Selene: Voting with Transparent Verifiability and Coercion-Mitigation

Peter Y A Ryan, Peter B Rønne, Vincenzo Iovino

Abstract. End-to-end verifiable voting schemes typically involves voters handling an encrypted ballot in order to confirm that their vote is accurately included in the tally. While this may be technically valid, from a public acceptance standpoint is may be problematic: many voters may not really understand the purpose of the encrypted ballot and the various checks that they can perform. In this paper we take a different approach and revisit an old idea: to provide each voter with a private tracking number. Votes are posted on a bulletin board in the clear along with their associated tracking number. This is appealing in that it provides voters with a very simple, intuitive way to verify their vote, in the clear. However, there are obvious drawbacks: we must ensure that no two voters are assigned the same tracker and we need to keep the trackers private.

In this paper, we propose a scheme that addresses both of these problems: we ensure that voters get unique trackers and we close off the coercer's window of opportunity by ensuring that the voters only learn their tracking numbers after votes have been posted. The resulting scheme provides receipt-freeness, and indeed a good level of coercion-resistance while also providesinga more immediately understandable form of verifiability. The cryptographyis under the bonnet as far as the voter is concerned.

The basic scheme still has a problem in some contexts: if the coercer is himself a voter there is a chance that the coerced voter might light on the coercer's tracker, or the coercer simply claims that it is his. We argue that in many contexts this may be an acceptable threat when weighed against the more transparent verification provided by the scheme. Nonetheless, we describe some elaborations of the basic scheme to mitigate such threats.

1 Introduction

The challenge with voting systems is to provide sufficient evidence to render the outcome beyond dispute while at the same time ensuring ballot secrecy and coercion resistance. Furthermore, the system has to be very easy to use and easily understandable. The response from the crypto community has been to develop the notion of End-to-End (E2E) Verifiability. A number of schemes have been proposed and some even implemented and deployed, for example, Pret a Voter [28] Wombat [3] and Scantegrity II [29], Helios https://vote.heliosvoting.org/, Civitas [10], Pretty Good Democracy [27].

Typically these schemes involve the creation of an encrypted version of the vote at the time of casting. The voter gets to retain a copy of the encrypted

vote which she can later confirm is correctly posted to a secure, append-only Web Bulletin Board (WBB). All the posted, encrypted ballots are then anonymously tabulated, either using mixes and decryption or exploiting homomorphic properties of the encryption to tabulate under encryption and then decrypt the result.

The assurance arguments are rather subtle though, and some people object to the use of crypto in voting on the grounds that the majority of the electorate will not really understand it and its role. Indeed, German Federal law, according to some interpretations, rules out the use of cryptography on the grounds that anyone should be able to understand the mechanisms without requiring any special knowledge. It is interesting therefore to explore the possibility of achieving some form of verifiability without the use of crypto. An early example of this is the article of Randell and Ryan [25] that uses scratch strips as an analogue of crypto. Another fine example is Rivest's ThreeBallot system [26].

Another approach is to have private ballot identifiers that allow voters to look up their vote in the clear on the WBB. Schneier in his book [30] for example suggests such an approach: voters are invited to invent their own random code and submit it with their vote. A slightly more sophisticated approach, in which the system and/or the voter's devices generates the numbers is presented in [1].

Introducing ballot identifiers has the appeal that it provides voters with a very simple, direct and easy to understand way to confirm that their vote is present and correct in the tally. There are however two significant drawbacks: care has to be taken to ensure that voters get distinct trackers and there is a danger of coercion. The first is an issue if for example the system could identify two voters likely to vote the same way and assigns them the same tracker. In this case it just posts one vote against this tracker and is free to stuff another vote of its own choice. The second danger is that a coercer requires the voter to hand over her tracker to allow him to check how she voted. Notice though that in this style of attack the coercer has a limited window of opportunity: he must request that the tracker be handed over before the results are published. It is this observation that we exploit to counter this threat: we arrange for the voters to learn their tracker numbers only after the information has been posted to the WBB.

