Chapter 1

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!

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 [34], and became one of the essential underlying primitives in cryptography. According to Goldreich, Micali, and Wigderson [32], 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.

1.1 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 **??**).

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 [27], 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 1. *[**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:*

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

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

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

- **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 1.1.1 ([27])**.** *Suppose that $\mathscr{L}$ has a unidirectional Zero-Knowledge proof system, then $\mathscr{L} \in$ **BPP**.*

Theorem 1.1.2 ([27])**.** *Suppose that $\mathscr{L}$ has a Zero-Knowledge proof system in which the verifier program is deterministic, then $\mathscr{L} \in$ **BPP**.*

Theorem 1.1.3 ([27])**.** *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 1.1.4 ([27])**.** *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:

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

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

1.2 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 [51] 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 [31] 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:

Theorem 1.2.1. *[27] The protocol demonstrated in Figure 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 1.2.2. *If one-way functions exist, then every **NP**-language has a zero-knowledge interactive proof system.*

1.3 Zero-Knowledge Proof Systems; Variants

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

FIGURE 1.1: The Zero-Knowledge proof system for Graph 3-Colouring [27]

• **Setting:**	$\mathcal{G} = (G, V, E, n) : V = \{1, \ldots, n\}, E = \{(i, j) : \text{ vertix } i \text{ connect to } j\}$
	$\mathrm{R}_g = \{(x, w) : x = \mathcal{G}, w = \phi : V \mapsto \{\text{blue}, \text{red}, \text{green}\}$
	$(i, j) \in E \implies \phi(i) \neq \phi(j)$
• **Inputs:**	Prover x, w, Verifier: x
Prover commitment:	
	1. Pick a random permutation σ over $\{\text{blue}, \text{red}, \text{green}\}$
	2. For $i = 1, \ldots n$ commitment to the value $\sigma(\phi(i))$
	3. Compute $\mathsf{Com} = \mathsf{Com}(\sigma(\phi(i)))$
	Prover $\xrightarrow{\mathsf{Com}}$ Verifier
Verifier challenge	
	Pick $e = (i, j) \xleftarrow{\$} E$
	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 $\sigma(\phi(i)) \neq \sigma(\phi(j))$

1.3.1 Simulator with Auxiliary Input

In the original definition, the simulator does not take any auxiliary input [34]. In contrast, in the revisited definition [33], 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:

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

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 [29] and some with a non-black-box simulator [1])

1.3.2 Perfect, Statistical, Computational ZK

Recall that three variants of indistinguishability; Perfect, statistical and computational indistinguishability(see **??**). Each indistinguishable interpretation provides a different form of Zero-Knowledge that has been widely investigated in the literature. [30]

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

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

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

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

- 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 [31], followed by a stronger result proven in [43, 7], which state that:

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

1.3.3 Expected Polynomial-Time Simulators

Following [2] 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 [28, 48], 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 2. *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* $\left\{\mathcal{V}iew\left[\mathsf{Verify}^{\rightleftharpoons\mathsf{Prove}}\right](x)\right\}_{x\in\mathscr{L}}$ *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.*

1.3.4 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 3 (**Knowledge Tightness** [27])**.** *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* 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 [27].

1.3.5 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 4. *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 1.*

1.4 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:

$$\begin{aligned} \mathscr{L}_g &: \mathrm{R}_g = \left\{(x,w) : x = (\mathbb{G}, g, h), h = g^w\right\} \\ \mathscr{L}_{\mathsf{ddh}} &: \mathrm{R}_{ddh} = \left\{(x,w) : x = \big((g,h),(u,v)\big), u = g^w, v = h^w\right\} \end{aligned} \tag{1.1}$$

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 5 (**Message-Specification Function**[27])**.** *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 6 (**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* $\mathsf{Extract}^{\rightleftharpoons \mathsf{Prove}_{x,w;r}}$ *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 1.4.1 ([27])**.** *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.*

1.5 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 7 (**Sigma-Protocol** [52])**.** *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 $\mathbb{G}$ **??**. Then the following is a sigma protocol:

FIGURE 1.2: 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:**	Prove : x, w, Verify : x
• **Protocol:**	
Prover Commitment:	
Round 1:	Pick 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 1.5.1. *[52] 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 [24], 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 0.8.4).

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,

FIGURE 1.3: 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 $g^{z_1} \cdot z_2^n = \mathsf{ct}^e \cdot \mathsf{Com}$

we frequently rely on the non-interactive version of this proof to design a verifiable e-voting protocol.

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

1.6 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.

1.6.1 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 [29]. Yet, the modified version (simulator with auxiliary inputs) retains its Zero-Knowledgeproperty after sequential repetition. [33]

1.6.2 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 counter-example 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") [27].

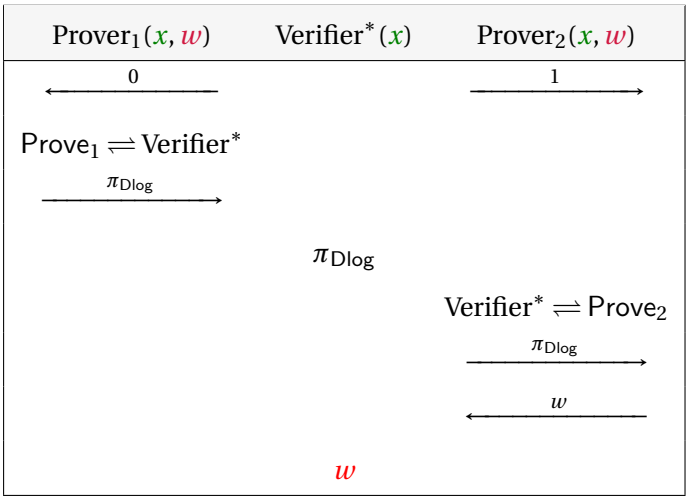
The parallel composition conjecture can also be refuted for probabilistic polynomial-time prover (with auxiliary inputs) and statistical Zero-Knowledge proof systems.

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 [20]. Consider the Zero-Knowledge system for discrete logarithm language in (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 1.4: 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*.

1.7 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 [20].

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

1.7.1 Witness Indistinguishability

Definition 8 (**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:*

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

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

Definition 9 (**Strong Witness Indistinguishable**[27]). *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*

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

Additional Note. According to [27] 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.

1.7.2 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 [20, 15, 47, 42, 17].

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

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

the following holds true:

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

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, ⟨Prove, Verify⟩ 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.

1.8 Non-Interactive Zero Knowledge Proof Systems

As stated in [6], 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 [33], 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.

