Revisiting Practical and Usable Coercion-Resistant Remote E-Voting [⋆]

Ehsan Estaji[1], Thomas Haines[2], Kristian Gjøsteen[2], Peter B. Rønne[1], Peter Y. A. Ryan[1], and Najmeh Soroush[1]

[1] SnT & University of Luxembourg, Luxembourg {firstname.lastname}@uni.lu
[2] Norwegian University of Science and Technology, Trondheim, Norway {firstname.lastname}@ntnu.no

Abstract. In this paper we revisit the seminal coercion-resistant e-voting protocol by Juels, Catalano and Jakobsson (JCJ) and in particular the attempts to make it usable and practical. In JCJ the user needs to handle cryptographic credentials and be able to fake these in case of coercion. In a series of three papers Neumann et al. analysed the usability of JCJ, and constructed and implemented a practical credential handling system using a smart card which unlock the true credential via a PIN code, respectively fake the credential via faking the PIN. We present several attacks and problems with the security of this protocol, especially an attack on coercion-resistance due to information leakage from the removal of duplicate ballots.
Another problem, already stressed but not solved by Neumann et al, is that PIN typos happen frequently and would invalidate the cast vote without the voter being able to detect this. We construct different protocols which repair these problems. Further, the smart card is a trusted component which can invalidate cast votes without detection and can be removed by a coercer to force abstention, i.e. presenting a single point of failure. Hence we choose to make the protocols hardware-flexible i.e. also allowing the credentials to be store by ordinary means, but still being PIN based and providing PIN error resilience. Finally, one of the protocols has a linear tally complexity to ensure an efficient scheme also with many voters.

Keywords: Electronic voting · coercion-resistance · usable security.

1 Introduction

One of the main threats in remote electronic voting is that they are inherently susceptible coercion-attacks due to the lack of a voting booth. In their seminal paper, Juels, Catalano and Jakobsson [10] gave a formal definition of coercion-resistance and further devised a protocol (JCJ) satisfying this strong security property. To achieve this, JCJ assumes a coercion-free setup phase where the voter get a credential which is essentially a cryptographic key. To cast a valid ballot this key needs to be entered correctly together with the vote. In case of coercion,

[⋆] This research were supported by the Luxembourg National Research Fund (FNR).

the voter can simply give a fake random credential to the coercer and even cast a vote together with the coercer using this fake credential – the corresponding vote will be removed in the tally process. The tally process of weeding out the ballots with fake credentials and duplicates, however, suffers from a quadratic complexity problem in the number of voters and cast ballots. Several paper are devoted to reduce the tally complexity in JCJ, see e.g. [18,2,6,20], however, each with their drawbacks. JCJ and similar constructions however also suffer from usability deficits, see also [14]. Especially, the voter intrinsically cannot directly check if a cast ballot is valid and will be counted, see however [8].

Moreover the handling and storing of long credentials is a notorious usability problem, getting even harder with a coercer present. The usability was analysed by Neumann et. al. [16,15,5] and led to a protocol using smart cards for handling voter's credentials. The stored credential is combined with a PIN code to produce the full credential which will be compared with the credential stored by the authorities on the bulletin board. In this paper we revisit this protocol and present several attacks on coercion-resistance and verifiability, but also possible repairs.

Whereas the the smart card provides a solution to the usability problem, it also comes with strong trust assumptions and problems

- The smart card is generally needs to be trusted. A malicious card could e.g. use the wrong credential invalidating the cast ballot without detection, and we cannot let the voter check if the ballot is correct without introducing coercion threats.
- The coercer can take the smart card away from the voter to force abstention.
- It is more expensive, less flexible and harder to update than a purely software solution.
- One of the attacks that we found is that a coercer can use the smart card to cast ballots on his own. This not only endangers coerced voter's real vote, but due to a leak of information in the weeding phase, the coercer can also detect, with non-negligible probability, whether the coerced voter has cast an independent ballot against his instructions.

In this paper we will present protocols that repair, or at least diminishes the attack probability of, the last point by constructing new duplicate removal methods in JCJ. Further, the protocols constructed in this paper are hardware-independent: they could use a smart card, or they can be implemented using combination of a digitally stored cryptographic length key and a PIN only known by the voter. The long credential could be stored in several places – or even hidden via steganography. At ballot casting time the software will take as input the digital key and the password to form the credential submitted with the vote. Depending on the level of coercion, the coerced voter can either fake the long credential or, for stronger levels of coercion, the voter can reveal the the digitally stored credential to the coercer, but fake the PIN. Due to our improved tally, the coercer will not know if he got faked credentails or PINs.

Another major problem with the original construction, already discussed as an open problem in [16], is the high chance of users doing a PIN typo error which will invalidate the vote and remain undetected. Note that naively giving feedback on the correctness of the PIN is not possible for coercion-resistance as it would allow the coercer to check whether he got a fake PIN or not. Instead, we will define a set of allowed PIN errors (e.g. chosen by the election administrator), and we will consider a ballot as valid both if it has a correct PIN or an allowed PIN error, but invalid for other PINs. We construct protocols which at tally time secretly check whether a given PIN is in the set of allowed PINs and will sort out invalid ballots. The protocols can accommodate general PIN error policies, however Wiseman et. al. [22] studied usual errors in PIN entries. Two frequent errors are transposition errors (i.e. entering "2134" instead of "1234") and wrong digit number errors (i.e. entering "1235" instead of "1234"). Correcting for both of these errors is however problematic, as we will see, since the set of independent PINs becomes small.

The outline of paper is as follows. In Section 2 we present attacks and problems of the orignal NV12 scheme. Our improved protocols are presented in Section 3. In Section 4 we make a preliminary analysis of how many independent PINs exist when allowing certain PIN errors. Finally we conclude in Section 5.

2 Analysis of NV12: Attacks and Problems

Neumann et al. [16] carried out a usability analysis of JCJ and proposed a new scheme (NV12) for handling the credentials and vote-casting. In [15] a few modification were made to prevent side-channel attacks and an efficiency analysis was done, and finally [5] presented a prototype implementation and its efficiency.

2.1 The scheme:

In this subsection we give a brief overview of the NV12 scheme, we refer to [15] and the JCJ/Civitas papers [10,4] for more details. The entities participating in the NV12 protocol are: **A supervisor:** who is in charge of running election and declaring election authorities; **The voter:** who intends to cast her vote; **The voter's smart card, reader and computer:** which serves as interface between the voter and the JCJ / Civitas system. The smart card reader has a screen and PIN entry interface; **A registrar:** who administrates the electoral register; **A supervised registration authority and a set of registration tellers:** that provide the voter with her credential;

A set of tabulation tellers: that are in change of the tallying process; **A set of ballot boxes:** to which voters cast their votes; **A bulletin board,** BB**:** that is used to publish information. The ballot boxes will publish to BB.

