004-5411. DOI: 10.1145/116825.116852. URL: https://doi.org/10.1145/116825.116852.

[126] Oded Goldreich and Yair Oren. "Definitions and Properties of Zero-Knowledge Proof Systems". In: *J. Cryptol.* 7.1 (1994), pp. 1–32. DOI: 10.1007/BF00195207. URL: https://doi.org/10.1007/BF00195207.

[127] Shafi Goldwasser and Silvio Micali. "Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information". In: *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*. Ed. by Harry R. Lewis et al. ACM, 1982, pp. 365–377. DOI: 10.1145/800070.802212. URL: https://doi.org/10.1145/800070.802212.

[128] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The Knowledge Complexity of Interactive Proof Systems". In: *SIAM J. Comput.* 18.1 (1989), pp. 186–208. DOI: 10.1137/0218012. URL: https://doi.org/10.1137/0218012.

[129] Shafi Goldwasser and Michael Sipser. "Private Coins versus Public Coins in Interactive Proof Systems". In: *Adv. Comput. Res.* 5 (1989), pp. 73–90.

[130] Shafi Goldwasser et al. "How to Run Turing Machines on Encrypted Data". In: *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. Lecture Notes in Computer Science. Springer, 2013, pp. 536–553. DOI: 10.1007/978-3-642-40084-1_30. URL: https://doi.org/10.1007/978-3-642-40084-1_30.

[131] Shafi Goldwasser et al. "Multi-input Functional Encryption". In: *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. Lecture Notes in Computer Science. Springer, 2014, pp. 578–602. DOI: 10.1007/978-3-642-55220-5_32. URL: https://doi.org/10.1007/978-3-642-55220-5_32.

[132] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. “Attribute-based encryption for circuits”. In: *STOC*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM, 2013, pp. 545–554. ISBN: 978-1-4503-2029-0.

[133] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. “Functional Encryption with Bounded Collusions via Multi-party Computation”. In: *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. Lecture Notes in Computer Science. Springer, 2012, pp. 162–179. DOI: 10.1007/978-3-642-32009-5_11. URL: https://doi.org/10.1007/978-3-642-32009-5_11.

[134] Vipul Goyal et al. “Attribute-based encryption for fine-grained access control of encrypted data”. In: *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, October 30 - November 3, 2006*. Ed. by Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati. ACM, 2006, pp. 89–98. DOI: 10.1145/1180405.1180418. URL: https://doi.org/10.1145/1180405.1180418.

[135] Vipul Goyal et al. “Bounded Ciphertext Policy Attribute Based Encryption”. In: *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*. Ed. by Luca Aceto et al. Vol. 5126. Lecture Notes in Computer Science. Springer, 2008, pp. 579–591. DOI: 10.1007/978-3-540-70583-3_47. URL: https://doi.org/10.1007/978-3-540-70583-3_47.

[136] Panagiotis Grontas et al. “Towards Everlasting Privacy and Efficient Coercion Resistance in Remote Electronic Voting”. In: *Financial Cryptography and Data Security - FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers*. Ed. by Aviv Zohar et al. Vol. 10958. Lecture Notes in Computer Science. Springer, 2018, pp. 210–231. DOI: 10.1007/978-3-662-58820-8_15. URL: https://doi.org/10.1007/978-3-662-58820-8_15.

[137] Jens Groth. “Efficient Zero-Knowledge Arguments from Two-Tiered Homomorphic Commitments”. In: *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. Lecture Notes in Computer Science. Springer, 2011, pp. 431–448. DOI: 10.1007/978-3-642-25385-0_23. URL: https://doi.org/10.1007/978-3-642-25385-0_23.

[138] Jens Groth. “Linear Algebra with Sub-linear Zero-Knowledge Arguments”. In: *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*. Ed. by Shai Halevi. Vol. 5677. Lecture Notes in Computer Science. Springer, 2009, pp. 192–208. DOI: 10.1007/978-3-642-03356-8_12. URL: https://doi.org/10.1007/978-3-642-03356-8_12.

[139] Jens Groth. “Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures”. In: *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*. Ed. by Xuejia Lai and Kefei Chen. Vol. 4284. Lecture Notes in Computer Science. Springer, 2006, pp. 444–459. DOI: 10.1007/11935230_29. URL: https://doi.org/10.1007/11935230_29.

[140] Jens Groth, Rafail Ostrovsky, and Amit Sahai. “New Techniques for Noninteractive Zero-Knowledge”. In: *J. ACM* 59.3 (2012), 11:1–11:35. DOI: 10.1145/2220357.2220358. URL: https://doi.org/10.1145/2220357.2220358.

[141] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "Non-interactive Zaps and New Techniques for NIZK". In: *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings.* Ed. by Cynthia Dwork. Vol. 4117. Lecture Notes in Computer Science. Springer, 2006, pp. 97–111. DOI: 10.1007/11818175_6. URL: https://doi.org/10.1007/11818175_6.

[142] Jens Groth, Rafail Ostrovsky, and Amit Sahai. "Perfect Non-interactive Zero Knowledge for NP". In: *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings.* Ed. by Serge Vaudenay. Vol. 4004. Lecture Notes in Computer Science. Springer, 2006, pp. 339–358. DOI: 10.1007/11761679_21. URL: https://doi.org/10.1007/11761679_21.

[143] Jens Groth and Amit Sahai. "Efficient Noninteractive Proof Systems for Bilinear Groups". In: *SIAM J. Comput.* 41.5 (2012), pp. 1193–1232. DOI: 10.1137/080725386. URL: https://doi.org/10.1137/080725386.

[144] Rolf Haenni and Oliver Spycher. "Secure Internet Voting on Limited Devices with Anonymized DSA Public Keys". In: *2011 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 11).* San Francisco, CA: USENIX Association, Aug. 2011. URL: https://www.usenix.org/conference/evtwote-11/secure-internet-voting-limited-devices-anonymized-dsa-public-keys.

[145] Thomas Haines and Johannes Müller. "A Novel Proof of Shuffle: Exponentially Secure Cut-and-Choose". In: *Information Security and Privacy - 26th Australasian Conference, ACISP 2021, Virtual Event, December 1-3, 2021, Proceedings.* Ed. by Joonsang Baek and Sushmita Ruj. Vol. 13083. Lecture Notes in Computer Science. Springer, 2021, pp. 293–308. DOI: 10.1007/978-3-030-90567-5_15. URL: https://doi.org/10.1007/978-3-030-90567-5_15.

[146] Thomas Haines and Johannes Müller. "How not to VoteAgain: Pitfalls of Scalable Coercion-Resistant E-Voting". In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 1406. URL: https://eprint.iacr.org/2020/1406.

[147] Thomas Haines et al. "How Not to Prove Your Election Outcome". In: *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020.* IEEE, 2020, pp. 644–660. DOI: 10.1109/SP40000.2020.00048. URL: https://doi.org/10.1109/SP40000.2020.00048.

[148] Iftach Haitner and Omer Reingold. "Statistically-hiding commitment from any one-way function". In: *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007.* Ed. by David S. Johnson and Uriel Feige. ACM, 2007, pp. 1–10. DOI: 10.1145/1250790.1250792. URL: https://doi.org/10.1145/1250790.1250792.

[149] Iftach Haitner, Alon Rosen, and Ronen Shaltiel. "On the (Im)Possibility of Arthur-Merlin Witness Hiding Protocols". In: *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings.* Ed. by Omer Reingold. Vol. 5444. Lecture Notes in Computer Science. Springer, 2009, pp. 220–237. DOI: 10.1007/978-3-642-00457-5_14. URL: https://doi.org/10.1007/978-3-642-00457-5_14.

[150] Ramzi A. Haraty, Hadi Otrok, and Abdul Nasser El-Kassar. "A Comparative Study of Elgamal Based Cryptographic Algorithms". In: *ICEIS 2004, Proceedings of the 6th International Conference on Enterprise Information Systems, Porto, Portugal, April 14-17, 2004.* 2004, pp. 79–84.

[151] Johan Håstad et al. “A Pseudorandom Generator from any One-way Function”. In: *SIAM J. Comput.* 28.4 (1999), pp. 1364–1396. DOI: 10.1137/S0097539793244708. URL: https://doi.org/10.1137/S0097539793244708.

[152] Carmit Hazay et al. “Efficient RSA Key Generation and Threshold Paillier in the Two-Party Setting”. In: *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings.* Ed. by Orr Dunkelman. Vol. 7178. Lecture Notes in Computer Science. Springer, 2012, pp. 313–331. DOI: 10.1007/978-3-642-27954-6_20. URL: https://doi.org/10.1007/978-3-642-27954-6_20.

[153] Kai He et al. “Generic Anonymous Identity-Based Broadcast Encryption with Chosen-Ciphertext Security”. In: *Information Security and Privacy - 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part II.* Ed. by Joseph K. Liu and Ron Steinfeld. Vol. 9723. Lecture Notes in Computer Science. Springer, 2016, pp. 207–222. DOI: 10.1007/978-3-319-40367-0_13. URL: https://doi.org/10.1007/978-3-319-40367-0_13.

[154] Sven Heiberg et al. “Improving the Verifiability of the Estonian Internet Voting Scheme”. In: *Electronic Voting - First International Joint Conference, E-Vote-ID 2016, Bregenz, Austria, October 18-21, 2016, Proceedings.* Ed. by Robert Krimmer et al. Vol. 10141. Lecture Notes in Computer Science. Springer, 2016, pp. 92–107. DOI: 10.1007/978-3-319-52240-1_6. URL: https://doi.org/10.1007/978-3-319-52240-1_6.

[155] Javier Herranz, Fabien Laguillaumie, and Carla Ràfols. “Constant Size Ciphertexts in Threshold Attribute-Based Encryption”. In: *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings.* Ed. by Phong Q. Nguyen and David Pointcheval. Vol. 6056. Lecture Notes in Computer Science. Springer, 2010, pp. 19–34. DOI: 10.1007/978-3-642-13013-7_2. URL: https://doi.org/10.1007/978-3-642-13013-7_2.

[156] Detlef Hühnlein et al. “A Cryptosystem Based on Non-maximal Imaginary Quadratic Orders with Fast Decryption”. In: *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding.* Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Springer, 1998, pp. 294–307. DOI: 10.1007/BFb0054134. URL: https://doi.org/10.1007/BFb0054134.

[157] Donald E. Eastlake III and Paul E. Jones. “US Secure Hash Algorithm 1 (SHA1)”. In: *RFC* 3174 (2001), pp. 1–22. DOI: 10.17487/RFC3174. URL: https://doi.org/10.17487/RFC3174.

[158] Russell Impagliazzo and Moti Yung. “Direct Minimum-Knowledge Computations”. In: *A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology.* CRYPTO '87. Berlin, Heidelberg: Springer-Verlag, 1987, 40–51. ISBN: 3540187960.

[159] Vincenzo Iovino and Karol Żebrowski. “Simulation-Based Secure Functional Encryption in the Random Oracle Model”. In: *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings.* 2015, pp. 21–39.

[160] Vincenzo Iovino et al. “Using Selene to Verify Your Vote in JCJ”. In: *Financial Cryptography and Data Security - FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers.* Ed. by Michael Brenner et al. Vol. 10323. Lecture Notes in Computer Science. Springer, 2017, pp. 385–403. DOI: 10.1007/978-3-319-70278-0_24. URL: https://doi.org/10.1007/978-3-319-70278-0_24.

[161] Yuval Ishai et al. “Zero-knowledge from secure multiparty computation”. In: *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*. Ed. by David S. Johnson and Uriel Feige. ACM, 2007, pp. 21–30. DOI: 10.1145/1250790.1250794. URL: https://doi.org/10.1145/1250790.1250794.