This paper presents a scheme that addresses both of these shortcomings by:

- Guaranteeing that voters get unique trackers.
- Arranging for voters to learn their tracker only after the votes and corresponding tracking numbers have been posted (in the clear).

We hope that by putting all the crypto under the bonnet, voters, election officials etc. may find such a scheme more acceptable that conventional E2E verifiable schemes, that require voters to handle encrypted ballots. Here the voters just have to handle tracking numbers and votes in the clear. The scheme is also interesting in that it appears to shift the trust model for voter devices: in usual E2E schemes we need to worry about the voter's device encrypting the vote correctly. This typically necessitates complicating the protocol with *Benaloh*

challenges, [4], or similar ballot assurance mechanisms. Now that voters get to check their vote in the clear, a misbehaving device can be detected more readily, and resulting in a simpler voting ceremony.

A possible problem with the basic scheme, pointed out by Bill Roscoe, is that a coerced voter might by mis-chance choose the coercer's tracking number when she is deploying her coercion evasion strategy. Perhaps even more worrying is the possibility that the coercer will simply claim, falsely, that the tracker revealed by the voter is his and hence he "knows" that voter has not revealed her true tracker. This puts the voter in a very difficult situation. It seems that her best strategy is to stick to her guns and insist that she has revealed her true tracker. She does not know whether or not the coercer is telling the truth and indeed, ironically, the coercer does not have any means to prove to her that it is his tracker.

In a large elections with a small number of candidates the odds of lighting on the coercer's tracker will typically be small (unless the coercer is backing a serious loser), but even the remote possibility may be worrying to some voters. Furthermore, the coercer might claim, falsely, that the tracker is his, placing the voter in a difficult situation. Of course, for this to arise, the coercer must himself be a voter, and so the attack does not arise if the coercer is an outsider.

It is not immediately obvious how to counter this danger, but an enhancement to the basic scheme which counter this possibility is described in section 8, however it comes at a cost of a less transparent tally. The Selene scheme is in any case targeted at low coercion threat environments and so in such a context this problem could be regarded as minor. We argue that, in some contexts, the benefits arising from the greater degree of transparency outweigh the rather remote threat. In any event, we will argue that the basic scheme still provides receipt-freeness, if not complete coercion resistance.

It is worth noting that the constructions presented here could be thought of as a possible add-on to other schemes to provide a transparent form of verifiability. Indeed we could start with a simple, un-verifiable scheme that simply delivers encrypted votes to the server.

2 Background

Coercion can come in many flavours, from implicit, the coercer does't have to say anything, folk just know how they are expected to vote, to full-on: your personal coercer is on hand 24/7 to assist you in making the right voting choice. Making a voting system resistant to the latter form is extremely difficult, arguably impossible if the coercer really is observing the voter throughout the voting period. The Selene scheme is aimed at contexts where the coercion threat is closer to the former end of the spectrum: the coercer will issue some instructions and ask some questions. Selene will manage to mitigate such coercion attacks and at the same time allow the voters to directly verify that their vote is counted as intended.

3 Cryptographic Primitives

The parties involved. The parties involved in the protocol are: the n voters, the t Tellers, and the Election Authority (EA) that initially sets up the protocol. Further a public WBB is used for verifiable communication whereas private channels are used to send secret information which need to equivocated in case of coercion.

Web Bulletin Board. In common with most E2E verifiable systems we will assume the existence of a secure WBB. This can be thought of as an append-only secure broadcast mechanism: everyone has a consistent view of the information posted and, once posted, information cannot be removed. For a more detailed discussion and a possible implementation see [20].

Signature, threshold encryption and verifiable re-encryption mix protocols. We assume an (k, t)-threshold ElGamal encryption system without any trusted authority and with verifiable proofs of honest decryption. The cryptosystem must be such that any subset k of the t parties are able to securely decrypt a given ciphertext with a corresponding transcript of correct decryption. In our case the t parties correspond to the Tellers. One example of such scheme is given in Cramer *et al.* [13]. In this work we denote by $\{x\}$ the ElGamal encryption of x. In addition we will make use of a digital signature scheme $S = (\mathsf{Sign}, \mathsf{Verify})$ to sign the encrypted votes [16]. Further, we will use re-encryption mix nets, in particular also parallel verifiable shuffles protocols [24].