The framework of the scheme is as follows

1. **Setup Phase.** This step is the same as JCJ/ Civitas; an election public key, pk, will be computed and published.
2. **Registration Phase.** After offline and online registration phases, the voter's credential divided by the chosen PIN is stored on the smart card alongside with a designated verifier proof.

3. **Voting Phase.** The voting procedure is split into two phases implementing Benaloh challenges to the vote encryption
   - **Challenge:** The smart card commits to an encryption of the vote by displaying $\text{hash}\big(\mathsf{enc}(\mathsf{vote}, \mathsf{pk}, r)\big)$. The voter notes down this hash, and if the encryption is challenged, the smart card releases the randomness r to the voter's computer, and the voter can verify the hash indeed was consistent with the vote choice via a third device. This challenge procedure can be reiterated.
   - **Cast:** When the voter chooses to cast, she then enters the PIN. Now, the ballot of the form $\langle\{\mathsf{CRD}\}_{\mathsf{pk}}, \{vote\}_{\mathsf{pk}}, \sigma, \phi\rangle$ is generated where σ is a zero-knowledge proof (ZKP) of well-formedness of the vote and ϕ is a ZKP of knowledge of both the credential and vote. This is sent anonymously to a ballot box. $\text{hash}(\langle\{\mathsf{CRD}\}_{\mathsf{pk}}, \{vote\}_{\mathsf{pk}}, \sigma, \phi\rangle)$ is displayed and written down by the voter, and can be checked with the stored ballot in the ballot box to ensure stored-as-cast verifiability.
4. **Tallying Phase.** This step is also the same as JCJ/ Civitas.

The important trust assumptions made in [15] are

- For privacy it was assumed:
  - Half of the remote registration tellers and the supervised registration authority are trustworthy.
  - Neither the smart cards nor smart card readers can be corrupted.
  - The adversary is not able to corrupt a threshold set of tabulation tellers.
- For coercion-resistance we further need:
  - There is a point in the voting phase, in which the adversary cannot control the voter.
  - The adversary cannot control the voter's computer.
  - The channel to the ballot boxes is anonymous
- For verifiability it was assumed:
  - The adversary is not able to corrupt smart cards. With the Benaloh challenges implemented this was reduced further to [16]: The adversary cannot control the voting environment and the verification environment at the same time.

2.2 Attacks

We will now present attacks and discuss how to repair these.

Benaloh challenge problem: The first attack is on individual verifiability. The Benaloh challenge is available for the user to challenge whether the encryption of the vote is done honestly. The smart card and reader commits to the hash of the encryption via the screen of the smart card reader. The problem is that this hash is not checked for the cast ballot. Instead, what is checked for the cast ballot is that the hash of the full ballot including the encryption of the credential and ZKPs matches what is received in the ballot box. This means that the smart

card can at first encrypt all votes honestly and commit to these. However, when the PIN is entered to cast a ballot, it can encrypt its own vote choice and include this in the ballot without being detected even if the verification environment is honest – this violates the trust assumption above.
Repair: Both the hash of the vote encryption and the full ballot needs to be compared with the values that can be calculated from the ballot received by the ballot box. This however reduces usability as now two hashes needs to be checked by the voter, a task which is not trivial. Particularly, the adversary can precompute hashes that are hard to distinguish for the voter - e.g. matching on the leading part. Another choice is to commit to the full ballot in the Benaloh challenge, however this requires the voter to enter the PIN for each challenge. Since it is a general problem in e-voting that verification checks are too infrequent among real voters, having to enter a PIN for each challenge further undermines the Benaloh challenge security. It might also happen that a voter would then maximally challenge once, and hence an efficient strategy for the adversary would be to cheat after the first challenge.

Brute force attack: The second attack in on coercion-resistance for a coercer demanding access to the smart card, alternatively on verifiability for a local adversary who manages to get access to the smart card undetected. The adversary could here simply try to guess the PIN and cast a vote. This is not detectable by the voter due to anonymity of the vote casting. Unfortunately, the PIN space cannot be scaled since it is upper bounded by the ability of the voter to remember and enter PINs correctly. Hence, the probability of guessing the PIN is not negligible. Further, the probability can be boosted by casting multiple votes. Note also, whereas we can assume that it is in the interest of the voter to use a correct smart card reader, the adversary can use a malicously constructed reader. Thus the ballot casting can be automated and the PIN space can be covered to get a probability of a valid cast vote to be 1. This is not impossible, e.g. according to [5] vote casting took about 13 seconds including network time. The theoretical value with network was around 8 seconds, and the value of modern smart cards should be much lower. However, even with the 2014 timings, the creation of the ballots (without sending) could be done in 22 hours. Note that whether the ballot is counted in the end will depend on the vote update policy, and when the voter is casting her own vote, however, here the adversary is free to optimise his strategy, e.g. try to cast last.
Repair: The smart card could demand that a certain time has to pass between each ballot cast. This time can however not be too long, otherwise a coercer might detect it or utilise it for a forced abstention. Thus this repair can only lower the probability for casting a ballot with correct PIN.

Leaky duplicate removal: This is an attack on coercion-resistance, but can also be an attack on verifiability to boost the attack above. In the simplest form the coercer uses the smart card to cast a vote with some trial PIN. The coercer wants to determine if this trial PIN is a correct PIN. According to the protocol the voter will cast her true vote using the correct PIN at some secret point during the voting phase. However, in the tally phase credentials are weeded using plaintext equivalence tests (PETs) of the encrypted credentials directly on the submitted ballots.[3] If the coercer now sees an equivalence with his submitted trial ballot, he can guess that it was the voter casting the other ballot, and probably with the correct PIN. Thus he has determined the correct PIN and that the voter defied his instructions in one go. To boost the attack he can simply try several PINs.[4] In standard JCJ such an attack would not work since the submitted trial credential would have the same probability of being identical to the coerced voter's credential as for it to be identical to any other voter's credential, and further the probability would be negligible.

A local adversary getting access to the smart card could also follow this strategy to try to know the PIN and cast valid votes. This might actually be detected by the voter if he checks the weeding on BB and sees a duplicate of his own vote (note this was also mentioned in [17]), but in the protocol the voter is not instructed to do this. Thus the PIN is not really protecting against unauthorized use of the smart card.