[162] Markus Jakobsson and Ari Juels. “Mix and Match: Secure Function Evaluation via Ciphertexts”. In: *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*. Ed. by Tatsuaki Okamoto. Vol. 1976. Lecture Notes in Computer Science. Springer, 2000, pp. 162–177. DOI: 10.1007/3-540-44448-3_13. URL: https://doi.org/10.1007/3-540-44448-3_13.

[163] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. “Designated Verifier Proofs and Their Applications”. In: *Advances in Cryptology - EUROCRYPT ’96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*. Ed. by Ueli M. Maurer. Vol. 1070. Lecture Notes in Computer Science. Springer, 1996, pp. 143–154. DOI: 10.1007/3-540-68339-9_13. URL: https://doi.org/10.1007/3-540-68339-9_13.

[164] Wojciech Jamroga et al. “Risk-Limiting Tallies”. In: *CoRR* abs/1908.04947 (2019). arXiv: 1908.04947. URL: http://arxiv.org/abs/1908.04947.

[165] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. “Zero-Knowledge Using Garbled Circuits: How To Prove Non-Algebraic Statements Efficiently”. In: *IACR Cryptol. ePrint Arch.* (2013), p. 73. URL: http://eprint.iacr.org/2013/073.

[166] Antoine Joux. “A One Round Protocol for Tripartite Diffie-Hellman”. In: *Algorithmic Number Theory, 4th International Symposium, ANTS-IV, Leiden, The Netherlands, July 2-7, 2000, Proceedings*. Ed. by Wieb Bosma. Vol. 1838. Lecture Notes in Computer Science. Springer, 2000, pp. 385–394. DOI: 10.1007/10722028_23. URL: https://doi.org/10.1007/10722028_23.

[167] Antoine Joux. “A One Round Protocol for Tripartite Diffie-Hellman”. In: *J. Cryptol.* 17.4 (2004), pp. 263–276. DOI: 10.1007/s00145-004-0312-y. URL: https://doi.org/10.1007/s00145-004-0312-y.

[168] Ari Juels, Dario Catalano, and Markus Jakobsson. “Coercion-Resistant Electronic Elections”. In: *Towards Trustworthy Elections, New Directions in Electronic Voting*. Ed. by David Chaum et al. Vol. 6000. Lecture Notes in Computer Science. Springer, 2010, pp. 37–63. ISBN: 978-3-642-12979-7. DOI: 10.1007/978-3-642-12980-3_2. URL: http://dx.doi.org/10.1007/978-3-642-12980-3_2.

[169] Tim van de Kamp et al. “Two-Client and Multi-client Functional Encryption for Set Intersection”. In: *Information Security and Privacy - 24th Australasian Conference, ACISP 2019, Christchurch, New Zealand, July 3-5, 2019, Proceedings*. Ed. by Julian Jang-Jaccard and Fuchun Guo. Vol. 11547. Lecture Notes in Computer Science. Springer, 2019, pp. 97–115. DOI: 10.1007/978-3-030-21548-4_6. URL: https://doi.org/10.1007/978-3-030-21548-4_6.

[170] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. “Constant-Size Commitments to Polynomials and Their Applications”. In: *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*. Ed. by Masayuki Abe. Vol. 6477. Lecture Notes in Computer Science. Springer, 2010, pp. 177–194. DOI: 10.1007/978-3-642-17373-8_11. URL: https://doi.org/10.1007/978-3-642-17373-8_11.

[171] Shuichi Katsumata and Shota Yamada. “Non-zero Inner Product Encryption Schemes from Various Assumptions: LWE, DDH and DCR”. In: *Public-Key Cryptography - PKC 2019 - 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part II*. Ed. by Dongdai Lin and Kazue Sako. Vol. 11443. Lecture Notes in Computer Science. Springer, 2019, pp. 158–188. DOI: 10.1007/978-3-030-17259-6_6. URL: https://doi.org/10.1007/978-3-030-17259-6_6.

[172] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014. ISBN: 9781466570269. URL: https://www.crcpress.com/Introduction-to-Modern-Cryptography-Second-Edition/Katz-Lindell/p/book/9781466570269.

[173] Jonathan Katz, Amit Sahai, and Brent Waters. “Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products”. In: *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*. Ed. by Nigel P. Smart. Vol. 4965. Lecture Notes in Computer Science. Springer, 2008, pp. 146–162. DOI: 10.1007/978-3-540-78967-3_9. URL: https://doi.org/10.1007/978-3-540-78967-3_9.

[174] Neal Koblitz. “Elliptic Curve Cryptosystems”. In: *Mathematics of Computation* 48.177 (Jan. 1987), pp. 203–209. ISSN: 0025-5718.

[175] Neal Koblitz. “Elliptic Curve Implementations of Zero-Knowledge Blobs”. In: *J. Cryptol.* 4.3 (1991), pp. 207–213. DOI: 10.1007/BF00196728. URL: https://doi.org/10.1007/BF00196728.

[176] Tadayoshi Kohno et al. “Analysis of an Electronic Voting System”. In: *2004 IEEE Symposium on Security and Privacy (S&P 2004), 9-12 May 2004, Berkeley, CA, USA*. IEEE Computer Society, 2004, p. 27. DOI: 10.1109/SECPRI.2004.1301313. URL: https://doi.org/10.1109/SECPRI.2004.1301313.

[177] Steve Kremer and Mark Ryan. “Analysis of an Electronic Voting Protocol in the Applied Pi Calculus”. In: *Programming Languages and Systems, 14th European Symposium on Programming,ESOP 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings*. Ed. by Shmuel Sagiv. Vol. 3444. Lecture Notes in Computer Science. Springer, 2005, pp. 186–200. DOI: 10.1007/978-3-540-31987-0_14. URL: https://doi.org/10.1007/978-3-540-31987-0_14.

[178] Caroline Kudla and Kenneth G. Paterson. “Non-interactive Designated Verifier Proofs and Undeniable Signatures”. In: *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*. Ed. by Nigel P. Smart. Vol. 3796. Lecture Notes in Computer Science. Springer, 2005, pp. 136–154. DOI: 10.1007/11586821_10. URL: https://doi.org/10.1007/11586821_10.

[179] Oksana Kulyk and Stephan Neumann. “Human Factors in Coercion Resistant Internet Voting - A Review of Existing Solutions and Open Challenges”. In: *Electronic Voting - 5th International Joint Conference, E-Vote-ID 2020*. 2020.

[180] Oksana Kulyk, Vanessa Teague, and Melanie Volkamer. “Extending Helios Towards Private Eligibility Verifiability”. In: *E-Voting and Identity - 5th International Conference, VoteID 2015, Bern, Switzerland, September 2-4, 2015, Proceedings*. Ed. by Rolf Haenni, Reto E. Koenig, and Douglas Wikström. Vol. 9269. Lecture Notes in Computer Science. Springer, 2015, pp. 57–73. DOI: 10.1007/978-3-319-22270-7_4. URL: https://doi.org/10.1007/978-3-319-22270-7_4.

[181] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. "A game-based definition of coercion resistance and its applications". In: *J. Comput. Secur.* 20.6 (2012), pp. 709–764. DOI: 10.3233/JCS-2012-0444.

[182] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. "Accountability: Definition and Relationship to Verifiability". In: *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010.* 2010, pp. 526–535.

[183] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. "Verifiability, privacy, and coercion-resistance: New insights from a case study". In: *2011 IEEE Symposium on Security and Privacy*. IEEE. 2011, pp. 538–553.

[184] Ralf Küsters et al. "Ordinos: A Verifiable Tally-Hiding E-Voting System". In: *IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020.* IEEE, 2020, pp. 216–235. DOI: 10.1109/EuroSP48549.2020.00022. URL: https://doi.org/10.1109/EuroSP48549.2020.00022.

[185] Benjamin Kuykendall and Mark Zhandry. "Towards Non-interactive Witness Hiding". In: *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part I.* Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12550. Lecture Notes in Computer Science. Springer, 2020, pp. 627–656. DOI: 10.1007/978-3-030-64375-1_22. URL: https://doi.org/10.1007/978-3-030-64375-1_22.

[186] Junzuo Lai et al. "Fully secure key-policy attribute-based encryption with constant-size ciphertexts and fast decryption". In: *9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14, Kyoto, Japan - June 03 - 06, 2014.* Ed. by Shiho Moriai, Trent Jaeger, and Kouichi Sakurai. ACM, 2014, pp. 239–248. DOI: 10.1145/2590296.2590334. URL: https://doi.org/10.1145/2590296.2590334.

[187] Lucie Langer et al. "A Taxonomy Refining the Security Requirements for Electronic Voting: Analyzing Helios as a Proof of Concept". In: *ARES 2010, Fifth International Conference on Availability, Reliability and Security, 15-18 February 2010, Krakow, Poland.* IEEE Computer Society, 2010, pp. 475–480. DOI: 10.1109/ARES.2010.106. URL: https://doi.org/10.1109/ARES.2010.106.

[188] Leonid A. Levin. "Average Case Complete Problems". In: *SIAM J. Comput.* 15.1 (1986), pp. 285–286. DOI: 10.1137/0215020. URL: https://doi.org/10.1137/0215020.

[189] Allison B. Lewko et al. "Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption". In: *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings.* Ed. by Henri Gilbert. Vol. 6110. Lecture Notes in Computer Science. Springer, 2010, pp. 62–91. DOI: 10.1007/978-3-642-13190-5_4. URL: https://doi.org/10.1007/978-3-642-13190-5_4.

[190] Jin Li et al. "Multi-authority ciphertext-policy attribute-based encryption with accountability". In: *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2011, Hong Kong, China, March 22-24, 2011.* Ed. by Bruce S. N. Cheung et al. ACM, 2011, pp. 386–390. DOI: 10.1145/1966913.1966964. URL: https://doi.org/10.1145/1966913.1966964.

[191] Yehuda Lindell. "How to Simulate It - A Tutorial on the Simulation Proof Technique". In: *Tutorials on the Foundations of Cryptography.* Ed. by Yehuda Lindell. Springer International Publishing, 2017, pp. 277–346. DOI: 10.1007/978-3-319-57048-8_6. URL: https://doi.org/10.1007/978-3-319-57048-8_6.

[192] H. Lipmaa and T. Toft. “Secure Equality and Greater-Than Tests with Sublinear Online Complexity”. In: *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, 2013, Proceedings*. Ed. by F. Fomin et al. Vol. 7966. Lecture Notes in Computer Science. Springer, 2013, pp. 645–656. DOI: 10.1007/978-3-642-39212-2_56. URL: https://doi.org/10.1007/978-3-642-39212-2_56.

[193] Helger Lipmaa. “On Diophantine Complexity and Statistical Zero-Knowledge Arguments”. In: *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*. Ed. by Chi-Sung Laih. Vol. 2894. Lecture Notes in Computer Science. Springer, 2003, pp. 398–415. DOI: 10.1007/978-3-540-40061-5_26. URL: https://doi.org/10.1007/978-3-540-40061-5_26.

[194] Wenbo Liu et al. “Efficient functional encryption for inner product with simulation-based security”. In: *Cybersecur.* 4.1 (2021), p. 2. DOI: 10.1186/s42400-020-00067-1. URL: https://doi.org/10.1186/s42400-020-00067-1.

[195] Zhen Liu et al. “Fully Secure Multi-authority Ciphertext-Policy Attribute-Based Encryption without Random Oracles”. In: *Computer Security - ESORICS 2011 - 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings*. Ed. by Vijay Atluri and Claudia Díaz. Vol. 6879. Lecture Notes in Computer Science. Springer, 2011, pp. 278–297. DOI: 10.1007/978-3-642-23822-2_16. URL: https://doi.org/10.1007/978-3-642-23822-2_16.