Plaintext equivalence test and proofs. Consider two ElGamal ciphertexts (a_1, b_1) and (a_2, b_2), encrypted with the same key sk, whose plaintexts are respectively m_1 and m_2. Assume that the private part sk is distributed among Tellers $T_1, \ldots, T_t$. Then $T_1, \ldots, T_t$ can execute a protocol to determine if $m_1 = m_2$ without leaking anything else. To do that, they compute $(a, b) = (a_1/b_1, a_2/b_2)$ and execute a threshold decryption of (a^r, b^r) for a random distributively generated exponent r. At the end if the decryption returns 1, then they agree on $m_1 = m_2$, or on $m_1 \neq m_2$ otherwise. For two ciphertexts c_1 and c_2, we denote by $\mathsf{PET}(c1, c2) = 1$ if the plaintext equivalence test holds for c_1 and c_2. Proofs of correct threshold decryption performed in a plaintext equivalence test are added for public verifiability.

RO model, the Fiat-Shamir heuristic, NIZKPoK and signatures. In our protocol we will make use of an ideal hash function modelled as a Random Oracle (RO, in short) [2]. The Fiat-Shamir heuristic [14] combined with the RO model will offer us the powerful tool of Non-Interactive Zero-Knowledge Proofs of Knowledge (NIZKPoK, in short). Actually we will need stronger variants of the Fiat-Shamir heuristic [6]. We will give more details in Appendix A.

Exponentiation mixes. Our protocol will benefit from the *exponentiation mix* construct from [19]. Here a list of Public Keys pk_i of the voters are put through a sequence of exponentiation mixes: the i-th mix server $\mathsf{MS_i}$ takes the batch of outputs from the previous server, raises each term to a secret common power s_i, subjects the resulting terms to a secret shuffle π_i and outputs the result to $\mathsf{MS_{i+1}}$. The first server takes the original list of PKs. The net effect is a list:

$$(g^{x_{\rho(i)} \cdot s})_{i \in [n]},$$

where:

$$\pi = \prod_{1}^{[n]} \rho_i, \quad s = \prod_{1}^{[n]} s_i,$$

and g^s is also published by the last Teller.

Any voter $V_i, i \in [n]$ can identify her pseudo-PK, pk^* in the list by computing $(g^s)^{x_i}$ and finding the match.

4 Related Work

E2E verifiable voting now has quite a long and rich literature, with many schemes having been proposed, both for in-person and remote, e.g. internet, voting. Here we will just mention some of the most closely related schemes. Note, Selene as presented here is intended for internet voting, but it would doubtless be straigforward to adapt it to in-person voting.

The most notable verifiable, internet voting scheme is Adida's Helios, https://vote.heliosvoting.org/. Helios is not receipt-free, but recently the Belenios RF scheme, [11], has been proposed to provide receipt freeness.

Juels et al [21] proposed a formal definition of coercion resistance and a credential-based mechanism to achieve this. The Civitas system, [10], http://www.cs.cornell.edu/projects/civitas/, implements this approach, with some enhancements.

The idea of voters having a private tracking number with which they can look up their vote in the clear on a bulletin board appears to go back the Schneier's "Applied Cryptography" book in which he suggests that voters choose a password to identify their vote. Much later the idea is revived for use in voting during ANR (Agence National de la Rechrche) funding committee meetings. A scheme that has some similarities to Selene in that votes appear in the clear alongside identifying number, is Trivitas, [8]. Here however the clear-text votes appear on the bulletin board at an intermediate step, followed by further mixing and filtering. Hence the voters do not verify their vote directly in the tally.

5 The Set-up Phase

The EA creates the threshold election key and keys share. Ideally this should be in a distributed, dealerless fashion [13]. We assume that voters already have

PK/SK pairs. When voters register for the election we assume that they, or more precisely their devices, create a fresh, ephemeral trapdoor key pair.