Repair: It is actually surprisingly hard to make a tally protocol which does not leak information to prevent this attack. The original JCJ protocol relies on the fact that guessing the real full credential can only happen with negligible chance. A first repair could be to mix the ballots before doing weeding, but after verifying the ZKPs. This makes it difficult to implement certain policies, like the last valid vote counts; however, it fits nicely with the policy that a random selection from the valid votes count. Unfortunately, this does not prevent the attack. The coercer could mark his ballot by casting it a certain number of times which is likely to be unique. He then checks if he sees this number of duplicates or one more. Even if mix between each duplicate removal, which would be horrible for an efficiency perspective, we do not get a leak-free tally. The distribution of time until a PET reveals a duplicate will depend on whether the PIN was correct or not. Especially the coercer could cast a lot of votes with the same trial PIN which would make detecting this more visible. There are other methods to limit the the information leak in the tally which we will present below. Further, we will present a protocol that does not leak information about the number of duplicates per voter, and does have linear tally complexity (compared to the quadratic in JCJ), but which has an obfuscated form of participation privacy.

Fake election identifier: This is an attack on verifiability. As mentioned in the original JCJ paper, the zero-knowledge proofs need to include a unique election

[3] In general this is not good for coercion-resistance since a coercer might detect a voter not following instructions across elections, see [8].

[4] Note that the coercer does not have to let the voter know that he follows this strategy. The voter only knows that the coercer has access to the card for some short time. Based on this, she could also decide not to cast her true vote at all, but then the protocol could not really be called coercion-resistant since the coercer has a very efficient strategy to force abstention.

identifier. This identifier is announced by the election administrator and prevents that ballots are copied from one election to another, i.e. the proofs would not verify when the wrong identifier is used. However, the smart card needs to be updated with this identifier before vote casting. However, we cannot trust this is done correctly, i.e. an adversary e.g. controlling the voter's computer could try to provide a wrong credential.
Repair: The voter could enter the election identifier herself, but this is error prone. The simplest solution is that the voter checks that the submitted ballot has a zero-knowledge proof that verifies according to the real election identifier. This could be done when the hash of the full ballot is checked, but will mean that the voter has to wait a bit longer before being able to do this check.

Smart card removal: An obvious forced abstention attack is that the coercer simply demand to hold the smart card during the election period.
Repair: This problem seems quite inherent to the smart card approach. We could let the voter hold several smart cards. However, holding several cards would be physical evidence which a voter with a local coercer probably would not want to risk. Further, the number of cards allowed per voter could necessarily not be bounded. If each voter were allowed to hold e.g. 5 cards, the coercer would simply ask for five cards. If this is troublesome it seems better to leave the smartcard only approach and allow the voter to also hold the credential as a piece of data as in standard JCJ. This can more easily be hidden (steganography could be an option here) even though theoretically this also has problems [19]. Our protocols below can be implemented with or without smart cards.

2.3 Security Problems

In this section we discuss some problems with the protocol, that do not fall under the category of attacks.

The main usability and verifiability problem with the protocol is that PIN entry is error prone, as was already stressed in the papers by Neumann et al. An obvious solution is to have a PIN check, e.g. a checksum check. However, this would mean that only certain PINs are valid PINs, and in order for a voter to present a fake PIN to a coercer, she would first have to prepare a valid fake PIN, which is less usable.

An option with higher usability is to have a policy of allowed PIN errors and accept full credentials that corresponds to the PIN being entered with allowed errors. This is the approach we will essentially follow in this paper, however our solutions will also work for checksum checks.

If JCJ had a method of verifying the cast votes, we would also be able to at least detect such PIN errors. Such a verification mechanism was suggested in [8] using the Selene approach. However, this check can only be made after vote casting has ended, thus too late to update a PIN typo.

Another problem is the assumption that the smart card is trustworthy. This does not seem like a valid assumption, at least for important election. The smart card could simply use a wrong credential in a ballot, which would invalidate the vote. Further, this cannot be detected since the smart card is the only holder of the credential. At least the encryption of the PIN could be Benaloh tested, but not the credential. Further, the smart card reader is also trusted. However, this might not be enough in practice. As an example, if the middleware on the reader allows the voter's computer or the network to display messages on the screen, e.g. to say it is waiting for a connection, then it could e.g. try to display fake hash values. A corrupted smart card could also easily break privacy by using the encryption choice as a subliminal channel for the vote choice. In light of this the smartcard can also be seen as *a single point of failure*. We will thus focus on hardware-independent protocols.

3 Protocol Description

In this section we will present two protocols which tolerate PIN errors and prevents leak of information in the deduplication phase.

In our voting scenario the voter has two keys: a long key which is stored on her device (smart card or another device) and a short PIN, which is memorized.

To efficiently evaluate whether a PIN is allowed we will use polynomial evaluation. To this end, given a user's PIN a, we generate an $\mathsf{ErrorList}_a = \{a_1 = a, a_2, \ldots, a_k\}$ of allowed PINs. Note the number of PINs here is constant for every voter and might contain duplicates. From this, we generate a polynomial, $\mathsf{poly}_{\mathrm{PIN}}(x) = \prod_{i=1}^{k}(x - a_i) = \sum_{i=0}^{k} p_i x^i$ which has all $\mathsf{ErrorList}_a$ members as its root. In order to check the validity of the PIN, typed by the voter, it is then sufficient check whether the polynomial value on this PIN is equal to zero or not.[5] It is obvious that this polynomial should kept secret otherwise an adversary can recover the PIN by factorizing the polynomial. Therefore we have to work with encrypted polynomials and a main challenge is the polynomial evaluation under this encryption. Assume we have $\mathsf{Enc}(\mathsf{poly}_{\mathrm{PIN}}(x)) = \sum_{i=0}^{k} \mathsf{cp}_i x^i$ and $\mathsf{CT}_{\mathrm{PIN}} = \mathsf{Enc}(\hat{a})$, we need to find a way to efficiently compute $\mathsf{Enc}(\mathsf{poly}_{\mathrm{PIN}}(\hat{a}))$.

The next challenge is to find a way to prove publicly that the individual voter's polynomial are correctly evaluated without endangering the coercion-resistance. This would e.g. rule out voters evaluating the polynomials on voter side only.