[196] Philipp Locher, Rolf Haenni, and Reto E. Koenig. “Coercion-Resistant Internet Voting with Everlasting Privacy”. In: *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*. Ed. by Jeremy Clark et al. Vol. 9604. Lecture Notes in Computer Science. Springer, 2016, pp. 161–175. DOI: 10.1007/978-3-662-53357-4_11. URL: https://doi.org/10.1007/978-3-662-53357-4_11.

[197] Wouter Lueks, Iñigo Querejeta-Azurmendi, and Carmela Troncoso. “VoteAgain: A Scalable Coercion-Resistant Voting System”. In: *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*. Ed. by Srdjan Capkun and Franziska Roesner. USENIX Association, 2020, pp. 1553–1570. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/lueks.

[198] Carsten Lund et al. “Algebraic Methods for Interactive Proof Systems”. In: *J. ACM* 39.4 (Oct. 1992), 859–868. ISSN: 0004-5411. DOI: 10.1145/146585.146605. URL: https://doi.org/10.1145/146585.146605.

[199] Carla Mascia, Massimiliano Sala, and Irene Villa. “A survey on Functional Encryption”. In: *CoRR* abs/2106.06306 (2021). arXiv: 2106.06306. URL: https://arxiv.org/abs/2106.06306.

[200] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001. URL: http://www.cacr.math.uwaterloo.ca/hac/.

[201] Victor S. Miller. “Use of Elliptic Curves in Cryptography”. In: *Advances in Cryptology - CRYPTO ’85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*. Ed. by Hugh C. Williams. Vol. 218. Lecture Notes in Computer Science. Springer, 1985, pp. 417–426. DOI: 10.1007/3-540-39799-X_31. URL: https://doi.org/10.1007/3-540-39799-X_31.

[202] Eduardo Morais et al. “A Survey on Zero Knowledge Range Proofs and Applications”. In: *CoRR* abs/1907.06381 (2019). arXiv: 1907.06381. URL: http://arxiv.org/abs/1907.06381.

[203] Moni Naor. “Bit Commitment Using Pseudorandomness”. In: *J. Cryptol.* 4.2 (1991), pp. 151–158. DOI: 10.1007/BF00196774. URL: https://doi.org/10.1007/BF00196774.

[204] Moni Naor and Omer Reingold. “On the Construction of Pseudorandom Permutations: Luby-Rackoff Revisited”. In: *J. Cryptol.* 12.1 (1999), pp. 29–66. DOI: 10.1007/PL00003817. URL: https://doi.org/10.1007/PL00003817.

[205] A. Silva Neto et al. “Usability Considerations For Coercion-Resistant Election Systems”. In: *Proceedings of the 17th Brazilian Symposium on Human Factors in Computing Systems, IHC 2018, Brazil, 2018.* Ed. by M. Mota et al. ACM, 2018, 40:1–40:10. ISBN: 978-1-4503-6601-4. DOI: 10.1145/3274192.3274232. URL: https://doi.org/10.1145/3274192.3274232.

[206] S. Neumann and M. Volkamer. “Civitas and the Real World: Problems and Solutions from a Practical Point of View”. In: *Seventh International Conference on Availability, Reliability and Security, Prague, ARES 2012, Czech Republic, August 20-24, 2012.* IEEE Computer Society, 2012, pp. 180–185. DOI: 10.1109/ARES.2012.75. URL: https://doi.org/10.1109/ARES.2012.75.

[207] S. Neumann et al. “Towards a practical jcj/civitas implementation”. In: *INFORMATIK 2013–Informatik angepasst an Mensch, Organisation und Umwelt* (2013).

[208] Stephan Neumann. “Evaluation and improvement of internet voting schemes based on legally-founded security requirements”. PhD thesis. Darmstadt University of Technology, Germany, 2016. URL: https://d-nb.info/1105390284.

[209] Minh-Huyen Nguyen, Shien Jin Ong, and Salil P. Vadhan. “Statistical Zero-Knowledge Arguments for NP from Any One-Way Function”. In: *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings.* IEEE Computer Society, 2006, pp. 3–14. DOI: 10.1109/FOCS.2006.71. URL: https://doi.org/10.1109/FOCS.2006.71.

[210] Takashi Nishide and Kouichi Sakurai. “Distributed Paillier Cryptosystem without Trusted Dealer”. In: *Information Security Applications - 11th International Workshop, WISA 2010, Jeju Island, Korea, August 24-26, 2010, Revised Selected Papers.* Ed. by Yongwha Chung and Moti Yung. Vol. 6513. Lecture Notes in Computer Science. Springer, 2010, pp. 44–60. DOI: 10.1007/978-3-642-17955-6_4. URL: https://doi.org/10.1007/978-3-642-17955-6_4.

[211] Tatsuaki Okamoto. “Receipt-Free Electronic Voting Schemes for Large Scale Elections”. In: *Security Protocols, 5th International Workshop, Paris, France, April 7-9, 1997, Proceedings.* Ed. by Bruce Christianson et al. Vol. 1361. Lecture Notes in Computer Science. Springer, 1997, pp. 25–35. DOI: 10.1007/BFb0028157. URL: https://doi.org/10.1007/BFb0028157.

[212] Tatsuaki Okamoto and Katsuyuki Takashima. “Adaptively Attribute-Hiding (Hierarchical) Inner Product Encryption”. In: *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* 99-A.1 (2016), pp. 92–117. DOI: 10.1587/transfun.E99.A.92. URL: https://doi.org/10.1587/transfun.E99.A.92.

[213] Tatsuaki Okamoto and Katsuyuki Takashima. “Hierarchical Predicate Encryption for Inner-Products”. In: *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings.* Ed. by Mitsuru Matsui. Vol. 5912. Lecture Notes in Computer Science. Springer, 2009, pp. 214–231. DOI: 10.1007/978-3-642-10366-7_13. URL: https://doi.org/10.1007/978-3-642-10366-7_13.

[214] Adam O’Neill. “Definitional Issues in Functional Encryption”. In: *IACR Cryptology ePrint Archive* 2010 (2010), p. 556. URL: http://eprint.iacr.org/2010/556.

[215] R. Ostrovsky and A. Wigderson. “One-way functions are essential for non-trivial zero-knowledge”. In: *[1993] The 2nd Israel Symposium on Theory and Computing Systems*. 1993, pp. 3–17. DOI: 10.1109/ISTCS.1993.253489.

[216] Pascal Paillier. “Public-Key Cryptosystems Based on Composite Degree Residuosity Classes”. In: *Advances in Cryptology - EUROCRYPT ’99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*. Ed. by Jacques Stern. Vol. 1592. Lecture Notes in Computer Science. Springer, 1999, pp. 223–238. DOI: 10.1007/3-540-48910-X_16. URL: https://doi.org/10.1007/3-540-48910-X_16.

[217] Jong Hwan Park. “Inner-product encryption under standard assumptions”. In: *Des. Codes Cryptography* 58.3 (2011), pp. 235–257.

[218] Rafael Pass and Alon Rosen. “Concurrent Non-Malleable Commitments”. In: *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings*. IEEE Computer Society, 2005, pp. 563–572. DOI: 10.1109/SFCS.2005.27. URL: https://doi.org/10.1109/SFCS.2005.27.

[219] Torben P. Pedersen. “Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing”. In: *CRYPTO 1991*. 1991, pp. 129–140.

[220] *Real-World Electronic Voting: Design, Analysis and Deployment*. en. URL: https://www.routledge.com/Real-World-Electronic-Voting-Design-Analysis-and-Deployment/Hao-Ryan/p/book/9780367658212 (visited on 2022-01-22).

[221] *Real-World Electronic Voting: Design, Analysis and Deployment*. en. URL: https://www.routledge.com/Real-World-Electronic-Voting-Design-Analysis-and-Deployment/Hao-Ryan/p/book/9780367658212 (visited on 2022-01-22).

[222] R. L. Rivest, A. Shamir, and L. Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. In: *Commun. ACM* 21.2 (1978), 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342. URL: https://doi.org/10.1145/359340.359342.

[223] Peter B. Roenne. “JCJ with Improved Verifiability Guarantees”. In: *The International Conference on Electronic Voting E-Vote-ID 2016*. 2016.

[224] P. B Rønne et al. “Coercion-Resistant Voting in Linear Time via Fully Homomorphic Encryption: Towards a Quantum-Safe Scheme”. In: *arXiv preprint arXiv:1901.02560* (2019).

[225] Peter Y A Ryan, Peter B Rønne, and Vincenzo Iovino. “Selene: Voting with transparent verifiability and coercion-mitigation”. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2016, pp. 176–192.

[226] Peter Y. A. Ryan et al. “Prêt à voter: a voter-verifiable voting system”. In: *IEEE Trans. Inf. Forensics Secur.* 4.4 (2009), pp. 662–673. DOI: 10.1109/TIFS.2009.2033233. URL: https://doi.org/10.1109/TIFS.2009.2033233.

[227] Peter Y. A. Ryan et al. “Who Was that Masked Voter? The Tally Won’t Tell!” In: *Electronic Voting - 6th International Joint Conference, E-Vote-ID 2021, Virtual Event, October 5-8, 2021, Proceedings*. Ed. by Robert Krimmer et al. Vol. 12900. Lecture Notes in Computer Science. Springer, 2021, pp. 106–123. DOI: 10.1007/978-3-030-86942-7_8. URL: https://doi.org/10.1007/978-3-030-86942-7_8.

[228] Amit Sahai and Hakan Seyalioglu. “Worry-free encryption: functional encryption with public keys”. In: *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*. Ed. by Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov. ACM, 2010, pp. 463–472. DOI: 10.1145/1866307.1866359. URL: https://doi.org/10.1145/1866307.1866359.

[229] Amit Sahai and Brent Waters. “Fuzzy Identity-Based Encryption”. In: *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings.* Ed. by Ronald Cramer. Vol. 3494. Lecture Notes in Computer Science. Springer, 2005, pp. 457–473. DOI: 10.1007/11426639_27. URL: https://doi.org/10.1007/11426639_27.

[230] Kazue Sako and Joe Kilian. “Receipt-Free Mix-Type Voting Scheme - A Practical Solution to the Implementation of a Voting Booth”. In: *Advances in Cryptology - EUROCRYPT ’95, International Conference on the Theory and Application of Cryptographic Techniques, Saint-Malo, France, May 21-25, 1995, Proceeding.* Ed. by Louis C. Guillou and Jean-Jacques Quisquater. Vol. 921. Lecture Notes in Computer Science. Springer, 1995, pp. 393–403. DOI: 10.1007/3-540-49264-X_32. URL: https://doi.org/10.1007/3-540-49264-X_32.

[231] Edouard Dufour Sans and David Pointcheval. “Unbounded Inner-Product Functional Encryption with Succinct Keys”. In: *Applied Cryptography and Network Security - 17th International Conference, ACNS 2019, Bogota, Colombia, June 5-7, 2019, Proceedings.* Ed. by Robert H. Deng et al. Vol. 11464. Lecture Notes in Computer Science. Springer, 2019, pp. 426–441. DOI: 10.1007/978-3-030-21568-2_21. URL: https://doi.org/10.1007/978-3-030-21568-2_21.

[232] Michael Schläpfer et al. “Efficient Vote Authorization in Coercion-Resistant Internet Voting”. In: *E-Voting and Identity - Third International Conference, VoteID 2011, Tallinn, Estonia, September 28-30, 2011, Revised Selected Papers.* Ed. by Aggelos Kiayias and Helger Lipmaa. Vol. 7187. Lecture Notes in Computer Science. Springer, 2011, pp. 71–88. DOI: 10.1007/978-3-642-32747-6_5. URL: https://doi.org/10.1007/978-3-642-32747-6_5.