We now describe the construction whose goal is to inform voters of their tracking numbers, i.e. the number assigned to them and unique to them, in a way that provides them with high confidence that it is correct but allowing them to deny it if coerced. We do this generating trapdoor, Pedersen-style commitments to the tracking numbers.

The tracking numbers should be rather sparse and easily distinguishable. Each voter holds the trapdoor to her commitment which allows her to lie to a coercer if necessary. An additional advantage of this construction is that it would appear unnecessary to authenticate the message notifying the voter of the recommitment value as it would be hard for an attacker to compute an alternative value that would open the commitment to another valid tracking number. This avoids the need to introduce an authentication mechanism to such notifications which could introduce coercion threats if done naively. Designated Verifier Signatures would be a way to sidestep such coercion threats, but they would significantly complicate the scheme. Now the voter can construct such a message herself for any tracking number of her choice.

Distributed Generation of the Encrypted Tracker Numbers The Election Authority publicly creates the tracking numbers n_i and also computes g^{n_i} (to ensure that the resulting values fall in the appropriate subgroup) as well as the (trivial) ElGamal encryptions of the g^{n_i}: $\{g^{n_i}\}_{PK_T}$ and posts these terms to the WBB.

$$n_i,\ g^{n_i},\ \{g^{n_i}\}_{\mathsf{pk}_T}$$

The (Mix) Tellers now put the last, encrypted terms through a sequence of verifiable, re-encryption mixes to yield:

$$\{g^{n_{\pi(i)}}\}'_{\mathsf{pk}_T}$$

These are now assigned to the voters' PKs

$$(\mathsf{pk}_i, \{g^{n_{\pi(i)}}\}'_{\mathsf{pk}_T})$$

Note that, thanks to the mixing, the assignment of these numbers to the voters is not known to any party, aside from a collusion of all the mix Tellers. Note also that as this is a verified mix, as long as all the input numbers are unique it is guaranteed that each voter will be assigned a unique (encrypted) number. We still need to ensure that the number revealed to each voter is the number assigned to them in the above construction, and we will see this next.

5.1 Distributed Generation of the Tracker Number Commitments

Now, each Teller is required to produce n pairs of terms of the form:

$$(\{h_i^{r_{i,j}}\}_{\mathsf{pk}_T}, \{g^{r_{i,j}}\}_{\mathsf{pk}_T})$$

We have to provide NIZKPoK proofs that these terms are well-formed, i.e. that the $r_{i,j}$ exponents in the two terms are indeed identical and known and that the Teller knows such value, we present these in Appendix A. In addition we will have to assume that such proofs be non-malleable as we will explain later.

Here, for notational convenience, $h_i := g^{x_i} = \mathsf{pk}_i$.

Thus we now have a $n \times t$ array of such pairs, the columns corresponding to the Tellers and the rows to the voter ids. Now, for each voter, we form the product across the columns of the first elements to give:

$$\{h_i^{r_i}\}_{\mathsf{pk}_T} = \prod_{j=1}^{t} \{h_i^{r_{i,j}}\}_{\mathsf{pk}_T}$$

Where, due to the multiplicative homomorphic properties of ElGamal,

$$r_i := \sum_{j=1}^{t} r_{i,j}$$

Now we form the product of the $\{h_i^{r_i}\}_{\mathsf{pk}_T}$ and the $\{g^{n_{\pi(i)}}\}_{\mathsf{pk}_T}$:

$$\{h_i^{r_i} \cdot g^{n_{\pi(i)}}\}_{\mathsf{pk}_T} = \{h_i^{r_i}\}_{\mathsf{pk}_T} \cdot \{g^{n_{\pi(i)}}\}_{\mathsf{pk}_T}$$

This gives us the encryption under the Teller's PK of the trapdoor commitments to the tracking numbers: $(h_i^{r_i} \cdot g^{n_{\pi(i)}})$. We can now have a threshold set of Tellers perform verified, partial decryptions of these terms to reveal the commitments:

$$C_i := h_i^{r_i} \cdot g^{n_{\pi(i)}}$$

All of these steps are posted, along with NIZKPoK proofs and audits, to the WBB.