Further, while solving this problem, we will also focus on efficient protocols to obtain a practical JCJ scheme with (almost) linear tally time in the number of voters.To obtain this we need to sacrifice perfect privacy. In the first scheme we only have participation privacy by obfuscation inspired by [6,11]. Here ballots are submitted with an ID and homomorphic Paillier encryption can then be used to evaluate the polynomial. Everybody, e.g. also a separate authority, can cast votes labelled with ID which will later be discarded as invalid. Thus the actual participation of the voter is obfuscated and the voter can deny having participated in the election. Optionally, we could also follow the JCJ alternative

[5] Note there is a small problem here since we are in composite order groups and the polynomials might have more roots than the allowed PINs. However, the probability in general is negligible.

method in [6] to achieve perfect privacy, however the cost will be that the voters twice have to defy the coercer and interact with the voting system. In the second scheme using BGN encryption, the information leak from duplicate removal will not be negligible, but bounded, and this scheme does not satisfy linear tally efficiency.

Due to space limitations, we will just explain the basic building blocks and their algorithm and suppress some details about ballot integrity and non-malleability from the zero-knowledge proofs, e.g. the inclusion of election identifiers and the correct form of the Fiat-Shamir transformations. Also, for simplicity, we describe the protocol with a single trusted party, but it is possible to distributively run this protocol. We will also not specify all parts of the distributed registration phase and the Benaloh challenges, this can be implemented as in the NV12 scheme with some obvious modifications and with the repairs mentioned above.

3.1 Paillier Instantiation

The first instantiation relies on the Paillier public-key cryptosystem which is a partially homomorphic and its security is based on the hardness of the decisional composite residuosity assumption. A ciphertext on message $m \in \mathbb{Z}_n$ has the form $\mathsf{CT} = (g^m \cdot r^n \mod n^2)$ which $n = pq$ and p, q are two same-length prime numbers, and g is a proper member of group $\mathbb{Z}^*_{n^2}$. Its homomorphic property allows us to evaluate the polynomial without decrypting the coefficients of the polynomials. Further it allows an efficient multi-party computation protocol to compare and (and hence sort) ciphertexts by plaintext values without decryption [13]. This algorithm is linear in the bit length, i.e. logarithmic in the security parameter, and can be made public verifiable [12]. Using this technique allows us to do the weeding process secure and efficient, but at the cost of all ballots being submitted with a voter identifier. To achieve participation privacy, obfuscating votes needs to be cast too.

eVoting Protocol with Paillier instantiation: In Set-Up phase, CA generates the pair of keys, for Paillier cryptosystem: $\mathsf{pk} = (n = pq, \mathbb{G}, g)$, $\mathsf{sk} = (p, q)$

1. **Registration Phase:** For voter $\mathsf{V}_{\mathtt{id}}$ the registrar, does the following steps:
   - Long credential: Pick $\mathsf{crd} \leftarrow \mathbb{Z}_n$, store crd on voter's device.
   - Short credential: Pick random PIN $a \in$ PIN-Set and send it to voter $\mathsf{V}_{\mathtt{id}}$.
   - Compute the error list for a based on the election policy: $\mathsf{ErrorList}_a = \{a_1 = a, a_2, \ldots, a_k\}$ and set $\mathsf{poly}_{\mathtt{id}} = \prod_{i=1}^{k}(x - \mathsf{crd} - a_i) = \sum_{i=0}^{k} p_i x^i$
   - Encrypt polynomial coefficients: For $i = 0, \ldots k : \mathsf{cp}_i = \mathsf{Enc}(p_i)$
   - Provide a designated proof of validity for the ciphertexts, cp_i, $i = 0, \ldots k$.
   - Publish $\mathsf{V}_{\mathtt{id}} : \big(\mathsf{CP} = (\mathsf{cp}_0, \ldots, \mathsf{cp}_k), \mathsf{Enc}(\mathsf{crd})\big)$ on bulletin board.
2. **Casting ballot:** Voter chooses her candidate m, and enter her choice of PIN, $\hat{a}$. The voting algorithm runs the following steps:
   - Encrypt m and long credential, $\mathsf{CT}_{\mathsf{vote}} = \mathsf{Enc}(m), \mathsf{CT}_{\mathsf{crd}} = \mathsf{Enc}(\mathsf{crd})$
   - For $i = 1, \ldots, k$ compute $\mathsf{cp}^*_i = \mathsf{cp}_i^{(\hat{a}+\mathsf{crd})^i} \cdot \mathsf{r}_i^{*n}$ and $\mathsf{CT}_i = \mathsf{Enc}((\hat{a} + \mathsf{crd})^i)$ for random number $\mathsf{r}_i, \mathsf{r}_i{}^*$. Provide a proof, π_{ballot}, (also proof of knowledge)

for the following relation:

$$\begin{aligned}\mathrm{R}_{\mathsf{ballot}} = \Big\{(x,w), x = \big(\mathsf{CT}_{\mathsf{vote}}, \mathsf{CT}_{\mathsf{crd}}, \mathsf{CT}_i, \mathsf{CP} = (\mathsf{cp}_i)_{i\in[k]}, \mathsf{CP}^* = (\mathsf{cp}^*_i)_{i\in[k]}\big)\\ w = \big(\mathsf{vote}, \mathsf{r}_{\mathsf{vote}}, \hat{a}, \mathsf{crd}, \mathsf{r}_{\mathsf{crd}}, \{\mathsf{r}_i, \mathsf{r}^*_i\}_{i\in[k]}\big) :\\ \mathsf{CT}_{\mathsf{vote}} = g^{\mathsf{vote}} \cdot h^{\mathsf{r}_{\mathsf{vote}}}, , \mathsf{vote} \in \texttt{List of candidats}, \mathsf{CT} = g^{\mathsf{crd}} \cdot h^{\mathsf{r}_{\mathsf{crd}}},\\ i = 1, \dots, k : \mathsf{CT}_i = g^{(\mathsf{crd}+\hat{a})^i} \cdot h^{\mathsf{r}_i}, \mathsf{cp}^*_i = \mathsf{cp}^{(\mathsf{crd}+\hat{a})^i} \cdot h^{\mathsf{r}^*_i}\Big\}\end{aligned}$$

This proof can be implemented efficiently using Sigma protocols and will rely on the DDH assumption, and will be given in a long version of the paper. They can be made non-interactive using the strong Fiat-Shamir heuristic. Note that the hash should contain all parts of the ballot.