1.8.1 NIZK in RHB Model

In the hidden-bits model of the NIZK proof system [19, 26], 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 10 (**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:*

$$\begin{gathered} \big\{(x, (I, s_I), \pi) | s \leftarrow \mathsf{crsGen}(1^\ell), \pi \leftarrow \mathsf{Prove}(s, x, w)\big\} \\ \approx \\ \big\{(x, (I, s_I), \pi) | s \xleftarrow{\$} \{0,1\}^{\mathsf{poly}}, \pi \leftarrow \mathsf{Sim}(s, x)\big\} \end{gathered} \tag{1.2}$$

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 1.8.1. *[19] 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.*

1.8.2 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 [16]. 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 11 (**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:*

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

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

$$\begin{gathered}\left\{(x, \mathsf{crs}, \pi) | \mathsf{crs} \leftarrow \mathsf{crsGen}(1^\ell), \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\right\}_\ell \\ \approx \\ \left\{(x, \mathsf{crs}, \pi) | \mathsf{crs} \leftarrow \mathsf{crsGen}(1^\ell), \pi \leftarrow \mathsf{Sim}(\mathsf{crs}, x)\right\}_\ell\end{gathered} \tag{1.3}$$

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 4 is replaced with:

$$\begin{gathered}\{(x, \mathsf{crs}, \pi) | \mathsf{crs} \leftarrow \mathsf{crsGen}(1^\ell), \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\}_\ell \\ \approx \\ \{(x, \mathsf{crs}, \pi) | \mathsf{crs}^* \leftarrow \mathsf{Sim}_1(1^\ell), \pi \leftarrow \mathsf{Sim}_2(\mathsf{crs}, x)\}_\ell\end{gathered} \tag{1.4}$$

Composable Zero-Knowledge: The composable zero-knowledge property was first introduced in [37], 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:

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

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

$$\begin{aligned}&\Pr\left[\mathcal{A}(\pi) = 1 \wedge (x, w) \in \mathrm{R}_n \mid \begin{array}{c}(\mathsf{crs}^*, \tau) \leftarrow \mathsf{Sim}_1(1^\ell) \\ (x, w) \leftarrow \mathcal{A}(\mathsf{crs}^*, \tau) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)\end{array}\right] \\ \approx &\Pr\left[\mathcal{A}(\pi) = 1 \wedge (x, w) \in \mathrm{R}_n \mid \begin{array}{c}(\mathsf{crs}^*, \tau) \leftarrow \mathsf{Sim}_1(1^\ell) \\ (x, w) \leftarrow \mathcal{A}(\mathsf{crs}^*, \tau) \\ \pi \leftarrow \mathsf{Sim}_2(\mathsf{crs}, x, \tau)\end{array}\right]\end{aligned} \tag{1.5}$$

Theorem 1.8.2. *[19]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 R_{OR-DDH} is shown in Figure **??**.

1.8.3 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 [19], 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 [38] 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 1.8.3 ([27])**.** *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.*

1.8.4 Fiat-Shamir Heuristic

The Fiat–Shamir heuristic, introduced in [21], 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 **??**).

In general, this approach replaces the verifier's challenge (See 0.5) 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 [11]. We refer to [5], 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 [12].

1.8.5 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 [44] and have been used as confirmation and denial proofs for undeniable signature schemes.

Definition 12 (**Designated Verifier NIZK**[46])**.** *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 12 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 [13].

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 $R = u^n \mod n^2$ key $= (\mathsf{pk}, \mathsf{vk}) = (E = R^e, e)$.

Prover steps:

$T' = (1+n)^t R^{\mathsf{r}} \mod n^2, X' = (1+n)^x E^{-\mathsf{r}} \mod n^2$ 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 $d \bmod n$ from D' and checks that $D = g^d$ the verifier accepts if and only if, both checks succeeded.

1.8.5.1 Implicit Zero-Knowledge Proof Systems

Benhamoda *et al.*[8] 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 13 ([8])**.** *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 [8] for additional information.

1.8.6 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 [22, 45, 14] and Zero-Knowledge arguments with sub-linear communication based on generalised Pedersen commitments [36, 35].

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 [10, 49, 50, 4, 3, 41, 23]

1.9 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 0.8.3 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 [39] and continuing with Groth-Sahai proof techniques [40]. 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 [25, 9], 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 [40].

1.9.1 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 14 (**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* $\mathsf{gk}, \mathsf{crs}, x$ *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.
$$\Pr\left[\mathsf{Verify}(\mathsf{gk}, \mathsf{crs}, x, \pi) = \mathsf{accept} \;\middle|\; \begin{array}{c}(\mathsf{gk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^{\ell}) \\ (\mathsf{crs}) \leftarrow \mathsf{Kgen}(\mathsf{gk}, \mathsf{sk}) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{gk}, \mathsf{crs}, x, w)\end{array}\right] = 1 \tag{1.6}$$

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

$$\Pr\left[\mathsf{Verify}(\mathsf{gk},\mathsf{crs},x^*,\pi)=\mathsf{accept} \mid \begin{array}{c}(\mathsf{gk},\mathsf{sk})\leftarrow\mathsf{Setup}(1^\ell)\\(\mathsf{crs})\leftarrow\mathsf{Kgen}(\mathsf{gk},\mathsf{sk})\\(x^*,\pi)\leftarrow\mathcal{A}(\mathsf{gk},\mathsf{crs})\\x^*\notin\mathscr{L}\end{array}\right]=0 \tag{1.7}$$

If we consider the PPT adversary, we call the proof system with the computational soundness or an argument 0.3.5.

3. **Perfect Culpable Soundness:** In the original paper [40] 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:

$$\Pr\left[\mathsf{Verify}(\mathsf{gk},\mathsf{crs},x^*,\pi)=\mathsf{accept} \mid \begin{array}{c}(\mathsf{gk},\mathsf{sk})\leftarrow\mathsf{Setup}(1^\ell)\\(\mathsf{crs})\leftarrow\mathsf{Kgen}(\mathsf{gk},\mathsf{sk})\\(x^*,\pi)\leftarrow\mathcal{A}(\mathsf{gk},\mathsf{crs})\\x^*\notin\mathscr{L}_{\mathsf{guilt}}\end{array}\right]=0 \tag{1.8}$$

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

FIGURE 1.5: Witness Indistinguishability Experiment

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

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

and,

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

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

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

1.10 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 7) over bilinear groups.

1.10.1 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.