It seems that the Tellers cannot cheat in any effective way here aside from injecting invalid randoms which will result eventually in the voters being unable to open their commitment to a valid tracking number. But in any case, any such cheating will be detected by checks on the NIZKPoK proofs or random audits.

Now, for each voter there will be a tuple of terms posted to the WBB:

$$(\mathsf{pk}_i, \quad \{g^{n_{\pi(i)}}\}_{\mathsf{pk}_T}, \quad h_i^{r_i} \cdot g^{n_{\pi(i)}}, \quad)$$

The last entry of this tuple is left blank awaiting the voter's encrypted vote.

5.2 Voting

Voter V_i casts her vote in the form:

$$(\mathsf{Sign}_{\mathsf{V_i}}(\{\mathsf{Vote_i}\}_{\mathsf{pk_T}}), \Pi_{\mathsf{i}}),$$

where the ballot is signed either with the voter's true PK, or with her pseudo-PK if this has been configured (see Section 3), and Π_i is a non-interactive proof of knowledge of the plaintext. The signature and proofs are needed to ensure *ballot independence* [15, 12], and to prevent an attacker copying, re-encrypting a previously cast vote as his own.[1] Note that in conjunction with Selene such a copying attack would be particularly virulent: the attacker copies the victim's vote and casts it as his own. When the votes and trackers are reveled he sees exactly how the victim voted.

It is important that the server check for duplication of encrypted votes. It is also advisable to post the votes only once voting is closed. The signatures and proofs are checked for validity and, if valid, the encrypted votes are now paired off with the PK or pseudo-PK (and encrypted tracking number) with which they were signed. Double votes are handled according to the policy in operation, e.g. only the last vote cast by V_i is retained. Thus we get a list of tuples on the WBB:

$$(\mathsf{pk}_i,\ \{g^{n_{\pi(i)}}\}_{\mathsf{pk}_T},\ (h_i^{r_i} \cdot g^{n_{\pi(i)}}),\ \mathsf{Sign}_{\mathsf{V_i}}(\{\mathsf{Vote_i}\}_{\mathsf{pk_T}}, \Pi_{\mathsf{i}}))$$

After voting has closed, the encrypted ballots are posted alongside their (pseudo-)identity.

5.3 Mixing and Decryption

Now, for each row on the WBB, the second and forth terms of these tuples are extracted and the signature and proofs striped off the forth term. This gives pairs of the form:

$$(\{g^{n_{\pi(i)}}\}_{\mathsf{pk}_T}, \{\mathsf{Vote_i}\}_{\mathsf{pk_T}})$$

These are now put through a verifiable, parallel shuffle, e.g. [24]. Once this is done, a threshold set of the Tellers perform a verifiable decryption of these shuffled pairs. All of these steps along with the proofs are posted to the WBB. Thus, finally we have a list of pairs: tracking number, vote:

$$(g^{n_{\pi(i)}}, \mathsf{Vote_i})$$

from which the tracker/vote pair can immediately be derived: $(n_{\pi(i)}, \mathsf{Vote_i})$.

[1] Actually, even though it is largely used it is not known which form of non-malleability is achieved by the so called Enc+PoK paradigm where one adds a proof of knowledge to an ElGamal ciphertext. Another possibility is to resort to threshold Cramer and Shoup [32]. Note that change will be completely transparent in Selene where the cast system can be essentially arbitrary.

5.4 Notification of Tracker Numbers

For the notification of tracking numbers we will think of the Pedersen commitments whose construction we described earlier as forming the β component, i.e. the $h^r \cdot m$, of an ElGamal encryption under the voter's PK, but with the α component, i.e. the g^r, kept hidden. Thus we think of an ElGamal encryption as being represented:

$$(\alpha, \beta) := (g^r, h^r \cdot m)$$

The goal then is to reveal the α term to the voter in a deniable fashion.