- Cast $\mathsf{ballot}_{\mathsf{V}} = (\mathsf{CT}_{\mathsf{crd}}, \mathsf{CT}_{\mathsf{vote}}, \{\mathsf{cp}^*_1, \dots, \mathsf{cp}^*_k\}, \pi_{\mathsf{ballot}})$ with her ID.
- Obfuscate: Everybody can cast (invalid) votes with any voter ID. This will obfuscate whether voter ID participated in the election as in [6,11]

3. **Tally Phase:** Using the Paillier encryption scheme, allows us to efficiently sort ciphertexts based on plaintext values without decrypting them, see [13]. This techniques can be done in a multi-party computation which provide privacy for the e-voting protocol. $\mathsf{MPC}_{\min}$ the algorithm that takes as input the ciphertexts $\mathsf{ct}_1 = \mathsf{Enc}(m_1), \mathsf{ct}_2 = \mathsf{Enc}(m_2), \dots, \mathsf{ct}_t = \mathsf{Enc}(m_t)$ and outputs the index i^* such that $\mathsf{ct}_{i^*} = \mathsf{Enc}(m_{i^*}) : m_{i^*} = \min\{m_1, \dots, m_t\}$. We use this algorithm in the Tally phase:
   - **Ballot Validity check:** In the first step, we remove exact ballot copies and all ballots with invalid proof π_{ballot}. In the next step we need to remove extra ballots for each voter, making sure a valid ballot is kept, if existing.
   - **Weeding:** Since each voter will be associated with possibly more than one ballot, we need to weed them. We make sure a valid ballot is chosen - if existing. Assume there are q ballots with the same ID, $\mathsf{ballot}_1, \dots, \mathsf{ballot}_q$, We now homomorphically combine the public ciphertext cp_0 with the submitted encryptions to obtain an encrypted polynomial evaluation for each ballot: $\mathsf{Enc}(\mathsf{poly}_{\mathtt{id}}(\mathsf{crd}_i + \hat{a}_i)) = \mathsf{cp}_0 \cdot \prod_{j=1}^{k} \mathsf{cp}^*_j, i = 1, \dots q$. Denote by $t_i = \mathsf{poly}_{\mathtt{id}}(\mathsf{crd}_i + \hat{a}_i)$ and note this is zero if the ballot has a valid credential and pin. We now verifiably mix the pairs $\mathsf{Enc}(t_i), \mathsf{Enc}(\mathsf{vote}_i)$ and run the $\mathsf{MPC}_{\min}$ algorithm on the first ciphertexts to determine the one with the minimal t_i. We only keep this ciphertext and the corresponding encrypted vote and discard the rest. Note that this will select valid ballots having $t_i = 0$ if they exist.[6]
   - **Ballot anonymization:** We delete the ID, run all the remaing pairs $\mathsf{Enc}(t), \mathsf{Enc}(\mathsf{vote})$ through a verifiable parallel mixnet for re-encryption and permutation.
   - **Final PIN and Credential validity check:** Finally, for each ballot, we decrypt the polynomial evaluation. All ballots with non-zero polynomial

[6] This will give a random correct vote. The policy "Last valid vote counts" can be implemented by adding the received order to t_i.

evaluation will be discarded. We need to do this step without revealing any information about t_i for non-zero evaluation. Thus the tally tellers first jointly and verifiably multiply some random number onto t_i and then decrypt. We accept ballots with output zero and discard the rest.

– **Vote decryption:** Decrypt the remaining vote ciphertexts and compute the voting result.

Error tolerance property of the scheme: Note the following computation:

$$\mathsf{cp}_i = g^{p_i} \cdot \mathsf{r}_i^n \ , \ \mathsf{cp}_i^* = \mathsf{cp}_i^{(\hat{a}+\mathsf{crd})^i} \cdot \mathsf{r}_i^{*n} \Rightarrow \mathsf{cp}_i^* = g^{(\hat{a}+\mathsf{crd})^i p_i} \cdot \mathsf{r}_i'^n$$
$$\Rightarrow \mathsf{cp}_0 \cdot \prod_{i=1}^{k} \mathsf{cp}_i^* = g^{\sum_{i=0}^{n}(\hat{a}+\mathsf{crd})^i p_i} \cdot \mathsf{r}^n = g^{\mathsf{Poly}_{\mathtt{id}}(\mathsf{crd}+\hat{a})} \cdot \mathsf{r}^n$$

Decrypting this gives us the polynomial evaluation. Note that this evaluation will only check if $\hat{a} + \mathsf{crd}$ is valid. This should be sufficient for security. However, to check that both the credential is corrected and the PIN is in the allowed space, we can use a distributed plaintext equivalence test [21] between the submitted credential and the registrered credential and add the outcome under encryption to the polynomial evaluation.

Security analysis: The main advantage of this instantiation is sorting the ciphertexts without decrypting them. Note that $\mathsf{poly}_{\mathtt{id},\mathrm{PIN}}$ has the range in nonnegative integers. Therefore if there is any ballot with valid credential and PIN, the output of $\mathsf{MPC}_{\min}$ will be a valid ballot. On the other hand, it does not reveal whether any ballot has a valid pin or not, thus sidestepping the attack on the standard duplicate removal.

3.2 BGN Instantiation

The second instantiation is based on composite order groups introduced by [3] and the Groth-Sahai NIWI-proof system [7] with security are based on the Subgroup decision assumption.

The main point of using those in this instantiation are, BGN is a homomorphic encryption scheme which can be efficiently implemented in a bilinear group. Having bilinear map allows us to do the polynomial evaluation in an efficient and secure way and also having the efficient NIWI-proof system.

Definition 1. ***BGN Cryptosystem*** *works as follows. Its Key-Generation algorithm,* KGen *outputs a pair of keys:* $\Big(\mathsf{pk} = (n, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, h = g'^q), \mathsf{sk} = (p, q)\Big)$ *which* $\mathbb{G} = \langle g \rangle$ *and* $\mathbb{G}_T$ *are two groups of order* n *and the secret key consists of two primes* p, q *such that* $n = pq$. $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ *is bilinear* $(\forall a, b \in \mathbb{Z}, g \in \mathbb{G} : \mathbf{e}(g^a, g^b) = \mathbf{e}(g, g)^{ab})$*, non-degenerate* $(\mathbb{G} = \langle g \rangle \Rightarrow \mathbf{e}(g, g) \neq 1_{\mathbb{G}_T})$ *and commutable map. A ciphertext on message* $m \in [T]$*, for* $T < q$ *has the form* $\mathsf{CT} = g^m h^{\mathsf{r}} \in \mathbb{G}$ *for some random number* r*. Decryption: raise the ciphertext to power* p *and compute the discrete log.*

BGN E-voting Protocol:

1. **SetUp Phase:** The central authority runs the BGN key-generation algorithm to generate $(\mathsf{sk}_{\mathsf{BGN}} = p, q, \mathsf{pk}_{\mathsf{BGN}} = (n, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, h)$. Then chooses four random group elements $f_1, f_2, f_3, f_4 \in \mathbb{G}$. Note that $\mathbb{G} = \langle g \rangle$ is a cyclic group so there exists a unique integers $z_i, i \in [4]$ such that $f_i = g^{z_i}$. Set the secret key of election as $\mathsf{SK}_{\mathsf{election}} = (p, f_1, f_2, f_3, f_4)$ and public key of election as $\mathsf{PK}_{\mathsf{election}} = (n, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, h)$. Publish $\mathsf{PK}_{\mathsf{election}}$ on the bulletin board.
2. **Registration Phase:** Registrar, $\mathcal{R}$, for voter V does the following steps:
   - Generate credential and pin: crd, a as in the Paillier instantiation.
   - Generate the list of errors, $\mathsf{ErrorList}_a = \{a_1 = a, a_2, \ldots, a_k\}$. Then compute $\mathsf{poly}_a = \prod_{i=1}^{k}(x - a_i) = \sum_{i=0}^{k} p_i x^i$ and the following ciphertexts: $i \in [k] : \mathsf{cp}_i = \mathsf{Enc}(p_i) = g^{p_i} h^{\mathsf{r}_i}, \mathsf{cp}_0 = g^{p_0} \cdot f_1^{\mathsf{crd}} h^{\mathsf{r}} = \mathsf{Enc}(p_0 + \mathsf{crd} \times z_1)$.

     Note that, technically cp_0 is the encryption of $p_0 + \mathsf{crd} \times z_1$. Although z_1 is not a known value to any parties, the registrar can compute cp_0 without knowing its value.
   - Generates a designated proof of validity of the polynomial poly_a and all cp_i, for $i = 0, \ldots k$.
   - Store $\mathsf{CP} = (\mathsf{cp}_0, \mathsf{cp}_1, \ldots, \mathsf{cp}_k), \mathsf{CRD} = g^{\mathsf{crd}}$ in the user device and publish $\mathsf{Enc}(\mathsf{crd}) = g^{\mathsf{crd}} \cdot h^{\mathsf{r}}, \mathsf{CP}$ on bulletin board.
3. **Casting ballot:** Voter V chooses her candidate vote, and enter her choice of PIN, $\hat{a}$. The voting algorithm runs the following steps:
   - Compute, $\mathsf{CT}_{\mathsf{vote}} = \mathsf{Enc}(\mathsf{vote})$ and $\mathsf{CT}_{\mathsf{crd}} = \mathsf{Enc}(\mathsf{crd}) = \mathsf{CRD} \cdot h^{\mathsf{r}}$.
   - PIN encryption: For $i = 1, \ldots, k$ compute $\mathsf{CA}_i = \mathsf{Enc}(\hat{a}^i)$.
   - Re-randomize cp_i for $i = 0, \ldots, k$ by multiplying in a random $h^{\mathsf{r}_i^*}$ to generate cp_i^*.
   - Set $\mathsf{CA} = (\mathsf{CA}_1, \ldots, \mathsf{CA}_k), \mathsf{CP}^* = (\mathsf{cp}_0^*, \ldots, \mathsf{cp}_k^*)$ and provide a proof (Proof of knowledge), π_{ballot} for the following relation, including a joint proof of plaintext-knowledge for all the other ciphertexts in the ballot and include the rest of the ballot in the hash for non-malleability. This proof can be generated using the Groth-Sahai technique.

$$\begin{aligned}\mathsf{R}_{\mathsf{ballot}} = \Big\{(x, w), x = \big(\mathsf{CT}_{\mathsf{vote}}, \mathsf{CT}_{\mathsf{crd}}, \mathsf{CA}\big), w = \big(\mathsf{vote}, \mathsf{r}_{\mathsf{vote}}, \mathsf{CRD}, \mathsf{r}_{\mathsf{crd}}, \hat{a}, \{\mathsf{r}_i\}_{i\in[k]}\big) :\\ \mathsf{CT}_{\mathsf{vote}} = g^{\mathsf{vote}} \cdot h^{\mathsf{r}_{\mathsf{vote}}}, \mathsf{vote} \in \texttt{List of candidats},\\ \mathsf{CT}_{\mathsf{crd}} = \mathsf{CRD} \cdot h^{\mathsf{r}_{\mathsf{crd}}}, \{\mathsf{CA}_i = g^{(\hat{a})^i} \cdot h^{\mathsf{r}_i}\}_{i=1,\ldots,k}\Big\}\end{aligned}$$

   - Cast $\mathsf{ballot} = (\mathsf{CT}_{\mathsf{vote}}, \mathsf{CT}_{\mathsf{CRD}}, \mathsf{CA}, \mathsf{CP}^*, \pi_{\mathsf{ballot}})$

Polynomial evaluation: The following computation shows how to evaluate the polynomial on the input value $\hat{a}$, the PIN that was used by the voter:

$$\mathbf{e}(\mathsf{CT}_{\mathsf{crd}}, f_1)^{-1} \cdot \mathbf{e}(\mathsf{cp}_0^*, g) \cdot \mathbf{e}(\mathsf{cp}_1^*, CA_1) \cdots \mathbf{e}(\mathsf{cp}_k^*, CA_k) =$$

$$\mathbf{e}(\mathsf{CRD} \cdot h^{\mathsf{r}}, f_1)^{-1} \cdot \mathbf{e}(g^{p_0}(f_1)^{\mathsf{crd}} h^{\mathsf{r}_0}, g) \cdot \mathbf{e}(g^{p_1} h^{r_1}, g^{\alpha_i} h^{\gamma_i}) \cdot \ldots \mathbf{e}(g^{p_k} h^{\mathsf{r}_k}, g^{\alpha_k} h^{\gamma_k}) =$$

$$\mathbf{e}(\mathsf{CRD}, f_1)^{-1} \cdot \mathbf{e}(h, f_1)^{-\mathsf{r}} \mathbf{e}(g^{p_0} f_1^{\mathsf{crd}} h^{\mathsf{r}_0}, g) \cdot \mathbf{e}(g^{p_1} h^{\mathsf{r}_1}, g^{a^i} h^{\gamma_i}) \cdot \ldots \mathbf{e}(g^{p_k} h^{\mathsf{r}_k}, g^{a^k} h^{\gamma_k}) =$$