FIGURE 1.6: $\mathsf{Exp}^{\mathsf{sim-zk}}_{\mathcal{A}}(1^\ell)$

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

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 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:

$$\mathsf{Eq} : \mathbf{e}(\mathcal{X},\beta) \cdot \mathbf{e}(\alpha,\mathcal{Y}) = t_T, \tag{1.9}$$

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 1.7: 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:
$$\vec{\mathcal{X}} = (\mathcal{X}_1, \ldots, \mathcal{X}_m) \in \mathbb{G}_1^m \,,\ \vec{\mathcal{Y}} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_n) \in \mathbb{G}_2^n,$$
$$\vec{x} = (x_1, \ldots, x_{m'}) \in \mathbb{Z}_n^{m'} \,,\ \vec{y} = (y_1, \ldots, y_{n'}) \in \mathbb{Z}_n^{n'}$$

- 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 $\mathbb{G}_1$:
$$\sum_{i=1}^{n'} y_i \mathcal{A}_i + \sum_{i=1}^{m} \sum_{j=1}^{n'} \gamma_{ij} y_j \mathcal{X}_i + \sum_{i=1}^{m} b_i \mathcal{X}_i = \mathcal{T}$$
$$(\vec{\mathcal{A}} \cdot \vec{y}) + (\vec{\mathcal{X}} . \Gamma \vec{\mathcal{Y}}) + (\vec{\mathcal{X}} . \vec{b}) = \mathcal{T}$$

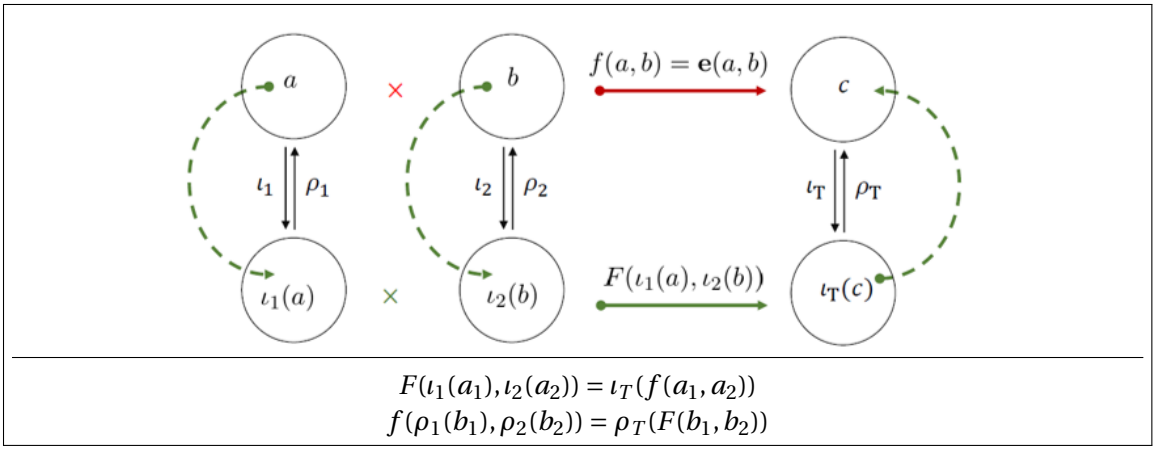
- $\mathsf{Eq}^{\mathsf{qe}}$: Quadratic equation in $\mathbb{Z}_n$:
$$\sum_{i=1}^{n} a_i y_i) + \sum_{i=1}^{m'} \sum_{j=1}^{n'} \gamma_{ij} x_i y_j + \sum_{i=1}^{m'} x_i b_i = t \mod n$$
$$(\vec{a} \cdot \vec{y})(\vec{x} \cdot \Gamma \vec{y}) + \vec{x} \cdot \vec{b} = t \mod n$$

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

$$\begin{aligned}\mathrm{R}_{GS} = \{(x, w) :& \\ & x = \mathsf{Eq} \in \mathcal{GS}^{\mathsf{Eq}}, \\ & w = (\{\alpha_i\} : \alpha_i \in (\mathbb{G}_1 \cup \mathbb{G}_2, \mathbb{G}_T) \\ & \mathsf{Eq}[w] = \mathsf{Eq}[\{g_i\}] = \mathsf{True} \\ & \}\end{aligned}$$

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



1.10.2 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 [40].

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:

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

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 $x_1, \dots x_m \in \mathbb{A}_1$ and $y_1, \dots, y_n \in \mathbb{A}_2$ 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 $a_1, \dots a_n \in \mathbb{A}_1, b_1, \dots b_m \in \mathbb{A}_2$ and $\gamma_{ij} \in \mathcal{R}$.

In order to avoid the heavy notation, we define

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

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 $u_1,\ldots,u_{\hat{m}} \xleftarrow{\$} \mathbb{B}$ and we let U be the space generated by elements $u_1,\ldots,u_n$ *i.e.*, $U = \langle u_1,\ldots,u_{\hat{m}}\rangle$. 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 $\mathsf{r}_1,\ldots,\mathsf{r}_{\hat{m}} \xleftarrow{\$} \mathcal{R}$ 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

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

we will write

$$\vec{c} = \iota_1(\vec{x}) + R\vec{u} \tag{1.10}$$

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

$$\begin{aligned}\forall i : \rho(u_i) = 0 \implies \rho(\mathsf{Com}) &= \rho(\iota(x) + \sum_{i=1}^{\hat{m}} \mathsf{r}_i u_i) = \\ &= \rho(\iota(x)) + \sum_{i=1}^{\hat{m}} \mathsf{r}_i \underbrace{\rho(u_i)}_{0} \\ &= \rho(\iota(x))\end{aligned} \tag{1.11}$$

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

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

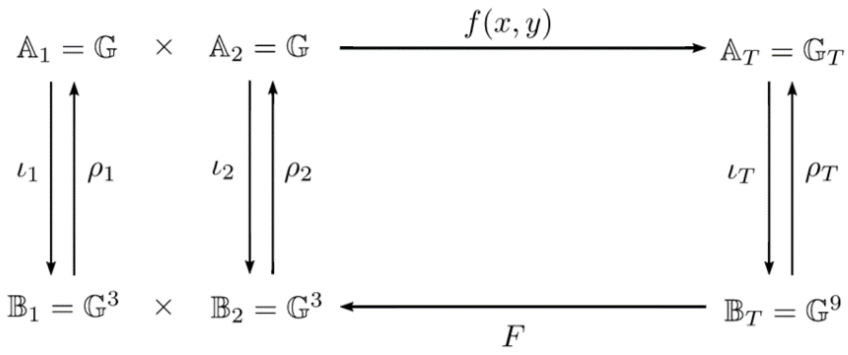
$$\begin{aligned}x \in \mathbb{A} \ \exists \alpha_i \in \mathcal{R} : \iota(x) = \sum_{i=1}^{\hat{m}} \alpha_i u_i \implies \mathsf{Com} &= \rho(\iota(x) + \sum_{i=1}^{\hat{m}} \mathsf{r}_i u_i) = \\ &= \sum_{i=1}^{\hat{m}} \alpha_i u_i + \sum_{i=1}^{\hat{m}} \mathsf{r}_i(u_i) \\ &= \sum_{i=1}^{\hat{m}} (\alpha_i + \mathsf{r}_i) u_i\end{aligned} \tag{1.12}$$

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

1.10.3 Groth-Sahai NIWI Proofs

We now outline how to prove the satisfiability of pairing product equations and in Section 0.11 we briefly present the Groth-Sahai proof system under the Subgroup Decision assumption and DLIN assumptions for pairing product equations.