[233] Claus-Peter Schnorr. “Efficient Signature Generation by Smart Cards”. In: *J. Cryptol.* 4.3 (1991), pp. 161–174. DOI: 10.1007/BF00196725. URL: https://doi.org/10.1007/BF00196725.

[234] Adi Shamir. “Identity-Based Cryptosystems and Signature Schemes”. In: *Advances in Cryptology, Proceedings of CRYPTO ’84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings.* Ed. by G. R. Blakley and David Chaum. Vol. 196. Lecture Notes in Computer Science. Springer, 1984, pp. 47–53. DOI: 10.1007/3-540-39568-7_5. URL: https://doi.org/10.1007/3-540-39568-7_5.

[235] Adi Shamir. “IP=PSPACE”. In: *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I.* IEEE Computer Society, 1990, pp. 11–15. DOI: 10.1109/FSCS.1990.89519. URL: https://doi.org/10.1109/FSCS.1990.89519.

[236] Adi Shamir and Nicko Van Someren. “Playing ‘hide and seek’with stored keys”. In: *International conference on financial cryptography.* Springer. 1999, pp. 118–124.

[237] Claude E. Shannon. “Communication theory of secrecy systems”. In: *Bell Syst. Tech. J.* 28.4 (1949), pp. 656–715. DOI: 10.1002/j.1538-7305.1949.tb00928.x. URL: https://doi.org/10.1002/j.1538-7305.1949.tb00928.x.

[238] Victor Shoup. “OAEP Reconsidered”. In: *J. Cryptol.* 15.4 (2002), pp. 223–249. DOI: 10.1007/s00145-002-0133-9. URL: https://doi.org/10.1007/s00145-002-0133-9.

[239] Joseph H. Silverman. *The arithmetic of elliptic curves.* Vol. 106. Graduate texts in mathematics. Springer, 1986. ISBN: 978-3-540-96203-8.

[240] Ben Smyth, Steven Frink, and Michael R. Clarkson. "Computational Election Verifiability: Definitions and an Analysis of Helios and JCJ". In: *IACR Cryptol. ePrint Arch.* (2015), p. 233. URL: http://eprint.iacr.org/2015/233.

[241] Ben Smyth, Steven Frink, and Michael R. Clarkson. "Election Verifiability: Cryptographic Definitions and an Analysis of Helios and JCJ". In: 2015.

[242] Najmeh Soroush et al. "Verifiable Inner Product Encryption Scheme". In: *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part I.* Ed. by Aggelos Kiayias et al. Vol. 12110. Lecture Notes in Computer Science. Springer, 2020, pp. 65–94. DOI: 10.1007/978-3-030-45374-9_3. URL: https://doi.org/10.1007/978-3-030-45374-9_3.

[243] Michael A. Specter, James Koppel, and Daniel J. Weitzner. "The Ballot is Busted Before the Blockchain: A Security Analysis of Voatz, the First Internet Voting Application Used in U.S. Federal Elections". In: *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020.* Ed. by Srdjan Capkun and Franziska Roesner. USENIX Association, 2020, pp. 1535–1553. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/specter.

[244] Drew Springall et al. "Security Analysis of the Estonian Internet Voting System". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014.* Ed. by Gail-Joon Ahn, Moti Yung, and Ninghui Li. ACM, 2014, pp. 703–715. DOI: 10.1145/2660267.2660315. URL: https://doi.org/10.1145/2660267.2660315.

[245] O. Spycher, R. Koenig R.and Haenni, and M. Schläpfer. "A new approach towards coercion-resistant remote e-voting in linear time". In: *International Conference on Financial Cryptography and Data Security.* Springer. 2011, pp. 182–189.

[246] Björn Terelius and Douglas Wikström. "Proofs of Restricted Shuffles". In: *Progress in Cryptology - AFRICACRYPT 2010, Third International Conference on Cryptology in Africa.* Ed. by Daniel J. Bernstein and Tanja Lange. Vol. 6055. Lecture Notes in Computer Science. Springer, 2010, pp. 100–113.

[247] Yiannis Tsiounis and Moti Yung. "On the Security of ElGamal Based Encryption". In: *Public Key Cryptography, First International Workshop on Practice and Theory in Public Key Cryptography, PKC '98, Pacifico Yokohama, Japan, February 5-6, 1998, Proceedings.* Ed. by Hideki Imai and Yuliang Zheng. Vol. 1431. Lecture Notes in Computer Science. Springer, 1998, pp. 117–134. DOI: 10.1007/BFb0054019. URL: https://doi.org/10.1007/BFb0054019.

[248] *UN Committee on Human Rights.* General Comment 25 of the Human Rights Committee. URL: https://www.ohchr.org/EN/Issues/Pages/HRElections.aspx.

[249] United Nations. *Universal Declaration of Human Rights.* Dec. 1948.

[250] Salil Pravin Vadhan and Shafi Goldwasser. "A Study of Statistical Zero-Knowledge Proofs". AAI0801528. PhD thesis. USA, 1999.

[251] Frederik Vercauteren. "Optimal pairings". In: *IEEE Trans. Inf. Theory* 56.1 (2010), pp. 455–461. DOI: 10.1109/TIT.2009.2034881. URL: https://doi.org/10.1109/TIT.2009.2034881.

[252] G. S. Vernam. "Cipher Printing Telegraph Systems For Secret Wire and Radio Telegraphic Communications". In: *Transactions of the American Institute of Electrical Engineers* XLV (), pp. 295–301.

[253] Giuseppe Vitto. *Factoring Primes to Factor Moduli: Backdooring and Distributed Generation of Semiprimes*. Cryptology ePrint Archive, Report 2021/1610. https://ia.cr/2021/1610. 2021.

[254] Melanie Volkamer. *Evaluation of Electronic Voting - Requirements and Evaluation Procedures to Support Responsible Election Authorities*. Vol. 30. Lecture Notes in Business Information Processing. Springer, 2009. ISBN: 978-3-642-01661-5. DOI: 10.1007/978-3-642-01662-2. URL: https://doi.org/10.1007/978-3-642-01662-2.

[255] Guojun Wang, Qin Liu, and Jie Wu. "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services". In: *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*. Ed. by Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov. ACM, 2010, pp. 735–737. DOI: 10.1145/1866307.1866414. URL: https://doi.org/10.1145/1866307.1866414.

[256] Brent Waters. "Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization". In: *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*. Ed. by Dario Catalano et al. Vol. 6571. Lecture Notes in Computer Science. Springer, 2011, pp. 53–70. DOI: 10.1007/978-3-642-19379-8_4. URL: https://doi.org/10.1007/978-3-642-19379-8_4.

[257] Brent Waters. "Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions". In: *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*. Ed. by Shai Halevi. Vol. 5677. Lecture Notes in Computer Science. Springer, 2009, pp. 619–636. DOI: 10.1007/978-3-642-03356-8_36. URL: https://doi.org/10.1007/978-3-642-03356-8_36.

[258] André Weil. "Sur les fonctions algébriques à corps de constantes fini". In: 1979.

[259] Roland Wen and Richard Buckland. "Masked Ballot Voting for Receipt-Free Online Elections". In: *E-Voting and Identity, Second International Conference, VoteID 2009, Luxembourg, September 7-8, 2009. Proceedings*. Ed. by Peter Y. A. Ryan and Berry Schoenmakers. Vol. 5767. Lecture Notes in Computer Science. Springer, 2009, pp. 18–36. DOI: 10.1007/978-3-642-04135-8_2. URL: https://doi.org/10.1007/978-3-642-04135-8_2.

[260] S. Wiseman, P. Cairns, and A. Cox. "A taxonomy of number entry error". In: *Proceedings of the 25th BCS Conference on Human-Computer Interaction*. British Computer Society. 2011, pp. 187–196.

[261] Scott Wolchok et al. "Security analysis of India's electronic voting machines". In: *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*. Ed. by Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov. ACM, 2010, pp. 1–14. DOI: 10.1145/1866307.1866309. URL: https://doi.org/10.1145/1866307.1866309.

[262] Peng Xu et al. "Anonymous Identity-Based Broadcast Encryption with Constant Decryption Complexity and Strong Security". In: *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2016, Xi'an, China, May 30 - June 3, 2016*. Ed. by Xiaofeng Chen, XiaoFeng Wang, and Xinyi Huang. ACM, 2016, pp. 223–233. DOI: 10.1145/2897845.2897853. URL: https://doi.org/10.1145/2897845.2897853.

[263] Jiang Zhang, Zhenfeng Zhang, and Aijun Ge. “Ciphertext policy attribute-based encryption from lattices”. In: *7th ACM Symposium on Information, Compuer and Communications Security, ASIACCS ’12, Seoul, Korea, May 2-4, 2012*. Ed. by Heung Youl Youm and Yoojae Won. ACM, 2012, pp. 16–17. DOI: `10.1145/2414456.2414464`. URL: `https://doi.org/10.1145/2414456.2414464`.

[264] Shiwei Zhang, Yi Mu, and Guomin Yang. “Achieving IND-CCA Security for Functional Encryption for Inner Products”. In: *Information Security and Cryptology - 12th International Conference, Inscrypt 2016, Beijing, China, November 4-6, 2016, Revised Selected Papers*. Ed. by Kefei Chen, Dongdai Lin, and Moti Yung. Vol. 10143. Lecture Notes in Computer Science. Springer, 2016, pp. 119–139. DOI: `10.1007/978-3-319-54705-3\_8`. URL: `https://doi.org/10.1007/978-3-319-54705-3\_8`.

Appendix

Appendix .A

Proof of Proposition 3

Prposition 3. If the DBDH assumption holds relative to $\mathsf{GroupGen}$, then H_1 and H_2 are computationally indistinguishable.

Proof. The simulator $\mathbb{B}$ takes as input $(g, A = g^{\alpha}, B = g^{\beta}, C = g^{\tau}, Z \overset{?}{=} \mathbf{e}(g, g^{\alpha\beta\tau}))$ and interacts with the adversary $\mathcal{A}$ impersonating the challenger.

SetUp phase. The adversary $\mathcal{A}$ sends two vectors $\overrightarrow{x}, \overrightarrow{y}$ to $\mathbb{B}$. The simulator picks

$$\Omega, k, \tilde{a}, \delta_b, \theta_b, w_{1,i}, t_{1,i}, \tilde{f}_{b,i}, \tilde{h}_{b,i} \overset{\$}{\leftarrow} \mathbb{Z}_p^{\star}$$

for $b = 1,2$ and $i \in [n]$. Then, for each $i \in [n]$, the simulator computes $w_{2,i}, t_{2,i}$ such that:

$$\Omega = \delta_1 w_{2,i} - \delta_2 w_{1,i} = \theta_1 t_{2,i} - \theta_2 t_{1,i}.$$

$\mathbb{B}$ computes the master public key components for $b \in [2], i \in [n]$ as follows:

$$\{W_{b,i} = g^{w_{b,i}}, F_{b,i} = B^{x_i \delta_b} \cdot g^{\tilde{f}_{b,i}}, T_{b,i} = g^{t_{b,i}}, H_{b,i} = B^{x_i \theta_b} \cdot g^{\tilde{h}_{b,i}}\}_{b \in [2], i \in [n]}$$
$$\{U_b = g^{\delta_b}, V_b = g^{\theta_b}\}_{b \in [2]}, h = g^{\Omega}, \Lambda = \mathbf{e}(A,B)^{-\Omega} \cdot \mathbf{e}(A,g)^{\tilde{a}}, K_1 = A^k,\ K_2 = B^{\frac{-\Omega}{k}} \cdot g^{\frac{\tilde{a}}{k}}.$$