Once the trackers and votes have been made available on the WBB for a sufficient period for the voters to note any alternative trackers as may be required to parry any attempted coercion, the Tellers send the voter V_j their share of the $g^{r_{j,i}}$ over a private channel:

$$T_j \rightarrow V_i : \ g^{r_{j,i}}$$

Once V_i's device has received these from all the Tellers it combines them to form g^{r_i}, the α term which along with the β term of the commitment $h_i^{r_i} \cdot g^{n_{\pi(i)}}$ to give the ElGamal encryption of $g^{n_{\pi(i)}}$ w.r.t. the voter's PK h_i:

$$(g^{r_i}, h_i^{r_i} \cdot g^{n_{\pi(i)}})$$

The voter can now decrypt this in the usual fashion using her secret key x_i, thus revealing $g^{n_{\pi(i)}}$ and hence $n_{\pi(i)}$.

The potential attacks of the Tellers and how we counter them. The α term is sent to the voter without any proof of origin. This is more user-friendly because such communications have to be deniable and should be faked by the voter in case of coercion. The point is that an adversary, even if colluding with all the Tellers, can only construct an α term that opens up to a valid tracker different from the true tracker of the voter with negligible probability. The precise statement and assumptions can be found in Appendix B.

The voter, or more precisely her device, can compute an alternative g'^{r_i} term that will decrypt to an alternative, valid tracker of her choice. Suppose that she wants her commitment to decrypt to the tracker value $m^* := g^{n^*}$, she inputs this to her device along with the commitment value β_i and the device computes the fake α term α':

$$\alpha' = \left(\frac{\beta_i}{m^*}\right)^{x^{-1}}$$

On the other hand it is intractable to anyone not knowing the trapdoor, i.e. x_i, to perform such a computation, see Appendix B for a proof. Note also that for the privacy of the tracking numbers we don't really need to encrypt the g^{r_i} terms, the trackers are still protected by the encryption under the voter's PK.

However, it is still important to send these terms to the voter over a private channel to ensure that they are deniable.

Another potential attack lies in the fact that a Teller could create his g^{r_j} term with knowledge of the g^{r_i}'s terms of the other Tellers so that the product of all r_i's be known to him. This would be possible if the NIZKPoK proofs be malleable and in fact this is the case if care is not taken when applying the Fiat-Shamir heuristic. In Appendix A we discuss how it is possible to use standard technique to make a NIZKPoK non-malleable. We stress that assuming that the NIZKPoK be non-malleable the aforementioned attack is nullified.

6 The Voter Experience

A goal of the design of this protocol is to make the voter experience as simple and intuitive as possible. We assume that the voters already possess public (signing) keys and trapdoor keys. First we describe the ceremony in the case that the voter does not experience any coercion. Then we describe the steps needed to counter a coercer.

6.1 The Core Ceremony

- The voter receives an invitation to vote along with a ballot.
- The voter inputs her choice and her device encrypts this under the Election PK and signs this. The device sends this to the Election Server. The device stores a copy of this.

After a suitable period the tracking number/vote pairs are anonymised and decrypted and displayed on the WBB. The voters receive an invite to visit the WBB, but will only be necessary at this stage if the voter has been coerced.

- After a suitable delay, the voter receives a notification of the α term, which she inputs to her device to allow it to extract her tracking number. Once she has this she can visit the WBB and confirm that her vote appears correctly against this tracker.

The last step is optional, to enable to voter to check that her vote was correctly recorded and entered into the tally. She can skip this if she is not interested in performing such a check.

6.2 The Ceremony in the Event of Coercion

If she is being coerced she needs to take some additional, coercion evasion steps, shown in italics:

- The voter receives an invitation to vote along with a ballot.