FIGURE 1.9: Commutative Diagram for Groth-Sahai Proofs in DLin setting



1.10.4 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 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:

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

and also $H_1, \ldots, H_k$ 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 13 that has variables $\vec{x}$ and $\vec{y}$ from groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

$$\vec{a}\cdot\vec{y} + \vec{x}\cdot\Gamma\cdot\vec{y} + \vec{x}\cdot\vec{b} = t \tag{1.13}$$

Prover and the verifier perform the following computations:

- **Prover:** Choose random matrix $T \xleftarrow{\$} Mat_{\hat{n}\times\hat{m}}(\mathcal{R})$ and $\mathsf{r}_1, \ldots, \mathsf{r}_\eta \xleftarrow{\$} \mathcal{R}$. 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 [40].

1.11 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 **??**).

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 $\mathbb{G}$:

$$\mathcal{R} = \mathbb{Z}_p,\ \mathbb{A}_1 = \mathbb{Z}_p, \mathbb{A}_2 = \mathbb{G},\ \mathbb{A}_T = \mathbb{G},\ f(x,\mathcal{Y}) = x\mathcal{Y} :\ \vec{a}\cdot\vec{\mathcal{Y}} + \vec{x}\vec{\mathcal{B}} + \vec{x}\cdot\Gamma\vec{\mathcal{Y}} = \mathcal{T} \tag{1.14}$$

  Quadratic equations:

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

- **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:
$$\iota : \mathbb{G} \mapsto \mathbb{G}^3; \qquad \iota(Y) = (0, 0, Y),$$
$$\rho : \mathbb{G}^3 \mapsto \mathbb{G}; \qquad \rho(Z_1, Z_2, Z_3) = Z_3 \cdot Z_1^{-\frac{1}{\alpha}} \cdot Z_2^{\frac{1}{\beta}}.$$
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 **??**) 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:

$$\begin{aligned}&\text{For exponents:} && x, y \in \mathbb{Z}_p : f(x,y) = x \cdot y \mod p\\ &\text{For group elements:} && x, y \in \mathbb{Z}_p : f(x,y) = \mathbf{e}(x,y)\end{aligned}$$

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:

$$\iota_T(z) : \mathbb{Z}_p \mapsto \mathbb{G}^9, \quad \iota_T(z) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & z \end{pmatrix}$$

$$\rho_T : \mathbb{G}^9 \mapsto \mathbb{Z}_p, \quad \rho_T(\begin{pmatrix} z_{11} & z_{12} & z_{13} \\ z_{21} & z_{22} & z_{23} \\ z_{31} & z_{32} & z_{33} \end{pmatrix}) = z_{33} z_{13}^{-\frac{1}{\alpha}} z_{23}^{-\frac{1}{\beta}} \left(z_{31} z_{11}^{-\frac{1}{\alpha}} z_{21}^{-\frac{1}{\beta}} \right)^{-\frac{1}{\alpha}} \left(z_{32} z_{12}^{-\frac{1}{\alpha}} z_{22}^{-\frac{1}{\beta}} \right)^{-\frac{1}{\beta}}$$

     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:

$$\begin{aligned}&\iota_T^{\mathsf{ms}}(\mathcal{Z}) : \mathbb{Z}_p \mapsto \mathbb{G}^9, && \iota_T^{\mathsf{ms}}(\mathcal{Z}) = \tilde{F}(\iota'(1), \iota(\mathcal{Z})) = \tilde{F}(u, (\mathcal{O}, \mathcal{O}, \mathcal{Z}))\\ &\rho_T^{\mathsf{ms}} : \mathbb{G}^9 \mapsto \mathbb{Z}_p, && \rho_T^{\mathsf{ms}} = \mathbf{e}^{-1}(\rho_T(z)) \text{ where } \mathbf{e}^{-1}(\mathbf{e}(g, \mathcal{Z})) := \mathcal{Z}.\end{aligned}$$

     In the soundness setting $\rho_T^{\mathsf{ms}} \cdot \iota_T^{\mathsf{ms}} = \mathbb{I}_{\mathbb{G}}$

  3. Quadratic Equation in $\mathbb{Z}_p$

$$\begin{aligned}&\tilde{\iota_T^{\mathsf{q}}}(z) : \mathbb{Z}_p \mapsto \mathbb{G}^9, && \iota_T^{\mathsf{q}}(\mathcal{Z}) = \tilde{F}(\iota'(1), \iota'(z))\\ &\iota_T^{\mathsf{q}}(z) : \mathbb{Z}_p \mapsto \mathbb{G}^9, && \iota_T^{\tilde{z}.\mathsf{q}}(\mathcal{Z}) = \tilde{F}(\iota'(1), \iota'(z))\\ &\rho_T^{\mathsf{q}} : \mathbb{G}^9 \mapsto \mathbb{Z}_p, && \rho_T^{\mathsf{z}.\mathsf{q}} = \log_{\mathbf{e}(g,g)}(\rho_T(z)).\end{aligned}$$

     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 $\mathsf{crs} := (u_1, u_2, u_3)$ for random integers $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^*$ and $r, s \xleftarrow{\$} \mathbb{Z}_p$

$$u_1 = (A = g^\alpha, \mathcal{O}, g), u_2 = (\mathcal{O}, B = g^\beta, g), u_3 = (A^r, B^s, g^{r+s-1}) = r\,u_1 + s u_2$$

  3. **Witness Indistinguishability String:** On input gk return $\mathsf{crs} := (u_1, u_2, u_3)$ for random integers $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^*$ and $r, s \xleftarrow{\$} \mathbb{Z}_p$

$$u_1 = (A = g^\alpha, \mathcal{O}, g), u_2 = (\mathcal{O}, B = g^\beta, g), u_3 = (A^r, B^s, g^{r+s}) = r\,u_1 + s u_2 - (\mathcal{O}, \mathcal{O}, g)$$

- **Prover**: On input $x = \mathsf{gk}, \mathsf{crs}, \mathsf{E}$ 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 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 $(\mathsf{gk}, \mathsf{crs})$, $\vec{c}, \vec{d}$ 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 1.11.1. *[40] 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 [18] 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 **??**, the one that we will use in section **??**.

1.12 OR Statements

[1]Some of our relations of Section **??** 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:

$$\begin{aligned}\mathrm{R}_{OR} = \{(x,w) |\ & x = (\mathsf{E}_1, \mathsf{E}_2), w = (\mathsf{idx}, w_1, w_2) : \mathsf{idx} \in \{1,2\} \wedge \\ & (\mathsf{E}_{\mathsf{idx}}, w_{\mathsf{idx}}) \in \mathrm{R}_E \ \wedge \ w_{\bar{\mathsf{idx}}} \in \mathbb{G}^3\},\end{aligned}$$

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 [37] and we will follow a different approach.