$$\mathbf{e}(\mathsf{CRD}, f_1)^{-1} \mathbf{e}(f_1, h^{\mathsf{r}}) \mathbf{e}(f_1, \mathsf{CRD}) \mathbf{e}(g^{p_0} h^{\mathsf{r}_0}, g) \cdot \mathbf{e}(g^{p_1} h^{\mathsf{r}_1}, g^{a^i} h^{\gamma_i}) \cdot \ldots \mathbf{e}(g^{p_k} h^{\mathsf{r}_k}, g^{a^k} h^{\gamma_k}) =$$

$$\cdot\, e(h^{\mathsf{r}}, f_1)(\prod_{i=0}^{k} \mathbf{e}(g^{p_i}, g^{\alpha_i})) \cdot (\prod_{i=0}^{k} \mathbf{e}(g^{p_i}, h^{\gamma_i})) \cdot (\prod_{i=0}^{k} \mathbf{e}(g^{\alpha_i}, h^{\mathsf{r}_i}))(\prod_{i=0}^{k} \mathbf{e}(h^{\gamma_i}, h^{\mathsf{r}_i}))$$

$$\mathbf{e}(g, g^{\sum_{i=0}^{k} p_i \alpha_i})) \cdot \mathbf{e}(g, h^{\mathsf{r}}) = \mathbf{e}(g, g^{\mathsf{poly}_a(\hat{a})}) \cdot \mathbf{e}(g, h^{\mathsf{r}})$$

Hence, if we raise above term to power p, if $\mathsf{poly}_a(\hat{a}) = 0$ the result is equal to 1 and otherwise not. Due to the secret f_1 and zero-knowledge proofs, malicious voters cannot construct a zero-evaluation dishonestly.

• **Tally Phase:** First, we check the validity of the proofs, π_{ballot}. In case any of any failure, the ballot will be discarded.

– Step 1: Compute the encrypted polynomial evaluation as above and provide a proof of its validity (efficient using the Groth-Sahai technique). Call this $\mathsf{Enc}_T(t)$ with t being the polynomial evaluation which can be seen as an encryption in the target space. Note that this is computed from the ballot alone. Now verifiably mix the tuples $(\mathsf{CT}_{\mathsf{crd}}, \mathsf{CT}_{\mathsf{vote}}, \mathsf{Enc}_T(t))$. For each ballot we now create $\mathsf{Enc}_T(\mathsf{crd} + t)$ and remove duplicates ballot having the same $\mathsf{crd} + t$ which basically means same credential and same error-equivalent PIN for honest ballots. We will do this via PETs. If we have a small number of voters, we can mix between each duplicate removal. For a larger number we suggest to split the board in two, remove duplicates separately, then mix and do duplicate removal again. This will decrease the information from the distribution of confirmed duplicates to a coercer carrying out the "leaky duplicate removal attack" mentioned in Sec. 2.

– Step 2: We now want to select eligible valid votes. We mix the above list and the list of registered encrypted credential. Then we perform PETs between each registered credential and the submitted credential and homomorphically add the polynomial value to this before decrypting the result. This will be one if the credential is correct and the polynomial evaluation is correct. When we get a positive test result we do a further PET against the credentials. This will reveal malicous authorities creating valid polynomial evaluations on their own. If this is positive too, we decrypt the vote and continue to the next registered credential.

4 PIN Space Coverings

Our voting protocol ensures that the voter's credential is validated even if they make certain typos in their PIN. This could e.g. be a transposition error or a single wrong digit.

The interesting question from a security viewpoint is now how much this reduces the entropy of the PINs. To have a precise research question, we investigate how many PINs an attacker needs to try to cover the whole PIN space. This is related to the brute force attack of an attacker holding the real credential e.g. in the smart card. We will not solve this exactly in generality, but give some upper and lower bounds. Note also, that users generally are not good at choosing random PINs as revealed in PIN frequency analyses. We thus recommend that the PIN should be generated uniformly at random and not chosen by the voter.

We first focus on the case where we allow PIN swaps and an error in one digit. Let us denote the PIN by $p_1p_2\cdots p_k$. We first compute the number of PINs covered by a PIN try. Let us start with the case $k = 2$. By $[p_1p_2]$, we mean the set of numbers covered by this PIN. Clearly $[p_1p_2] = \{p_1p_2, p_2p_1, p_1*, *p_2\}$, where $* \in \{0, 1, 2, \ldots, 9\}$. After removing the repeated cases we will have $|[p_1p_2]| = 20$ for the case $p_1 \neq p_2$ and it will be 19 for the case $p_1 = p_2$. Actually, for $2r$ distinct digits $p_1, \ldots, p_{2r}$, one can verify that the r 2-digits numbers $p_1p_2, p_3p_4, \ldots, p_{2r-1}p_{2r}$ will cover a total of $20r - 2\binom{r}{2}$ PINs. The formula can also be used to give an upper bound of PINs cover by r PIN trys, and thus it shows that the attacker needs at least 8 PINs to cover the entire PIN space of all 2-digits numbers. Since the attacker is trying to cover the PIN space with the minimum number of attempts, a good strategy seems to be to add PINs with distinct digits as much as possible to the basis. In the case there is no possible new PIN with distinct digits, we will then add a PIN which increase the size of current basis the most, and so forth until the PIN space is covered. We have implemented an algorithm in Python following this idea, but using random sampling to find the next optimal element for efficiency. For the case of 2-digits PIN, a basis of size 9 was found which is close to the theoretical lower bound.

Let us now consider the case of 3-digit PINs. For any PIN $p_1p_2p_3$ the maximum size of all covered PIN, $|[p_1p_2p_3]|$ is 30. Therefore 34 will be an lower bound for the size of basis of PIN space in this case.

Assume that only swapping errors are tolerated. For 2-digit PINs, finding a basis is equivalent to finding a basis for upper triangular matrices. There the basis size is 55 which the Python code also finds. For $k \geq 3$, an upper estimate of the cover of a single PIN is k (including itself) thus $10^k/k$ is a lower bound.

We collect the lower theoretical bounds and the upper bounds resulting from our Python code for PIN lengths between 2 and 5 in Table 1. We ran the code 1000 times in the case of 2,3 and 4 and just one time for the case 5.

PIN Length	2	3	4	5
S+W Lower Bound	8	34	250	2000
S+W Upper Bound	9	78	713	6490
S Upper Bound	55	465	4131	

Table 1. S+W means the system accepts swapping errors and wrong digit errors, where S means a system that just tolerate swapping errors.