– The voter inputs her choice and her device encrypts this under the Election PK and signs this. The device sends this to the Election Server. The device stores a copy of this and the tracker commitment.
  This is strictly optional (and arguably perhaps unnecessary).
– *After a suitable period the Tracker numbers and votes are anonymised and decrypted. The (tracker, vote) pairs are displayed on the WBB and the voter receives an invite to visit this. She visits the WBB and notes down a tracking number that appears against the vote demanded by the coercer.*
– *The voter inputs this fake tracking number into her device and it outputs a fake α' term that coupled with her commitment, the β term of the ElGamal encryption of her tracker, will decrypt to the fake tracker.*
– After a suitable delay, the voter receives a notification of her "true" α term, which she inputs to her device to allow it to extract her tracking number from the commitment.
– *If the coercer demands that she reveal her tracking number she "reveals" the fake one. If he further demands that she reveals the* alpha *notification value, she reveals the fake α' she computed earlier.*
– Once she has her tracker she can visit the WBB and confirm that her vote appears correctly against this tracker.

Of course, she should also notify the appropriate authorities that coercion was attempted.

6.3 Enhancements

Here we describe some optional enhancements to the basic scheme.

Pseudonymous Credentials We can add an extra layer of anonymity by generating pseudonyms using the rather elegant *exponentiation mix* described in Section 3.

After the mixing, the WBB will be equipped with a list of pairs comprising a pseudo-PK, and an encrypted tracking number:

$$(\mathsf{pk}^*_{\rho(i)}, \{g^{n_{\pi(i)}}\}_{\mathsf{pk}_T})$$

This helps conceal the identities of those who cast votes, thus helping to counter forced abstention attacks. Note however that a really determined coercer can still force a voter to reveal her singing key and so can breach this anonymity. To counter such a coercer we need a more sophisticated mechanism described in the next section.

6.4 Malleable Signatures

Cortier *et al.* [11], following Blazy *et al.* [7], put forward an elegant mechanism to make the voter to lose control and knowledge of the randomness used to form the final ballot stored in the ballot box. This is done by introducing a voting

server, that is trusted for receipt-freeness (i.e., it is assumed not to collaborate with potential coercers) but not for privacy and correctness.

The voting server receives a ballot c of a voter along with her signature σ. The server then re-randomizes c producing the new ballot c' and is further able to compute a new signature σ' for c'. The voting server posts on the WBB c' and σ' and anyone can check that σ' be a valid signature on c' under the verification key of the voter.

Moreover, the voting server can compute this transformed signature on the re-randomized ciphertext without knowing neither decryption key nor signing key nor the plaintext. The security guarantess that it is infeasible to compute a signature on a ciphertext that encrypts a message of which no encryption has been signed. The same mechanism can be added to Selene to prevent coercion attacks at casting time.

6.5 Selene as an Add-on

It is interesting to note that the constructions described above could in many cases be added to an existing scheme, one without any verification features or perhaps one having conventional E2E verification involving encrypted receipts. Indeed, in some cases it could even be retro-fitted to an election that had already taken place. Suppose that a Helios vote had been conducted and contested. The trapdoor commitments to the trackers could be generated and associated to the voters as described above and the mixes and decryptions performed afresh. For this to work, the base scheme must use encryption such that we can run a parallel shuffle with the corresponding encrypted trackers.

7 Analysis

In this section we give a brief informal analysis of the security properties of Selene. A full, formal security analysis is postponed for future research.

7.1 Verifiability and Verification

If we think of Selene as an add-on to a base scheme, the universal verifiability of Selene is at least as strong as the base vote casting. In section 5.2 this is a Helios like scheme, but as mentioned in section 6.5 it could also be a more general scheme. Such schemes most often provide tallied-as-stored security, i.e. that the vote is tallied as cast by the device of the voter.

However, Selene could also to some extent be added to a vote casting scheme without universal verifiability. Indeed, the strength of Selene is to provide additional individual direct verification that the vote is tallied as intended by the voter.