By hypothesis, $\mathcal{P}_{\mathbb{GS}}$ takes as input a system of equations E as a statement and a solution $(g_1, \ldots, g_m)$ 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. $\mathsf{E}_{\mathsf{OR}} \approx \mathrm{R}_{OR}$. With this notation, we mean that there are two efficiently computable functions f and g such that:

$$\exists w = (\mathsf{idx}, w_1, w_2) \left(x = (\mathsf{E}_1, \mathsf{E}_2), w\right) \in \mathrm{R}_{OR} \Leftrightarrow \exists \tilde{w} \left(\mathsf{E}_{\mathsf{OR}} = f(x), \tilde{w}\right) \in \mathrm{R}_E.$$

$$\left(x, w\right) \in \mathrm{R}_{OR} \Rightarrow \left(f(x), g(x,w)\right) \in \mathrm{R}_{OR}.$$

   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 E_{OR} with the properties above. Consider two systems of pairing product equations E_1 and E_2 - same structure as in 7. For simplicity, we assume the equations are over two variables (the general case is straightforward).

$$\mathsf{E}_1 : \mathbf{e}(\mathcal{X}_1, a_1) \cdot \mathbf{e}(\mathcal{X}_2, a_2) = \tau_1 \,, \mathsf{E}_2 : \mathbf{e}(\mathcal{Y}_1, b_1) \cdot \mathbf{e}(\mathcal{Y}_2, b_2) = \tau_2$$

[1]This section is part of our publication in [53]

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

$$(\mathcal{Z}_{\mathsf{idx}} \hookleftarrow g_{\mathsf{idx}}, \mathcal{X}_1 \hookleftarrow g_1, \mathcal{X}_2 \hookleftarrow g_2, \mathcal{Y}_1 \hookleftarrow g_3, \mathcal{Y}_2 \hookleftarrow g_4, \mathcal{Z}_{11} \hookleftarrow g_{11}, \ldots, \mathcal{Z}_{22} \hookleftarrow g_{22})$$

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 $z_{22} = 2 - \mathsf{idx}$.
- $\mathbf{e}(\mathcal{Z}_{11}, \mathcal{Z}_{22}) = 1 \Rightarrow (z_{11} = 0 \vee z_{22} = 0)$
- $z_{11} = 0 \wedge z_{11} = 1 - \mathsf{idx} \Rightarrow \mathbf{e}(\mathcal{X}_1 \hookleftarrow g_1, a_1) \cdot \mathbf{e}(\mathcal{X}_2 \hookleftarrow g_2, a_2) = \tau_1$
  $\Rightarrow (\mathsf{E}_1[g_1, g_2] = \mathsf{True} \wedge \mathsf{idx} = 1)$
- Similarly, $z_{22} = 0 \wedge z_{22} = 2 - \mathsf{idx}$
  $\Rightarrow \mathbf{e}(\mathcal{Z}_{21}, \mathcal{Z}_{22}) = 1 \Rightarrow (\mathsf{E}_2[g_3, g_4] = \mathsf{True} \wedge \mathsf{idx} = 2)$

The above facts imply that:

$$\mathsf{E}_{\mathsf{OR}}[(g_{\mathsf{idx}}, g_1, \ldots, g_4, g_{11}, \ldots, g_{22})] = \mathsf{True} \Rightarrow$$
$$\Big((\mathsf{E}_1[g_1, g_2, \alpha_1] = \mathsf{True} \wedge \mathsf{idx} = 1\Big) \vee \Big(\mathsf{E}_2[g_3, g_4, \alpha_2] = \mathsf{True} \wedge \mathsf{idx} = 2)\Big),$$

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:

$$\mathsf{E}_{\mathsf{OR}}: \begin{cases} \mathbf{e}(\mathcal{X}_1, a_1) \cdot \mathbf{e}(\mathcal{X}_1, a_2) \cdot \mathbf{e}(\mathcal{Z}_{11}, \mathcal{Z}_{12}) = \tau_1 \\ \mathbf{e}(\mathcal{X}_1, a_1') \cdot \mathbf{e}(\mathcal{X}_2, a_2') \cdot \mathbf{e}(\mathcal{Z}_{11}, \mathcal{Z}_{13}) = \tau_1' \\ \mathbf{e}(\mathcal{Y}_1, b_1) \cdot \mathbf{e}(\mathcal{Y}_2, b_2) \cdot \mathbf{e}(\mathcal{Z}_{21}, \mathcal{Z}_{22}) = \tau_2 \\ \mathbf{e}(\mathcal{Y}_1, b_1') \cdot \mathbf{e}(\mathcal{Y}_2, b_2') \cdot \mathbf{e}(\mathcal{Z}_{23}, \mathcal{Z}_{22}) = \tau_2' \\ \mathbf{e}(\mathcal{Z}_{11}, \mathcal{Z}_{22}) = 1 \\ \mathbf{e}(\mathcal{Z}_{11}, g) \cdot \mathbf{e}(\mathcal{Z}_{\mathsf{idx}}, g) = \mathbf{e}(g, g) \\ \mathbf{e}(\mathcal{Z}_{22}, g) \cdot \mathbf{e}(\mathcal{Z}_{\mathsf{idx}}, g) = \mathbf{e}(g^2, g) \end{cases}$$

Bibliography

[1] Boaz Barak. “How to Go Beyond the Black-Box Simulation Barrier”. In: *FOCS*. 2001, pp. 106–115. DOI: 10.1109/SFCS.2001.959885.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] Mihir Bellare and Moti Yung. “Certifying Permutations: Noninteractive Zero-Knowledge Based on Any Trapdoor Permutation”. In: *J. Cryptol.* 9.3 (1996), pp. 149–166.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

Appendix

Appendix .A

Proof of Proposition ??

Prposition 3. If the DBDH assumption holds relative to GroupGen, then H_1 and H_2 are computationally indistinguishable.

Proof. The simulator $\mathbb{B}$ takes as input $(g, A = g^{\alpha}, B = g^{\beta}, C = g^{\tau}, Z \stackrel{?}{=} \mathbf{e}(g, g^{\alpha\beta\tau}))$ 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

$$\Omega, k, \tilde{a}, \delta_b, \theta_b, w_{1,i}, t_{1,i}, \tilde{f}_{b,i}, \tilde{h}_{b,i} \xleftarrow{\$} \mathbb{Z}_p^{\star}$$

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:

$$\Omega = \delta_1 w_{2,i} - \delta_2 w_{1,i} = \theta_1 t_{2,i} - \theta_2 t_{1,i}.$$