By doing so, $\mathbb{B}$ knows all secret parameters except $\{f_{b,i}, h_{b,i}\}_{b \in [2], i \in [n]}$ which implicitly are set $f_{b,i} = x_i \delta_b \beta + \tilde{f}_{b,i}$, $h_{b,i} = x_i \theta_b \beta + \tilde{h}_{b,i}$. The following shows the simulator generates the well-form master public key, with same distribution in both hybrids:

$$\Lambda = \mathbf{e}(A,B)^{-\Omega} \cdot \mathbf{e}(A,g)^{\tilde{a}} = \mathbf{e}(g^{\alpha}, g^{\beta})^{-\Omega} \cdot \mathbf{e}(g^{\alpha}, g)^{\tilde{a}} = \mathbf{e}(g, g^{-\alpha\beta\Omega + \alpha\tilde{a}}) \Rightarrow g' = g^{-\alpha\beta\Omega + \alpha\tilde{a}},$$
$$\mathbf{e}(K_1, K_2) = \mathbf{e}(A^k, B^{\frac{-\Omega}{k}} \cdot g^{\frac{\tilde{a}}{k}}) = \mathbf{e}(A, B^{-\Omega} \cdot g^{\tilde{a}}) = \Lambda.$$

Token query phase. First, notice that in the token query phase, $\mathcal{A}$ is allowed to ask the token for some vectors $\overrightarrow{v} \in \mathbb{Z}_p^n$ such that $c_x = \langle \vec{x}, \vec{v} \rangle \neq 0$. To generate a token for vector $\overrightarrow{v}$, the simulator first chooses random elements $\tilde{\lambda}_1, \tilde{\lambda}_2, r_i, \Phi_i \overset{\$}{\leftarrow} \mathbb{Z}_p$, for $i = 1, \ldots, n$ and compute the token components as follows:

$$\{K_{3,i} = g^{-\delta_2 r_i} \cdot (g^{\tilde{\lambda}_1} \cdot A^{-1/2c_x})^{v_i w_{2,i}}, K_{4,i} = g^{\delta_1 r_i - \tilde{\lambda}_1 v_i w_{1,i}} \cdot A^{v_i w_{1,i}/2c_x}\}_{i \in [n]},$$
$$\{K_{5,i} = g^{-\theta_2 \Phi_i} \cdot (g^{\tilde{\lambda}_2} \cdot A^{-1/2c_x})^{v_i t_{2,i}}, K_{6,i} = g^{\theta_1 \Phi_i - \tilde{\lambda}_2 v_i t_{1,i}} \cdot A^{v_i t_{1,i}/2c_x}\}_{i \in [n]},$$
$$K_B = \prod_{i=1}^{n} g^{-(r_i + \Phi_i)}, K_A = \Psi_1 \cdot \Psi_2 \cdot A^{\tilde{a}}$$

which:

$$\Psi_1 = B^{-\tilde{\lambda}_1 \Omega c_x} \cdot \prod_{i=1}^{n} F_{1,i}^{\delta_2 r_i} F_{2,i}^{-\delta_1 r_i} g^{-\lambda_1 v_i (\tilde{f}_{1,i} w_{2,i} - \tilde{f}_{2,i} w_{1,i})}$$
$$\Psi_2 = B^{-\tilde{\lambda}_2 \Omega c_x} \cdot \prod_{i=1}^{n} H_{1,i}^{\theta_2 \Phi_i} H_{2,i}^{-\theta_1 \Phi_i} g^{-\lambda_2 v_i (\tilde{h}_{1,i} t_{2,i} - \tilde{h}_{2,i} t_{1,i})}$$

Note that $g^{\tilde{\lambda}_1} \cdot A^{-1/2c_x} = g^{\tilde{\lambda}_1 - \alpha/2c_x}$ and $g^{\tilde{\lambda}_2} \cdot A^{-1/2c_x} = g^{\tilde{\lambda}_2 - \alpha/2c_x}$, hence by defining $\lambda_b = \tilde{\lambda}_b - \alpha/(2c_x)$, for $b = 1,2$, we see $\{K_{j,i}\}_{j=3,4,5,6, i\in[n]}$ has the proper structure. For component K_A consider the following computation:

$$\begin{aligned}
& f_{1,i} w_{2,i} - f_{2,i} w_{1,i} = (x_i \delta_1 \beta + \tilde{f}_{1,i}) w_{2,i} - (x_i \delta_2 \beta + \tilde{f}_{2,i}) w_{1,i} = \\
& x_i \beta (\delta_1 w_{2,i} - \delta_2 w_{1,i}) + \tilde{f}_{1,i} w_{2,i} - \tilde{f}_{2,i} w_{1,i} = \Omega x_i \beta + (\tilde{f}_{1,i} w_{2,i} - \tilde{f}_{2,i} w_{1,i}) \\
\Rightarrow & K_{3,i}^{-f_{1,i}} K_{4,i}^{-f_{2,i}} = g^{-f_{1,i}(-\delta_2 r_i + v_i w_{2,i} \lambda_1) - f_{2,i}(\delta_1 r_i - v_i w_{1,i} \lambda_1)} = \\
& = F_{1,i}^{\delta_2 r_i} F_{2,i}^{-\delta_1 r_i} g^{\lambda_1 v_i \left(-f_{1,i} w_{2,i} + f_{2,i} w_{1,i}\right)} = F_{1,i}^{\delta_2 r_i} F_{2,i}^{-\delta_1 r_i} g^{-\lambda_1 v_i \left(\Omega x_i \beta + (\tilde{f}_{1,i} w_{2,i} - \tilde{f}_{2,i} w_{1,i})\right)} \\
& = F_{1,i}^{\delta_2 r_i} F_{2,i}^{-\delta_1 r_i} g^{-\lambda_1 v_i (\tilde{f}_{1,i} w_{2,i} - \tilde{f}_{2,i} w_{1,i})} \cdot g^{-\lambda_1 v_i \Omega x_i \beta} \\
\Rightarrow & \prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} K_{4,i}^{-f_{2,i}} = \prod_{i=1}^{n} F_{1,i}^{\delta_2 r_i} F_{2,i}^{-\delta_1 r_i} g^{-\lambda_1 v_i (\tilde{f}_{1,i} w_{2,i} - \tilde{f}_{2,i} w_{1,i})} \cdot g^{-\lambda_1 \Omega \beta \sum_{i=1}^{n} v_i x_i} \\
& = \prod_{i=1}^{n} F_{1,i}^{\delta_2 r_i} F_{2,i}^{-\delta_1 r_i} g^{-\lambda_1 v_i (\tilde{f}_{1,i} w_{2,i} - \tilde{f}_{2,i} w_{1,i})} \cdot g^{-(\tilde{\lambda}_1 - \alpha/(2c_x)) \Omega \beta \langle \vec{x}, \vec{v} \rangle} \\
& = \underbrace{\left(\prod_{i=1}^{n} F_{1,i}^{\delta_2 r_i} F_{2,i}^{-\delta_1 r_i} g^{-\lambda_1 v_i (\tilde{f}_{1,i} w_{2,i} - \tilde{f}_{2,i} w_{1,i})} \right) \cdot B^{-\tilde{\lambda}_1 \Omega c_x}}_{\Psi_1} \cdot g^{\Omega \alpha \beta/2} = \Psi_1 \cdot g^{\Omega \alpha \beta/2}
\end{aligned}$$

With same computation we conclude $\prod_{i=1}^{n} K_{5,i}^{-h_{1,i}} K_{6,i}^{-h_{2,i}} = \Psi_2 \cdot g^{\Omega \alpha \beta/2}$ hence:

$$\begin{aligned}
\Rightarrow K_A = g' \cdot \prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} K_{4,i}^{-f_{2,i}} K_{5,i}^{-h_{1,i}} K_{6,i}^{-h_{2,i}} &= g' \cdot \Psi_1 \cdot g^{\Omega \alpha \beta/2} \cdot \Psi_2 \cdot g^{\Omega \alpha \beta/2} \\
&= \Psi_1 \cdot \Psi_2 \cdot g^{-\alpha \beta \Omega + \alpha \tilde{a}} \cdot g^{\alpha \beta \Omega} = \Psi_1 \cdot \Psi_2 \cdot g^{\alpha \tilde{a}} = \Psi_1 \cdot \Psi_2 \cdot A^{\tilde{a}}
\end{aligned}$$

Challenge phase. The simulator chooses random elements

$$s_1, \tilde{s}_2, \tilde{s}_3, \tilde{s}_4, s'_1, \tilde{s}'_2, \tilde{s}'_3 \xleftarrow{\$} \mathbb{Z}_p^\star \ : \ \tilde{s}_3 \neq \tilde{s}'_3$$

and implicitly define the new randomnesses:

$$s_2 = \tau + \tilde{s}_2, s_2 = \tau + \tilde{s}'_2, s_3 = -\beta\tau + \tilde{s}_3, s_3 = -\beta\tau + \tilde{s}'_3, s_4 = -\beta\tau + \tilde{s}_4$$

The challenge ciphertext is computed as follows (for $i \in [n]$)

$$\begin{aligned}
& \mathrm{ct}_1 = C^{\tilde{s}_2} = g^{\tau \tilde{s}_2}, \mathrm{ct}'_1 = C^{\tilde{s}'_2} = g^{\tau \tilde{s}'_2}, \mathrm{ct}_2 = h^{s_1}, \mathrm{ct}'_2 = h^{s'_1} \\
& \mathrm{ct}_{3,i} = g^{s_1 \tilde{w}_{1,i}} \cdot g^{\tilde{s}_2 \tilde{f}_{1,i}} \cdot B^{\tilde{\delta}_1 x_i \tilde{s}_2} \cdot C^{\tilde{f}_{1,i}} \cdot g^{\tilde{\delta}_1 x_i \tilde{s}_3}, \ \mathrm{ct}'_{3,i} = g^{s'_1 \tilde{w}_{1,i}} \cdot C^{\tilde{s}'_2 \tilde{f}_{1,i}} \cdot B^{\tilde{\delta}_1 x_i \tilde{s}'_3} \\
& \mathrm{ct}_{4,i} = g^{s_1 \tilde{w}_{2,i}} \cdot C^{\tilde{s}_2 \tilde{f}_{2,i}} \cdot B^{\tilde{\delta}_2 x_i \tilde{s}_3}, \mathrm{ct}'_{4,i} = g^{s'_1 \tilde{w}_{2,i}} \cdot C^{\tilde{s}'_2 \tilde{f}_{2,i}} \cdot B^{\tilde{\delta}_2 x_i \tilde{s}'_3} \\
& \mathrm{ct}_{5,i} = g^{s_1 \tilde{t}_{1,i}} \cdot C^{\tilde{s}_2 \tilde{h}_{1,i}} Z^{\tilde{\theta}_1 x_i}, \mathrm{ct}'_{5,i} = g^{s'_1 \tilde{t}_{1,i}} \cdot C^{\tilde{s}'_2 \tilde{h}_{1,i}} Z^{\tilde{\theta}_1 x_i} \\
& \mathrm{ct}_{6,i} = g^{s_1 \tilde{t}_{2,i}} \cdot C^{\tilde{s}_2 \tilde{h}_{2,i}} Z^{\tilde{\theta}_2 x_i}, \mathrm{ct}'_{6,i} = g^{s'_1 \tilde{t}_{2,i}} \cdot C^{\tilde{s}'_2 \tilde{h}_{2,i}} Z^{\tilde{\theta}_2 x_i}
\end{aligned}$$

$$\mathsf{ct}_8 = Z^{\tilde{\Omega}} \cdot \mathbf{e}(A,C)^{-\tilde{a}} \cdot \Lambda^{-\tilde{s}_2} \cdot m_0, \mathsf{ct}'_8 = Z^{\tilde{\Omega}} \cdot \mathbf{e}(A,C)^{-\tilde{a}} \cdot \Lambda^{-\tilde{s}'_2} \cdot m_0$$

$$\begin{aligned}\mathsf{ct}_{3,i} &= W_{1,i}^{s_1} \cdot F_{1,i}^{s_2} \cdot \delta_1^{x_i s_3} = g^{s_1 w_{1,i}} \cdot g^{f_{1,i}(\tilde{s}_2+\gamma)} \cdot g^{x_i\delta_1(-\gamma\beta+\tilde{s}_3)} = g^{s_1 w_{1,i}} \cdot g^{f_{1,i}\tilde{s}_2} g^{\gamma(f_{1,i}-x_i\delta_1\beta)} \cdot g^{x_i\delta_1\tilde{s}_3)} \\ &= W_{1,i}^{s_1} \cdot g^{\tilde{s}_2(\beta\tilde{\delta}_1 x_i + \tilde{f}_{1,i})} \cdot g^{\gamma(f_{1,i}-x_i\tilde{\delta}_1\beta)} \cdot g^{\tilde{\delta}_1 x_i \tilde{s}_3} = W_{1,i}^{s_1} \cdot B^{\tilde{s}_2\delta_1 x_i} \cdot g^{\tilde{s}_2\tilde{f}_{1,i}} \cdot C^{\tilde{f}_{1,i}} \cdot g^{\delta_1 x_i \tilde{s}_3}\end{aligned}$$

Same computation shows other components generated properly.