5 Conclusions and Outlook

In this paper we have presented attacks and repairs on the NV12 scheme, especially, we have also presented protocols which are resilient to human errors in the form of PIN typos. It is interesting to notice that the digitally stored key could be combined or replaced with a key derived from biometric data. An important future direction is to make the error correction here so efficient that we can allow using noisy biometric data without fuzzy extraction.

For the Paillier-based system that we have presented it would be natural to add the tally system from Ordinos [12] since this is also based on Paillier encryption. Ordinos will only reveal the winner or the ranking of the candidates in the election, and will thus help for coercion-resistance in the case where there are candidates which expected to only get few or no votes. Another method that could used in both protocols is the risk-limiting tally method described in [9] which gives plausible deniability for the voter.

The PIN space analysis might be of general interest, and more precise results should be found. Interestingly, the one-digit error in k-digit PINs is related to Rook-polynomials, [1], in a k-dimensional chessboard.

Finally, some socio-tehcnical research questions are: 1) Which type of PIN errors do voters do when the are in a vote setting and do not get any feedback on the correctness of the PIN. 2) Related to this, what it the optimal PIN policy that corrects as many PIN typos while still keeping the entropy of the PIN space sufficiently high. 3) If we do not use a smart card, or use both a smart card and key storage: how well can voters be trained to handle, fake and hide secret keys.

Of course a main missing part is to provide proofs of security for our protocols.

Acknowledgments. This work was supported by the Luxembourg National Research Fund (FNR) and the Research Council of Norway for the joint project SURCVS and by the FNR CORE project FESS.

References

1. R.B.J.T. Allenby and A. Slomson. *How to Count: An Introduction to Combinatorics, Second Edition.* Discrete Mathematics and Its Applications. Taylor & Francis, 2011.
2. R. Araújo, A. Barki, S. Brunet, and J.s Traoré. Remote electronic voting can be efficient, verifiable and coercion-resistant. In *International Conference on Financial Cryptography and Data Security*, pages 224–232. Springer, 2016.
3. D. Boneh, E. Goh, and K. Nissim. Evaluating 2-dnf formulas on ciphertexts. In *In TCC*, pages 325–341, 2005.
4. M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a secure voting system. In *2008 IEEE Symposium on Security and Privacy, 18-21 May 2008, Oakland, California, USA*, pages 354–368. IEEE Computer Society, 2008.
5. C. Feier, S. Neumann, and M. Volkamer. Coercion-resistant internet voting in practice. In E. Plödereder, L. Grunske, E. Schneider, and D. Ull, editors, *44. Jahrestagung der Gesellschaft für Informatik, Informatik 2014, Big Data - Komplexität meistern, 2014*, volume P-232 of *LNI*, pages 1401–1414. GI, 2014.

6. P. Grontas, A. Pagourtzis, A. Zacharakis, and B. Zhang. Towards everlasting privacy and efficient coercion resistance in remote electronic voting. In *International Conference on Financial Cryptography and Data Security*, pages 210–231. Springer, 2018.
7. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. *Electronic Colloquium on Computational Complexity (ECCC)*, 14, 01 2007.
8. V. Iovino, A. Rial, P.B Rønne, and P. YA Ryan. Using Selene to verify your vote in JCJ. In *International Conference on Financial Cryptography and Data Security*, pages 385–403. Springer, 2017.
9. Wojciech Jamroga, Peter B Roenne, Peter YA Ryan, and Philip B Stark. Risk-limiting tallies. In *International Joint Conference on Electronic Voting*, pages 183–199. Springer, 2019.
10. Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Towards Trustworthy Elections*, pages 37–63. Springer, 2010.
11. O. Kulyk, V. Teague, and M. Volkamer. Extending helios towards private eligibility verifiability. In R. Haenni, R. E. Koenig, and D. Wikström, editors, *E-Voting and Identity*, pages 57–73, Cham, 2015. Springer International Publishing.
12. R. Küsters, J. Liedtke, J. Mueller, D. Rausch, and A. Vogt. Ordinos: A verifiable tally-hiding e-voting system. *IACR Cryptol. ePrint Arch.*, 2020:405, 2020.
13. H. Lipmaa and T. Toft. Secure equality and greater-than tests with sublinear online complexity. In F. Fomin, R. Freivalds, M. Z. Kwiatkowska, and D. Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, 2013, Proceedings*, volume 7966 of *Lecture Notes in Computer Science*, pages 645–656. Springer, 2013.
14. A. Silva Neto, M.Leite, R.Araújo, M. Pereira Mota, N. Sampaio Neto, and J. Traoré. Usability considerations for coercion-resistant election systems. In M. Mota, B. Serique Meiguins, R. Prates, and H.Candello, editors, *Proceedings of the 17th Brazilian Symposium on Human Factors in Computing Systems, IHC 2018, Brazil, 2018*, pages 40:1–40:10. ACM, 2018.
15. S. Neumann, C. Feier, M. Volkamer, and R. Koenig. Towards a practical jcj/civitas implementation. *INFORMATIK 2013–Informatik angepasst an Mensch, Organisation und Umwelt*, 2013.
16. S. Neumann and M. Volkamer. Civitas and the real world: Problems and solutions from a practical point of view. In *Seventh International Conference on Availability, Reliability and Security, Prague, ARES 2012, Czech Republic, August 20-24, 2012*, pages 180–185. IEEE Computer Society, 2012.
17. Peter B. Roenne. JCJ with improved verifiability guarantees. In *The International Conference on Electronic Voting E-Vote-ID 2016*, 2016.
18. P. B Rønne, A. Atashpendar, K. Gjøsteen, and P. YA Ryan. Coercion-resistant voting in linear time via fully homomorphic encryption: Towards a quantum-safe scheme. *arXiv preprint arXiv:1901.02560*, 2019.
19. Adi Shamir and Nicko Van Someren. Playing 'hide and seek'with stored keys. In *International conference on financial cryptography*, pages 118–124. Springer, 1999.
20. O. Spycher, R. Koenig, R.and Haenni, and M. Schläpfer. A new approach towards coercion-resistant remote e-voting in linear time. In *International Conference on Financial Cryptography and Data Security*, pages 182–189. Springer, 2011.
21. Pei-Yih Ting and Xiao-Wei Huang. Distributed paillier plaintext equivalence test. *I. J. Network Security*, 6(3):258–264, 2008.
22. S. Wiseman, P. Cairns, and A. Cox. A taxonomy of number entry error. In *Proceedings of the 25th BCS Conference on Human-Computer Interaction*, pages 187–196. British Computer Society, 2011.