$\mathbb{B}$ 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 $\tilde{\lambda}_1, \tilde{\lambda}_2, r_i, \Phi_i \xleftarrow{\$} \mathbb{Z}_p$, 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 $i \in [n]$)

$$\begin{aligned}
& \mathsf{ct}_1 = C^{\tilde{s}_2} = g^{\tau \tilde{s}_2}, \mathsf{ct}_1' = C^{\tilde{s}_2'} = g^{\tau \tilde{s}_2'}, \mathsf{ct}_2 = h^{s_1}, \mathsf{ct}_2' = h^{s_1'} \\
& \mathsf{ct}_{3,i} = g^{s_1 \tilde{w}_{1,i}} \cdot g^{\tilde{s}_2 \tilde{f}_{1,i}} \cdot B^{\tilde{\delta}_1 x_i \tilde{s}_2} \cdot C^{\tilde{f}_{1,i}} \cdot g^{\tilde{\delta}_1 x_i \tilde{s}_3}, \; \mathsf{ct}_{3,i}' = g^{s_1' \tilde{w}_{1,i}} \cdot C^{\tilde{s}_2' \tilde{f}_{1,i}} \cdot B^{\tilde{\delta}_1 x_i \tilde{s}_3'} \\
& \mathsf{ct}_{4,i} = g^{s_1 \tilde{w}_{2,i}} \cdot C^{\tilde{s}_2 \tilde{f}_{2,i}} \cdot B^{\tilde{\delta}_2 x_i \tilde{s}_3}, \mathsf{ct}_{4,i}' = g^{s_1' \tilde{w}_{2,i}} \cdot C^{\tilde{s}_2' \tilde{f}_{2,i}} \cdot B^{\tilde{\delta}_2 x_i \tilde{s}_3'} \\
& \mathsf{ct}_{5,i} = g^{s_1 \tilde{t}_{1,i}} \cdot C^{\tilde{s}_2 \tilde{h}_{1,i}} Z^{\tilde{\theta}_1 x_i}, \mathsf{ct}_{5,i}' = g^{s_1' \tilde{t}_{1,i}} \cdot C^{\tilde{s}_2' \tilde{h}_{1,i}} Z^{\tilde{\theta}_1 x_i} \\
& \mathsf{ct}_{6,i} = g^{s_1 \tilde{t}_{2,i}} \cdot C^{\tilde{s}_2 \tilde{h}_{2,i}} Z^{\tilde{\theta}_2 x_i}, \mathsf{ct}_{6,i}' = g^{s_1' \tilde{t}_{2,i}} \cdot C^{\tilde{s}_2' \tilde{h}_{2,i}} Z^{\tilde{\theta}_2 x_i} \\
& \mathsf{ct}_8 = Z^{\tilde{\Omega}} \cdot \mathbf{e}(A, C)^{-\tilde{a}} \cdot \Lambda^{-\tilde{s}_2} \cdot m_0, \mathsf{ct}_8' = Z^{\tilde{\Omega}} \cdot \mathbf{e}(A, C)^{-\tilde{a}} \cdot \Lambda^{-\tilde{s}_2'} \cdot m_0
\end{aligned}$$

$$\mathsf{ct}_{3,i} = W_{1,i}^{s_1} \cdot F_{1,i}^{s_2} \cdot \delta_1^{x_i s_3} = g^{s_1 w_{1,i}} \cdot g^{f_{1,i}(\tilde{s}_2 + \gamma)} \cdot g^{x_i \delta_1 (-\gamma\beta + \tilde{s}_3)} = g^{s_1 w_{1,i}} \cdot g^{f_{1,i} \tilde{s}_2} g^{\gamma (f_{1,i} - x_i \delta_1 \beta)} \cdot g^{x_i \delta_1 \tilde{s}_3)}$$

$$= W_{1,i}^{s_1} \cdot g^{\tilde{s}_2(\beta\tilde{\delta}_1 x_i + \tilde{f}_{1,i})} \cdot g^{\gamma(f_{1,i} - x_i\tilde{\delta}_1\beta)} \cdot g^{\tilde{\delta}_1 x_i \tilde{s}_3} = W_{1,i}^{s_1} \cdot B^{\tilde{s}_2\delta_1 x_i} \cdot g^{\tilde{s}_2\tilde{f}_{1,i}} \cdot C^{\tilde{f}_{1,i}} \cdot g^{\delta_1 x_i \tilde{s}_3}$$

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,

$$\Lambda^{-s_2} = \Lambda^{-\tau-\tilde{s}_2} = (\mathbf{e}(A,B)^{-\Omega} \cdot \mathbf{e}(A,g)^{\tilde{a}})^{-\tau} \cdot \Lambda^{-\tilde{s}_2} = \mathbf{e}(h,g)^{\alpha\beta\tau} \cdot \mathbf{e}(A,C)^{-\tilde{a}} \cdot \Lambda^{-\tilde{s}_2},$$
$$\mathsf{ct}_8 = Z^{\Omega} \cdot \mathbf{e}(A,C)^{-\tilde{a}} \cdot \Lambda^{-\tilde{s}_2} \cdot m_0 = Z^{\Omega} \cdot \Lambda^{-s_2} \cdot \mathbf{e}(h,g)^{-\alpha\beta\tau} \cdot m_0 = Z^{\Omega}\mathbf{e}(h, g^{-\alpha\beta\tau}) \cdot \Lambda^{-s_2} \cdot m_0$$

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 1

Proposition 1. *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:

$$\{W_{b,i} = B^{\delta_b x_i} A^{\tilde{w}_{b,i}}, F_{b,i} = g^{f_{b,i}}, T_{b,i} = B^{\theta_b x_i} A^{\tilde{t}_{b,i}}, H_{b,i} = B^{\theta_b x_i} g^{\tilde{h}_{b,i}}\}_{b\in[2], i\in[n]}$$
$$\{U_b = g^{\delta_b}, V_b = g^{\theta_b}\}_{b\in[2]}, , h = A^{\tilde{\Omega}}, \Lambda = \mathbf{e}(g,g'), K_1 = g^k,\ K_2 = g'^{\frac{1}{k}}.$$

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 $\tilde{\lambda}_1, \tilde{\lambda}_2, \{\tilde{r}_i, \tilde{\Phi}_i\}_{i\in[n]} \xleftarrow{\$} \mathbb{Z}_p^\star$, 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:

$$K_{5,i}^{-h_{1,i}} K_{6,i}^{-h_{2,i}} = g^{h_{1,i}\theta_2 \tilde{\Phi}_i} g^{-h_{1,i}\tilde{\lambda}_2 v_i \tilde{t}_{2,i}} g^{-h_{2,i}\theta_1 \tilde{\Phi}_i} g^{h_{2,i}\tilde{\lambda}_2 v_i \tilde{t}_{1,i}}$$

$$= H_{1,i}^{\theta_2 \tilde{\Phi}_i} \cdot H_{1,i}^{-\tilde{\lambda}_2 v_i \tilde{t}_{2,i}} H_{2,i}^{-\theta_1 \tilde{\Phi}_i} \cdot H_{2,i}^{\tilde{\lambda}_2 v_i \tilde{t}_{1,i}} \Rightarrow K_{5,i}^{-h_{1,i}} K_{6,i}^{-h_{2,i}} \text{ is computable}$$

This shows that the simulator can compute the token as the real challenger.