Analyzing the game: There exists two cases $Z = \mathbf{e}(g,g)^{\alpha\beta\tau}$ or it is a random element in $\mathbb{Z}_p$. Also note,

$$\begin{aligned}&\Lambda^{-s_2} = \Lambda^{-\tau-\tilde{s}_2} = (\mathbf{e}(A,B)^{-\Omega} \cdot \mathbf{e}(A,g)^{\tilde{a}})^{-\tau} \cdot \Lambda^{-\tilde{s}_2} = \mathbf{e}(h,g)^{\alpha\beta\tau} \cdot \mathbf{e}(A,C)^{-\tilde{a}} \cdot \Lambda^{-\tilde{s}_2}, \\ &\mathsf{ct}_8 = Z^{\Omega} \cdot \mathbf{e}(A,C)^{-\tilde{a}} \cdot \Lambda^{-\tilde{s}_2} \cdot m_0 = Z^{\Omega} \cdot \Lambda^{-s_2} \cdot \mathbf{e}(h,g)^{-\alpha\beta\tau} \cdot m_0 = Z^{\Omega}\mathbf{e}(h,g^{-\alpha\beta\tau}) \cdot \Lambda^{-s_2} \cdot m_0\end{aligned}$$

1. If $Z = \mathbf{e}(g,g)^{\alpha\beta\tau} \Rightarrow Z^{\Omega} \cdot \mathbf{e}(h,g^{-\alpha\beta\tau}) = \mathbf{e}(h,g^{\alpha\beta\tau}) \cdot \mathbf{e}(h,g^{-\alpha\beta\tau}) = 1_{G_T}$ $\mathsf{ct}_8 = \Lambda^{-s_2} \cdot m_0 \Rightarrow$ $\mathcal{A}$ interacts with H_1
2. If Z is a random element then ct_8 is also a random element hence $\mathcal{A}$ interact with H_2

□

Proof of Proposition 7

Proposition 7. *Under DLin assumption* H_2 *and* H_3 *are indistinguishable for all PPT adversaries.*

Proof. The simulator takes as input:

$$(g, A = g^{\alpha}, B = g^{\beta}, C = g^{\tau}, D = g^{\alpha\eta}, Z \stackrel{?}{=} g^{\beta(\eta+\tau)})$$

and by interacting with the adversary $\mathcal{A}$, distinguish between $g^{\beta(\eta+\tau)}$ and a random element.

SetUp phase. The adversary $\mathcal{A}$ sends two vectors $\overrightarrow{x}, \overrightarrow{y}$ to $\mathbb{B}$. The simulator picks $g' \xleftarrow{\$} \mathbb{G}$ and $\delta_b, \theta_b, \tilde{w}_{1,i}, \tilde{t}_{1,i}, f_{b,i}, \tilde{h}_{b,i}, \tilde{\Omega}, r \xleftarrow{\$} \mathbb{Z}_p^{\star}$ for $b = 1,2$ and $i \in [n]$. Then, for each $i \in [n]$, the simulator computes $w_{2,i}, t_{2,i}$ such that:

$$\tilde{\Omega} = \delta_1 \tilde{w}_{2,i} - \delta_2 \tilde{w}_{1,i} = \theta_1 \tilde{t}_{2,i} - \theta_2 \tilde{t}_{1,i}.$$

$\mathbb{B}$ computes the master public key components for $b \in [2], i \in [n]$ as follows:

$$\begin{aligned}&\{W_{b,i} = B^{\delta_b x_i} A^{\tilde{w}_{b,i}}, F_{b,i} = g^{f_{b,i}}, T_{b,i} = B^{\theta_b x_i} A^{\tilde{t}_{b,i}}, H_{b,i} = B^{\theta_b x_i} g^{\tilde{h}_{b,i}}\}_{b\in[2],i\in[n]} \\ &\{U_b = g^{\delta_b}, V_b = g^{\theta_b}\}_{b\in[2]},, h = A^{\tilde{\Omega}}, \Lambda = \mathbf{e}(g,g'), K_1 = g^k,\ K_2 = g'^{\frac{1}{k}}.\end{aligned}$$

The simulator knows exact value of $\{f_{b,i}, \delta_b, \theta_b\}_{i\in[n],b\in[2]}$ and for the rest implicitly define the secret parameters as follows:

$$w_{b,i} = \beta\delta_b x_i + \alpha\tilde{w}_{b,i}, t_{b,i} = \beta\theta_b x_i + \alpha\tilde{t}_{b,i}, h_{b,i} = \beta\theta_b x_i + \tilde{h}_{b,i}, \Omega = \alpha\tilde{\Omega}$$

Observe that:

$$\begin{aligned}\delta_1 w_{2,i} - \delta_2 w_{1,i} &= \delta_1(\cancel{\beta\delta_2 x_i} + \alpha\tilde{w}_{2,i}) - \delta_2(\cancel{\beta\delta_1 x_i} + \alpha\tilde{w}_{1,i})\\ &= \alpha(\tilde{w}_{2,i}\delta_1 - \delta_2\tilde{w}_{1,i})\\ &= \alpha(\tilde{t}_{2,i}\theta_1 - \theta_2\tilde{t}_{1,i})\\ &= \theta_1 t_{2,i} - \theta_2 t_{1,i}\\ &= \alpha\tilde{\Omega}\\ &= \Omega\end{aligned}$$

Thus, the simulator generates the master public key as the real setup algorithm.

Token query phase. The simulator chooses $\tilde{\lambda}_1, \tilde{\lambda}_2, \{\tilde{r}_i, \tilde{\Phi}_i\}_{i\in[n]} \overset{\$}{\leftarrow} \mathbb{Z}_p^\star$, and then implicitly defines the following randomnesses:

$$\lambda_1 = -\frac{\tilde{\lambda}_2}{\alpha} + \tilde{\lambda}_1, \lambda_2 = \frac{\tilde{\lambda}_2}{\alpha}, r_i = -\frac{\beta v_i x_i\tilde{\lambda}_2}{\alpha} + \tilde{r}_i, \Phi_i = \frac{\beta v_i x_i \tilde{\lambda}_2}{\alpha} + \tilde{\Phi}_i\alpha$$

By that setting the simulator generates the token as:

$$\log_g K_{3,i} = -\delta_2 r_i + \lambda_1 v_i w_{2,i} = -\delta_2(-\frac{\beta v_i x_i\tilde{\lambda}_2}{\alpha} + \tilde{r}_i) + (-\frac{\tilde{\lambda}_2}{\alpha} + \tilde{\lambda}_1) v_i w_{2,i} =$$

$$\frac{v_i\tilde{\lambda}_2}{\alpha}\underbrace{(x_i\delta_2\beta - w_{2,i})}_{-\alpha\tilde{w}_{2,i}} - \delta_2\tilde{r}_i + \tilde{\lambda}_1 v_i w_{2,i} = -v_i\tilde{\lambda}_2\tilde{w}_{2,i} - \delta_2\tilde{r}_i + \tilde{\lambda}_1 v_i w_{2,i}$$

$$\Rightarrow K_{3,i} = g^{-\delta_2 r_i}\cdot g^{\lambda_1 v_i w_{2,i}} = g^{-\delta_2\tilde{r}_i}\cdot g^{-\tilde{\lambda}_2 v_i \tilde{w}_{2,i}}\cdot W_{2,i}^{\tilde{\lambda}_1 v_i} \Rightarrow K_{3,i} \text{ is computable}$$

$$\log_g K_{5,i} = -\theta_2\Phi_i + \lambda_2 v_i t_{2,i} = -\theta_2(\frac{\beta v_i x_i\tilde{\lambda}_2}{\alpha} + \tilde{\Phi}_i) + (\frac{\tilde{\lambda}_2}{\alpha}) v_i t_{2,i}$$

$$= -\frac{v_i\tilde{\lambda}_2}{\alpha}\underbrace{(\theta_2\beta x_i - t_{2,i})}_{-\alpha\tilde{t}_{2,i}} - \theta_2\tilde{\Phi}_i = v_i\tilde{\lambda}_2\tilde{t}_{2,i} - \theta_2\tilde{\Phi}_i$$

$$\Rightarrow K_{5,i} = g^{-\theta_2\Phi_i}\cdot g^{\lambda_2 v_i t_{2,i}} = g^{-\theta_2\tilde{\Phi}_i}\cdot g^{v_i\tilde{\lambda}_2\tilde{t}_{2,i}} \Rightarrow K_{5,i} \text{ is computable.}$$

Same computation: $K_{4,i} = g^{\delta_1\tilde{r}_i}\cdot g^{\tilde{\lambda}_1 v_i \tilde{w}_{1,i}}\cdot W_{1,i}^{-\tilde{\lambda}_1 v_i}, K_{6,i} = g^{\theta_1\tilde{\Phi}_i} g^{-\tilde{\lambda}_2 v_i \tilde{t}_{1,i}}$

$$K_B = \prod_{i=1}^{n} g^{-(r_i+\Phi_i)} = \prod_{i=1}^{n} g^{-(-\frac{\beta v_i x_i\tilde{\lambda}_2}{\alpha} + \tilde{r}_i + \frac{\beta v_i x_i \tilde{\lambda}_2}{\alpha} + \tilde{\Phi}_i)} = \prod_{i=1}^{n} g^{-(\tilde{r}_i + \tilde{\Phi}_i)}$$

To compute K_A notice that the simulator knows $f_{b,i}$ hence $\prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} K_{4,i}^{-f_{2,i}}$ is computable. For the remaining part $K_{5,i}^{-h_{1,i}} K_{6,i}^{-h_{2,i}}$, consider the following:

$$\begin{aligned}K_{5,i}^{-h_{1,i}} K_{6,i}^{-h_{2,i}} &= g^{h_{1,i}\theta_2\tilde{\Phi}_i} g^{-h_{1,i}\tilde{\lambda}_2 v_i \tilde{t}_{2,i}} g^{-h_{2,i}\theta_1\tilde{\Phi}_i} g^{h_{2,i}\tilde{\lambda}_2 v_i \tilde{t}_{1,i}}\\ &= H_{1,i}^{\theta_2\tilde{\Phi}_i}\cdot H_{1,i}^{-\tilde{\lambda}_2 v_i\tilde{t}_{2,i}} H_{2,i}^{-\theta_1\tilde{\Phi}_i}\cdot H_{2,i}^{\tilde{\lambda}_2 v_i \tilde{t}_{1,i}} \Rightarrow K_{5,i}^{-h_{1,i}} K_{6,i}^{-h_{2,i}} \text{ is computable}\end{aligned}$$

This shows that the simulator can compute the token as the real challenger.