Challenge Phase: To generate the ciphertext, $\mathbb{B}$ chooses random elements

$$\tilde{s}_1, \ldots, \tilde{s}_3, \tilde{s}'_1, \ldots, \tilde{s}'_3 \xleftarrow{\$} \mathbb{Z}_p^\star : \tilde{s}_3 \neq \tilde{s}'_3$$

and computes the challenge ciphertext as follows:

- $\mathsf{ct}_1 = C \cdot g^{\tilde{s}_2} = g^{\tau + \tilde{s}_2} \Rightarrow s_2 = \tau + \tilde{s}_2$,
- $\mathsf{ct}'_1 = C \cdot g^{\tilde{s}'_2} = g^{\tau + \tilde{s}'_2} \Rightarrow s'_2 = \tau + \tilde{s}_2$
- $\mathsf{ct}_2 = D^{\tilde{\Omega}} \cdot A^{\tilde{\Omega}\tilde{s}_1} = (g^{\alpha\tilde{\Omega}})^{(\eta + \tilde{s}_1)} = h^{\eta + \tilde{s}_1} \Rightarrow s_1 = \eta + \tilde{s}_1$
- $\mathsf{ct}'_2 = D^{\tilde{\Omega}} \cdot A^{\tilde{\Omega}\tilde{s}'_1} \Rightarrow s'_1 = \eta + \tilde{s}'_1$
- $\mathsf{ct}_{3,i} = W_{1,i}^{\tilde{s}_1} \cdot F_{1,i}^{\tilde{s}_2} \cdot U_1^{\tilde{s}_3 x_i} \cdot D^{\tilde{w}_{1,i}} \cdot C^{f_{1,i}} = W_{1,i}^{\tilde{s}_1} \cdot F_{1,i}^{\tilde{s}_2 + \tau} \cdot U_1^{\tilde{s}_3 x_i} \cdot g^{\eta\alpha\tilde{w}_{1,i}} \cdot F_{1,i}^{\tau} =$

$= W_{1,i}^{\tilde{s}_1} \cdot F_{1,i}^{\tilde{s}_2 + \tau} \cdot U_1^{\tilde{s}_3 x_i} \cdot g^{\eta(w_{1,i} - \beta\delta_1 x_i)} = W_{1,i}^{\tilde{s}_1 + \eta} \cdot F_{1,i}^{\tilde{s}_2 + \tau} \cdot U_1^{(\tilde{s}_3 - \eta\beta)x_i} \Rightarrow s_3 = -\eta\beta + \tilde{s}_3$

$$\bullet\ \mathsf{ct}_{4,i} = W_{2,i}^{\tilde{s}_1} \cdot F_{2,i}^{\tilde{s}_2} \cdot U_2^{\tilde{s}_3 x_i} \cdot D^{\tilde{w}_{2,i}} \cdot C^{f_{2,i}}, (\text{ similar computation as } \mathsf{ct}_{3,i})$$

and,

$$\begin{aligned}
&\bullet\ \mathsf{ct}'_{3,i} = W_{1,i}^{\tilde{s}'_1} \cdot F_{1,i}^{\tilde{s}'_2} \cdot U_1^{\tilde{s}'_3 x_i} \cdot D^{\tilde{w}_{1,i}} \cdot C^{f_{1,i}} \\
&\bullet\ \mathsf{ct}'_{4,i} = W_{2,i}^{\tilde{s}'_1} \cdot F_{2,i}^{\tilde{s}'_2} \cdot U_2^{\tilde{s}'_3 x_i} \cdot D^{\tilde{w}_{2,i}} \cdot C^{f_{2,i}} \\
&\bullet\ \mathsf{ct}_{5,i} = T_{1,i}^{\tilde{s}_1} \cdot D^{\tilde{t}_{1,i}} \cdot H_{1,i}^{\tilde{s}_2} \cdot C^{\tilde{h}_{1,i}} \cdot Z^{\theta_1 x_i}, \\
&\bullet\ \mathsf{ct}'_{5,i} = T_{1,i}^{\tilde{s}'_1} \cdot D^{\tilde{t}_{1,i}} \cdot H_{1,i}^{\tilde{s}'_2} \cdot C^{\tilde{h}_{1,i}} \cdot Z^{k\theta_1 x_i} \\
&\bullet\ \mathsf{ct}_{6,i} = T_{2,i}^{\tilde{s}_1} \cdot D^{\tilde{t}_{2,i}} \cdot H_{2,i}^{\tilde{s}_2} \cdot C^{\tilde{h}_{2,i}} \cdot Z^{\theta_2 x_i}, \\
&\bullet\ \mathsf{ct}'_{6,i} = T_{2,i}^{\tilde{s}'_1} \cdot D^{\tilde{t}_{2,i}} \cdot H_{2,i}^{\tilde{s}'_2} \cdot C^{\tilde{h}_{2,i}} \cdot Z^{\theta_2 x_i}
\end{aligned}$$

Analysis the game: First, notice that:

$$\begin{aligned}
&D^{\tilde{t}_{1,i}} = g^{\eta\alpha\tilde{t}_{1,i}} = g^{\eta(t_{1,i}-\beta\theta_1 y_i)} = T_{1,i}^{\eta} \cdot g^{-\beta\eta\theta_1 y_i}, D^{\tilde{t}_{1,i}} = T_{1,i}^{\eta} \cdot g^{-k\beta\eta\theta_1 y_i} \\
&C^{\tilde{h}_{1,i}} = g^{\tau(h_{1,i}-\beta\theta_1 y_i)} = H_{1,i}^{\tau} \cdot g^{-\beta\tau\theta_1 y_i}, C^{\tilde{h}_{1,i}} = H_{1,i}^{\tau} \cdot g^{-\beta\tau\theta_1 y_i} \\
&\quad \Rightarrow \\
&\mathsf{ct}_{5,i} = T_{1,i}^{\tilde{s}_1} \cdot D^{\tilde{t}_{1,i}} \cdot H_{1,i}^{\tilde{s}_2} \cdot C^{\tilde{h}_{1,i}} \cdot Z^{\theta_1 y_i} = \\
&\qquad = T_{1,i}^{\tilde{s}_1} \cdot T_{1,i}^{\eta} \cdot g^{-\beta\eta\theta_1 y_i} \cdot H_{1,i}^{\tilde{s}_2} \cdot H_{1,i}^{\tau} \cdot g^{-\beta\tau\theta_1 y_i} \cdot Z^{\theta_1 \vec{y}_i} \\
&\qquad = T_{1,i}^{\eta+\tilde{s}_1} \cdot H_{1,i}^{\tau+\tilde{s}_2} \cdot (g^{-\beta(\tau+\eta)} \cdot Z)^{\theta_1 y_i} = T_{1,i}^{s_1} \cdot H_{1,i}^{s_2} \cdot (g^{-\beta(\tau+\eta)} \cdot Z)^{\theta_1 y_i} \\
&\mathsf{ct}'_{5,i} = T_{1,i}^{s'_1} \cdot H_{1,i}^{s'_2} \cdot (g^{-\beta(\tau+\eta)} \cdot Z)^{\theta_1 y_i}
\end{aligned}$$