Challenge Phase: To generate the ciphertext, $\mathbb{B}$ chooses random elements

$$\tilde{s}_1, \dots, \tilde{s}_3, \tilde{s}'_1, \dots, \tilde{s}'_3 \xleftarrow{\$} \mathbb{Z}_p^\star : \ \tilde{s}_3 \neq \tilde{s}'_3$$

and computes the challenge ciphertext as follows:

- $\mathsf{ct}_1 = C \cdot g^{\tilde{s}_2} = g^{\tau + \tilde{s}_2} \Rightarrow s_2 = \tau + \tilde{s}_2,$
- $\mathsf{ct}'_1 = C \cdot g^{\tilde{s}'_2} = g^{\tau + \tilde{s}'_2} \Rightarrow s'_2 = \tau + \tilde{s}_2$
- $\mathsf{ct}_2 = D^{\tilde{\Omega}} \cdot A^{\tilde{\Omega}\tilde{s}_1} = (g^{\alpha\tilde{\Omega}})^{(\eta + \tilde{s}_1)} = h^{\eta + \tilde{s}_1} \Rightarrow s_1 = \eta + \tilde{s}_1$
- $\mathsf{ct}'_2 = D^{\tilde{\Omega}} \cdot A^{\tilde{\Omega}\tilde{s}'_1} \Rightarrow s'_1 = \eta + \tilde{s}'_1$
- $\mathsf{ct}_{3,i} = W_{1,i}^{\tilde{s}_1} \cdot F_{1,i}^{\tilde{s}_2} \cdot U_1^{\tilde{s}_3 x_i} \cdot D^{\tilde{w}_{1,i}} \cdot C^{f_{1,i}} = W_{1,i}^{\tilde{s}_1} \cdot F_{1,i}^{\tilde{s}_2 + \tau} \cdot U_1^{\tilde{s}_3 x_i} \cdot g^{\eta\alpha\tilde{w}_{1,i}} \cdot F_{1,i}^{\tau} =$
$= W_{1,i}^{\tilde{s}_1} \cdot F_{1,i}^{\tilde{s}_2 + \tau} \cdot U_1^{\tilde{s}_3 x_i} \cdot g^{\eta(w_{1,i} - \beta\delta_1 x_i)} = W_{1,i}^{\tilde{s}_1 + \eta} \cdot F_{1,i}^{\tilde{s}_2 + \tau} \cdot U_1^{(\tilde{s}_3 - \eta\beta)x_i} \Rightarrow s_3 = -\eta\beta + \tilde{s}_3$
- $\mathsf{ct}_{4,i} = W_{2,i}^{\tilde{s}_1} \cdot F_{2,i}^{\tilde{s}_2} \cdot U_2^{\tilde{s}_3 x_i} \cdot D^{\tilde{w}_{2,i}} \cdot C^{f_{2,i}}$, (similar computation as $\mathsf{ct}_{3,i}$)

and,

- $\mathsf{ct}'_{3,i} = W_{1,i}^{\tilde{s}'_1} \cdot F_{1,i}^{\tilde{s}'_2} \cdot U_1^{\tilde{s}'_3 x_i} \cdot D^{\tilde{w}_{1,i}} \cdot C^{f_{1,i}}$
- $\mathsf{ct}'_{4,i} = W_{2,i}^{\tilde{s}'_1} \cdot F_{2,i}^{\tilde{s}'_2} \cdot U_2^{\tilde{s}'_3 x_i} \cdot D^{\tilde{w}_{2,i}} \cdot C^{f_{2,i}}$
- $\mathsf{ct}_{5,i} = T_{1,i}^{\tilde{s}_1} \cdot D^{\tilde{t}_{1,i}} \cdot H_{1,i}^{\tilde{s}_2} \cdot C^{\tilde{h}_{1,i}} \cdot Z^{\theta_1 x_i},$
- $\mathsf{ct}'_{5,i} = T_{1,i}^{\tilde{s}'_1} \cdot D^{\tilde{t}_{1,i}} \cdot H_{1,i}^{\tilde{s}'_2} \cdot C^{\tilde{h}_{1,i}} \cdot Z^{k\theta_1 x_i}$
- $\mathsf{ct}_{6,i} = T_{2,i}^{\tilde{s}_1} \cdot D^{\tilde{t}_{2,i}} \cdot H_{2,i}^{\tilde{s}_2} \cdot C^{\tilde{h}_{2,i}} \cdot Z^{\theta_2 x_i},$
- $\mathsf{ct}'_{6,i} = T_{2,i}^{\tilde{s}'_1} \cdot D^{\tilde{t}_{2,i}} \cdot H_{2,i}^{\tilde{s}'_2} \cdot C^{\tilde{h}_{2,i}} \cdot Z^{\theta_2 x_i}$

Analysis the game: First, notice that:

$$\begin{aligned}
&D^{\tilde{t}_{1,i}} = g^{\eta\alpha\tilde{t}_{1,i}} = g^{\eta(t_{1,i} - \beta\theta_1 y_i)} = T_{1,i}^{\eta} \cdot g^{-\beta\eta\theta_1 y_i}, D^{\tilde{t}_{1,i}} = T_{1,i}^{\eta} \cdot g^{-k\beta\eta\theta_1 y_i} \\
&C^{\tilde{h}_{1,i}} = g^{\tau(h_{1,i} - \beta\theta_1 y_i)} = H_{1,i}^{\tau} \cdot g^{-\beta\tau\theta_1 y_i}, C^{\tilde{h}_{1,i}} = H_{1,i}^{\tau} \cdot g^{-\beta\tau\theta_1 y_i} \\
&\quad \Rightarrow \\
&\mathsf{ct}_{5,i} = T_{1,i}^{\tilde{s}_1} \cdot D^{\tilde{t}_{1,i}} \cdot H_{1,i}^{\tilde{s}_2} \cdot C^{\tilde{h}_{1,i}} \cdot Z^{\theta_1 y_i} = \\
&\qquad = T_{1,i}^{\tilde{s}_1} \cdot T_{1,i}^{\eta} \cdot g^{-\beta\eta\theta_1 y_i} \cdot H_{1,i}^{\tilde{s}_2} \cdot H_{1,i}^{\tau} \cdot g^{-\beta\tau\theta_1 y_i} \cdot Z^{\theta_1 \tilde{y}_i} \\
&\qquad = T_{1,i}^{\eta + \tilde{s}_1} \cdot H_{1,i}^{\tau + \tilde{s}_2} \cdot (g^{-\beta(\tau+\eta)} \cdot Z)^{\theta_1 y_i} = T_{1,i}^{s_1} \cdot H_{1,i}^{s_2} \cdot (g^{-\beta(\tau+\eta)} \cdot Z)^{\theta_1 y_i} \\
&\mathsf{ct}'_{5,i} = T_{1,i}^{s'_1} \cdot H_{1,i}^{s'_2} \cdot (g^{-\beta(\tau+\eta)} \cdot Z)^{\theta_1 y_i}
\end{aligned}$$

$$\text{If } Z = g^{\beta(\eta+\tau)} \qquad \Rightarrow \begin{cases} g^{-\beta(\tau+\eta)} \cdot Z = 1_{\mathbb{G}} \Rightarrow \mathsf{ct}_{5,i} = T_{1,i}^{s_1} \cdot H_{1,i}^{s_2} \\ g^{(-\beta(\tau+\eta)} \cdot Z = 1_{\mathbb{G}} \Rightarrow \mathsf{ct}_{5,i} = T_{1,i}^{s'_1} \cdot H_{1,i}^{s'_2} \end{cases}$$

$\Rightarrow$ The adversary interact with hybrid H_3

$$\text{If } Z = g^r \quad \Rightarrow \begin{cases} g^{-\beta(\tau+\eta)} \cdot Z = g^{r-\beta(\tau+\eta)} \xrightarrow{s_4 = r-\beta(\tau+\eta)} \mathsf{ct}_{5,i} = T_{1,i}^{s_1} \cdot H_{1,i}^{s_2} \cdot U_1^{s_4 y_i} \\ g^{(-\beta(\tau+\eta)} \cdot Z = g^{r'} \Rightarrow \mathsf{ct}_{5,i} = T_{1,i}^{s'_1} \cdot H_{1,i}^{s'_2} \cdot U_1^{r' y_i} \end{cases}$$

$\Rightarrow$ The adversary interact with hybrid H_2

□

Proof of Proposition 8

Proposition 8. *Under DLin assumption* H_3 *and* H_4 *are indistinguishable for all PPT adversaries* $\mathcal{A}$.

Proof. The simulator takes as input $(g, A = g^\alpha, B = g^\beta, C = g^\tau, D = g^{\alpha\eta}, Z \overset{?}{=} g^{\beta(\eta+\tau)})$ and by interacting with the adversary $\mathcal{A}$, distinguish between $g^{\beta(\eta+\tau)}$ and a random element.

SetUp phase. Generating master public key is same as in 7, except that instead of $\overrightarrow{x} = (x_1, \ldots, x_n)$ we use $\overrightarrow{y} = (y_1, \ldots, y_n)$ to compute $\{T_{b,i}, H_{b,i}\}_{i\in[n]}$:

$$\{W_{b,i} = B^{\delta_b x_i} A^{\tilde{w}_{b,i}}, F_{b,i} = g^{f_{b,i}}, T_{b,i} = B^{\theta_b y_i} A^{\tilde{t}_{b,i}}, H_{b,i} = B^{\theta_b y_i} g^{\tilde{h}_{b,i}}\}_{b\in[2], i\in[n]}$$
$$\{U_b = g^{\delta_b}, V_b = g^{\theta_b}\}_{b\in[2]},, h = A^{\tilde{\Omega}}, \Lambda = \mathbf{e}(g, g'), K_1 = g^k,\ K_2 = g'^{\frac{1}{k}}.$$

Which means the simulator implicitly defines:

$$w_{b,i} = \beta\delta_b x_i + \alpha\tilde{w}_{b,i}, t_{b,i} = \beta\theta_b y_i + \alpha\tilde{t}_{b,i}, h_{b,i} = \beta\theta_b y_i + \tilde{h}_{b,i}, \Omega = \alpha\tilde{\Omega}$$

Proving that the simulator generates the master public key, with the distribution same as a real challenger is same as 7.