$$\text{If } Z = g^{\beta(\eta+\tau)} \quad \Rightarrow \begin{cases} g^{-\beta(\tau+\eta)} \cdot Z = 1_{\mathbb{G}} \Rightarrow \mathsf{ct}_{5,i} = T_{1,i}^{s_1} \cdot H_{1,i}^{s_2} \\ g^{(-\beta(\tau+\eta)} \cdot Z = 1_{\mathbb{G}} \Rightarrow \mathsf{ct}_{5,i} = T_{1,i}^{s'_1} \cdot H_{1,i}^{s'_2} \end{cases}$$

$\Rightarrow$ 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 2

Proposition 2. *Under DLin assumption* H_3 *and* H_4 *are indistinguishable for all PPT adversaries* $\mathcal{A}$.

Proof. The simulator takes as input $(g, A = g^{\alpha}, B = g^{\beta}, C = g^{\tau}, D = g^{\alpha\eta}, Z \stackrel{?}{=} g^{\beta(\eta+\tau)})$ and by interacting with the adversary $\mathcal{A}$, distinguish between $g^{\beta(\eta+\tau)}$ and a random element.

SetUp phase. Generating master public key is same as in 1, 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 1.

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:

$$\begin{aligned}
\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} \\
&\text{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}} \\
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}$$

$$\begin{aligned}
K_B &= \prod_{i=1}^n g^{-(r_i+\Phi_i)} = \prod_{i=1}^n g^{-\tilde{r}_i + \frac{c_y\tilde{\lambda}_2 v_i x_i\beta}{\alpha} - \tilde{\Phi}_i - \frac{c_x\tilde{\lambda}_2 v_i y_i\beta}{\alpha}} = \prod_{i=1}^n g^{-(\tilde{r}_i+\tilde{\Phi}_i) + \frac{c_y\tilde{\lambda}_2\beta}{\alpha}(c_y v_i x_i - c_x v_i y_i)} = \\
&= g^{-\sum_{i=1}^n(\tilde{r}_i+\tilde{\Phi}_i)} g^{\frac{\tilde{\lambda}_2\beta}{\alpha}(c_x\sum_{i=1}^n v_i y_i - c_y\sum_{i=1}^n v_i x_i)} = \prod_{i=1}^n g^{-(\tilde{r}_i+\tilde{\Phi}_i)}\cdot g^{\frac{\tilde{\lambda}_2\beta}{\alpha}(c_y c_x - c_x c_y)} = \prod_{i=1}^n g^{-(\tilde{r}_i+\tilde{\Phi}_i)}
\end{aligned}$$

This shows that the simulator can compute the token as the real challenger.

Generating the challenge ciphertext. Simulator does the exact steps as in 1 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 $\mathrm{R}_{\mathsf{ballot}}$ defined in **??**:

Variables:

$$\mathcal{M} = g^{\mathsf{vote}}, \mathcal{M}_i = g^{\mathsf{vote}^i}, \mathcal{X} = g^{\mathsf{crd}}, \mathcal{A}_i = g^{\hat{a}^i}, \mathcal{CA}_i = \mathsf{CA}_i,$$
$$\hat{\mathcal{M}} = h^{\mathsf{r}_v}, \hat{\mathcal{M}}_i = h^{\mathsf{r}_i}, \hat{\mathcal{X}} = h^{\mathsf{r}_{\mathsf{crd}}}, \hat{\mathcal{A}}_i = h^{\mathsf{r}'_i}, \hat{\mathcal{CA}}_i = h^{\mathsf{r}^*_i}$$

Equation

$$\begin{aligned}
&\mathbf{e}(\mathsf{CT}_{\mathsf{vote}}, g) = \mathbf{e}(g^{\mathsf{vote}} \cdot h^{\mathsf{r}_v}, g) = \mathbf{e}(\mathcal{M}, g) \cdot \mathbf{e}(\hat{\mathcal{M}}, g) \\
&\mathbf{e}(\mathsf{CT}_{\mathsf{crd}}, g) = \mathbf{e}(g^{\mathsf{crd}} \cdot h^{\mathsf{r}_{\mathsf{crd}}}, g) = \mathbf{e}(\mathcal{X}, g) \cdot \mathbf{e}(\hat{\mathcal{X}}, g) \\
&\text{for } i = 1, \ldots, k: \\
&\mathbf{e}(\mathsf{CA}_i, g) = \mathbf{e}(g^{\hat{a}^i} \cdot h^{\mathsf{r}'_i}, g) = \mathbf{e}(\mathcal{A}_i, g) \cdot \mathbf{e}(\hat{\mathcal{A}}_i, g) \\
&\mathbf{e}(\mathsf{CA}_i, g) = \mathbf{e}(g^{\hat{a}^i}, g) = \mathbf{e}(g^{\hat{a}^{i-1}}, g^{\hat{a}}) = \mathbf{e}(\mathcal{A}_{i-1}, \mathcal{A}_1) \\
&\mathbf{e}(\mathsf{CA}^*_i, g) = \mathbf{e}(\mathsf{CA}_i \cdot h^{\mathsf{r}^*_i}, g) = \mathbf{e}(\mathcal{CA}_i, g) \cdot \mathbf{e}(\hat{\mathcal{CA}}_i, g) \\
&\mathbf{e}(\mathcal{M}_k, g_2) \cdot \mathbf{e}(\mathcal{M}_{k-1}, g_2)^{p_1} \cdot \ldots \cdot \mathbf{e}(\mathcal{M}_1, g_2)^{p_{k-1}} \cdot \mathbf{e}(g_1, g_2)^{p_k} = 1_{\mathbb{G}_T} \\
&\text{for } i = 1, 2, \ldots, m: \\
&\mathbf{e}(\mathcal{M}_i, g) \cdot \mathbf{e}(\mathcal{M}_{i-1}, \mathcal{M})^{-1} = 1_{\mathbb{G}_T}
\end{aligned} \tag{15}$$

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:

$$\mathsf{CT}_{\mathsf{vote}} = \mathsf{Enc}(\mathsf{vote}) : \mathsf{Poly}_C(\mathsf{vote}) = 0$$

Where the polynomial $\mathsf{Poly}_{\mathcal{C}}$ is defined as follows:

$$\mathsf{Poly}_{\mathcal{C}} = \prod_{i=1}^{m}(x - \mathsf{c}_i) = \sum_{i=0}^{m} c_i x^i$$