Token query phase. The simulator chooses $\tilde{\lambda}_1, \tilde{\lambda}_2, \{\tilde{r}_i, \tilde{\Phi}_i\}_{i\in[n]} \xleftarrow{\$} \mathbb{Z}_p^\star$, and then implicitly defines the following randomnesses:

$$\lambda_1 = \tilde{\lambda}_1 - \frac{c_y\tilde{\lambda}_2}{\alpha},\ \lambda_2 = \frac{c_x\tilde{\lambda}_2}{\alpha},\ r_i = \tilde{r}_i - \frac{c_y\beta v_i x_i\tilde{\lambda}_2}{\alpha},\ \Phi_i = \tilde{\Phi}_i + \frac{c_x\beta v_i y_i\tilde{\lambda}_2}{\alpha}$$

The following computation shows the simulator can compute the token without knowing the exact value of the randomnesses:

$$\log_g K_{3,i} = -\delta_2 r_i + \lambda_1 v_i w_{2,i} = -\delta_2(\tilde{r}_i - \frac{c_y\beta v_i x_i\tilde{\lambda}_2}{\alpha}) + (\tilde{\lambda}_1 - \frac{c_y\tilde{\lambda}_2}{\alpha}) v_i w_{2,i}$$
$$= -\delta_2\tilde{r}_i + \frac{c_y\tilde{\lambda}_2}{\alpha} v_i \underbrace{(\beta x_i - w_{2,i})}_{\alpha\tilde{w}_{2,i}} + \tilde{\lambda}_1 v_i w_{2,i} = -\delta_2\tilde{r}_i + c_y\tilde{\lambda}_2 v_i\tilde{w}_{2,i} + \tilde{\lambda}_1 v_i w_{2,i} =$$
$$\Rightarrow K_{3,i} = g^{-\delta_2\tilde{r}_i} \cdot g^{c_y\tilde{\lambda}_2 v_i\tilde{w}_{2,i}} \cdot W_{2,i}^{\tilde{\lambda}_1 v_i} \Rightarrow K_{3,i} \text{is compuatble}$$

$$\log_g K_{5,i} = -\theta_2\Phi_i + \lambda_2 v_i t_{2,i} = -\theta_2(\tilde{\Phi}_i + \frac{c_x\beta v_i y_i\tilde{\lambda}_2}{\alpha}) + \frac{c_x\tilde{\lambda}_2}{\alpha} v_i t_{2,i} = -\theta_2\tilde{\Phi}_i + \frac{c_x v_i\tilde{\lambda}_2}{\alpha}\underbrace{(t_{2,i} - \theta_2\beta y_i)}_{\alpha\tilde{t}_{2,i}} =$$
$$\Rightarrow K_{5,i} = g^{-\theta_2\tilde{\Phi}_i} \cdot g^{c_x\tilde{\lambda}_2 v_i\tilde{t}_{2,i}} \Rightarrow K_{5,i} \text{ is compuatble}$$

Similar computation $\Rightarrow K_{4,i} = g^{\delta_1\tilde{r}_i} \cdot g^{-c_y\tilde{\lambda}_2 v_i\tilde{w}_{1,i}} \cdot W_{1,i}^{-\tilde{\lambda}_1 v_i}, K_{6,i} = g^{\theta_1\tilde{\Phi}_i} \cdot g^{-c_x\tilde{\lambda}_2 v_i\tilde{t}_{1,i}}$

$$\begin{aligned}K_A &= g' \cdot \prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} K_{4,i}^{-f_{2,i}} K_{5,i}^{-h_{1,i}} K_{6,i}^{-h_{2,i}} = g' \cdot \prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} K_{4,i}^{-f_{2,i}} K_{5,i}^{-\beta\theta_1 y_i - \tilde{h}_{1,i}} K_{6,i}^{-\beta\theta_2 y_i + \tilde{h}_{2,i}} \\ &= g' \cdot \prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} K_{4,i}^{-f_{2,i}} K_{5,i}^{-\tilde{h}_{1,i}} K_{6,i}^{-\tilde{h}_{2,i}} (\cancel{g^{-\theta_2 \Phi_i}} g^{\lambda_2 v_i t_{2,i}})^{-\beta\theta_1 y_i} (\cancel{g^{-\theta_1 \Phi_i}} g^{\lambda_2 v_i t_{1,i}})^{-\beta\theta_2 y_i} \\ &= g' \cdot \prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} K_{4,i}^{-f_{2,i}} K_{5,i}^{-\tilde{h}_{1,i}} K_{6,i}^{-\tilde{h}_{2,i}} g^{-\lambda_2 v_i \beta y_i (t_{2,i}\theta_1 - t_{1,i}\theta_2)} \\ &= g' \cdot \prod_{i=1}^{n} K_{3,i}^{-f_{1,i}} K_{4,i}^{-f_{2,i}} K_{5,i}^{-\tilde{h}_{1,i}} K_{6,i}^{-\tilde{h}_{2,i}} B^{-\tilde{\lambda}_2 v_i y_i \tilde{\Omega}} \Rightarrow K_A \text{ is computable}\end{aligned}$$

$$K_B = \prod_{i=1}^{n} g^{-(r_i+\Phi_i)} = \prod_{i=1}^{n} g^{-\tilde{r}_i + \frac{c_y \tilde{\lambda}_2 v_i x_i \beta}{\alpha} - \tilde{\Phi}_i - \frac{c_x \tilde{\lambda}_2 v_i y_i \beta}{\alpha}} = \prod_{i=1}^{n} g^{-(\tilde{r}_i+\tilde{\Phi}_i) + \frac{c_y \tilde{\lambda}_2 \beta}{\alpha}(c_y v_i x_i - c_x v_i y_i)} =$$
$$= g^{-\sum_{i=1}^{n}(\tilde{r}_i+\tilde{\Phi}_i)} g^{\frac{\tilde{\lambda}_2 \beta}{\alpha}(c_x \sum_{i=1}^{n} v_i y_i - c_y \sum_{i=1}^{n} v_i x_i)} = \prod_{i=1}^{n} g^{-(\tilde{r}_i+\tilde{\Phi}_i)} \cdot g^{\frac{\tilde{\lambda}_2 \beta}{\alpha}(c_y c_x - c_x c_y)} = \prod_{i=1}^{n} g^{-(\tilde{r}_i+\tilde{\Phi}_i)}$$

This shows that the simulator can compute the token as the real challenger.

Generating the challenge ciphertext. Simulator does the exact steps as in 7 to generate the challenge ciphertext for all components except $\mathsf{ct}_{5,i}, \mathsf{ct}_{6,i}, \mathsf{ct}'_{5,i}, \mathsf{ct}'_{6,i}$, which instead $\{x_i\}_i$ puts $\{y_i\}_i$ as power of Z:

$$\mathsf{ct}_{5,i} = T_{1,i}^{\tilde{s}_1} \cdot D^{\tilde{t}_{1,i}} \cdot H_{1,i}^{\tilde{s}_2} \cdot C^{\tilde{h}_{1,i}} \cdot Z^{\theta_1 y_i} \qquad \mathsf{ct}_{6,i} = T_{2,i}^{\tilde{s}_1} \cdot D^{\tilde{t}_{2,i}} \cdot H_{2,i}^{\tilde{s}_2} \cdot C^{\tilde{h}_{2,i}} \cdot Z^{\theta_2 y_i}$$
$$\mathsf{ct}'_{5,i} = T_{1,i}^{\tilde{s}'_1} \cdot D^{\tilde{t}_{1,i}} \cdot H_{1,i}^{\tilde{s}'_2} \cdot C^{\tilde{h}_{1,i}} \cdot Z^{\theta_1 y_i} \qquad \mathsf{ct}'_{6,i} = T_{2,i}^{\tilde{s}'_1} \cdot D^{\tilde{t}_{2,i}} \cdot H_{2,i}^{\tilde{s}'_2} \cdot C^{\tilde{h}_{2,i}} \cdot Z^{\theta_2 y_i}$$

Analysis the game:

- If $Z = g^{\beta(\eta+\tau)} \Rightarrow \mathsf{ct}_{5,i} = T_{1,i}^{s_1} H_{1,i}^{s_2} Z^{-\theta_1 y_i} Z^{\theta_1 y_i} = T_{1,i}^{s_1} H_{1,i}^{s_2} \Rightarrow \mathcal{A}$ interacts with hybrid H_3
- If $Z = g^{r} \Rightarrow \mathsf{ct}_{5,i} = T_{1,i}^{s_1} H_{1,i}^{s_2} \cdot g^{-\beta(\eta+\tau)\theta_1 y_i} \cdot g^{\theta_1 y_i r} = T_{1,i}^{s_1} H_{1,i}^{s_2} \cdot g^{(r-\beta(\eta+\tau))\theta_1 y_i}$
  $\xrightarrow{s_4 = r-\beta(\eta+\tau)\neq 0} \mathsf{ct}_{5,i} = T_{1,i}^{s_1} H_{1,i}^{s_3} V_1^{s_4 y_i}$ which is hybrid H_4

□

Appendix .B

Groth-Sahai Pairing product Equation for BGN relation

Using the Groth-Sahai proof technique we have the following equation for the relation $\mathsf{R}_{\mathsf{ballot}}$ defined in 7.17:

Variables:

$$\mathcal{M} = g^{\mathsf{vote}}, \mathcal{M}_i = g^{\mathsf{vote}^i}, \mathcal{X} = g^{\mathsf{crd}}, \mathcal{A}_i = g^{\hat{a}^i}, \mathcal{CA}_i = \mathsf{CA}_i,$$
$$\hat{\mathcal{M}} = h^{\mathsf{r}_v}, \hat{\mathcal{M}}_i = h^{\mathsf{r}_i}, \hat{\mathcal{X}} = h^{\mathsf{r}_{\mathsf{crd}}}, \hat{\mathcal{A}}_i = h^{\mathsf{r}'_i}, \mathcal{C}\hat{\mathcal{A}}_i = h^{\mathsf{r}^*_i}$$

Equation

$$\begin{aligned}
&\mathbf{e}(\mathsf{CT}_{\mathsf{vote}}, g) = \mathbf{e}(g^{\mathsf{vote}} \cdot h^{\mathsf{r}_v}, g) = \mathbf{e}(\mathcal{M}, g) \cdot \mathbf{e}(\hat{\mathcal{M}}, g) \\
&\mathbf{e}(\mathsf{CT}_{\mathsf{crd}}, g) = \mathbf{e}(g^{\mathsf{crd}} \cdot h^{\mathsf{r}_{\mathsf{crd}}}, g) = \mathbf{e}(\mathcal{X}, g) \cdot \mathbf{e}(\hat{\mathcal{X}}, g) \\
\text{for } & i = 1, \ldots, k : \\
&\mathbf{e}(\mathsf{CA}_i, g) = \mathbf{e}(g^{\hat{a}^i} \cdot h^{\mathsf{r}'_i}, g) = \mathbf{e}(\mathcal{A}_i, g) \cdot \mathbf{e}(\hat{\mathcal{A}}_i, g) \\
&\mathbf{e}(\mathsf{CA}_i, g) = \mathbf{e}(g^{\hat{a}^i}, g) = \mathbf{e}(g^{\hat{a}^{i-1}}, g^{\hat{a}}) = \mathbf{e}(\mathcal{A}_{i-1}, \mathcal{A}_1) \\
&\mathbf{e}(\mathsf{CA}^*_i, g) = \mathbf{e}(\mathsf{CA}_i \cdot h^{\mathsf{r}^*_i}, g) = \mathbf{e}(\mathcal{CA}_i, g) \cdot \mathbf{e}(\hat{\mathcal{CA}}_i, g) \\
&\mathbf{e}(\mathcal{M}_k, g_2) \cdot \mathbf{e}(\mathcal{M}_{k-1}, g_2)^{p_1} \cdot \ldots \cdot \mathbf{e}(\mathcal{M}_1, g_2)^{p_{k-1}} \cdot \mathbf{e}(g_1, g_2)^{p_k} = 1_{\mathbb{G}_T} \\
\text{for } & i = 1, 2, \ldots, m : \\
&\mathbf{e}(\mathcal{M}_i, g) \cdot \mathbf{e}(\mathcal{M}_{i-1}, \mathcal{M})^{-1} = 1_{\mathbb{G}_T}
\end{aligned} \tag{4}$$

Notice that to prove the validity of the encrypted vote, $\mathsf{vote} \in \mathsf{cList} = \{\mathsf{c}_1, \ldots, \mathsf{c}_m\}$ the voter is required to prove:

$$\mathsf{CT}_{\mathsf{vote}} = \mathsf{Enc}(\mathsf{vote}), \mathsf{vote} \in \mathsf{cList}$$

Or equivalently prove that:

$$\mathsf{CT}_{\mathsf{vote}} = \mathsf{Enc}(\mathsf{vote}) : \mathsf{Poly}_{\mathcal{C}}(\mathsf{vote}) = 0$$

Where the polynomial $\mathsf{Poly}_{\mathcal{C}}$ is defined as follows:

$$\mathsf{Poly}_{\mathcal{C}} = \prod_{i=1}^{m} (x - \mathsf{c}_i) = \sum_{i=0}^{m} c_i x^i$$