The security of the tracker construction relies on interested parties checking the proofs and calculations done on WBB as follows, but note the latter are universally verifiable:

- Check that the trackers, n_i, written in plain on the WBB are indeed unique and their exponentiations g^{n_i} and the trivial encryptions thereof are correct (section 5).
- Check the ZK proofs for the mix of the encrypted trackers (section 5). This is to ensure both privacy and verifiability. We will elaborate on this in next subsection.
- Check the ZK proofs from the Tellers that the terms $\{h_i^{r_{i,j}}\}_{\mathsf{pk}_T}, \{g^{r_{i,j}}\}_{\mathsf{pk}_T}$ are well-formed. Further, it is checked that these are correctly multiplied together to give a commitment to the tracking number (section 5.1). It then follows from the theorem in appendix B that an adversary with overwhelming probability cannot fake the α term, which the voter receives and uses together with the commitment to decrypt the tracker. This of course assumes that the voter's secret key $x_i = \log_g h_i$ is not known to the adversary. We will comment on this below.
- Check the proofs in the verifiable parallel shuffle of the voter/tracker pairs and their decryption (section 5.3). As in a standard voting scheme using mixing for tallying this ensures that the tally is correct and in this case it further means that the tracker in the commitment is indeed the one shown next to the vote in the tally.

We conclude that if these checks are performed then a voter, who decrypts to a valid tracker, can be confident that this is the unique tracker assigned to her and the corresponding vote on the tally board is the vote stored encrypted on WBB.

More elaborate schemes also provide some security for the vote being stored as intended, even when the voter's device is malicious e.g. via Benaloh challenges [4] or by employing hardware tokens [18]. Selene, can however also provide verifiability in this respect. Checking the vote in the tally can reveal if a malicious device altered the intended vote. This requires that the voter checks her vote on an app or another device not controlled by the adversary. Further, the signature key used to cast the vote can also be different from the secret key x_i used to retrieve the tracker. In this case the device used to cast the vote does not even need to know x_i. This means that the adversary cannot calculate an alternative value for the α term and it will be more difficult to launch an attack. A voter can then even use the same device to receive the α term, then store it and first then reveal the secret key to get the tracker. Later the voter can then check if it gives the right tracker on another device.

7.2 Ballot privacy

The Selene scheme requires that the underlying ballot casting mechanism provides good privacy. Thus the encryption algorithm and its implementation used to encrypt the vote should ensure the secrecy of the vote. The first mix of the encrypted trackers means that only an adversary controlling all the mix servers would know the association of the tracking numbers to the voters, assuming that the proofs of the mixing have been checked. The posted commitments to

the tracking numbers are perfectly hiding unless the adversary colludes with all the Tellers. Finally the parallel mix preserve ballot privacy for both the vote and the tracker just like in a standard vote scheme using tallying via mix nets. Finally, the α term, if this should come in the hand of an adversary, does not reveal the tracker since it just a part of an ElGamal encryption of the tracker.

7.3 Receipt-freeness

In their seminal paper Benaloh and Tuinstra [5] defines receipt-free (which they call uncoercibility) informally as "no voter should be able to convince any other participant of the value of its vote".

If the vote casting scheme is receipt-free, e.g. by employing the model of BeleniosRF [11] for the vote casting, then Selene is receipt-free. Basically the extra information that the voter has in Selene is the unique tracking number. However, the voter can simply fake this (and importantly the corresponding α term) since the tally board is presented before the tracker retrieval. We do need to assume that he attacker cannot monitor the communication of the α terms to the voters. As mentioned before, it can happen that the voter chooses a fake tracker which coincide the tracker of the coercer, however, this does not constitute a proof of how she voted, it just undermines her claim to that tracker and associated vote.

To which extent this makes Selene vote buyer resistant is a subject of future research. The point is that even though the voter cannot prove her vote, she does have extra information, namely the tracker which is unique to her.

We also mention that Italian style (aka signature) attacks may be possible here when we are dealing with complex ballots. For some voting methods we may be able to counter this by splitting up the ballot into components and mixing separately.

7.4 Coercion: Threats and Mitigation

For Selene to be coercion resistant, we firstly need that this is true for the vote casting part. Some degree of coercion resistance can be obtained by combining BeleniosRF [11] with vote updating. Another possibility for partial coercion resistance is to use the scheme by Kulyk, Teague and Volkamer [22] where each voter can cast several vote values and only the sum of these will count in the end. The total number of votes are hidden in a cloud of null votes which any participant can cast for the voter.

For Selene the extra tracker verification step however also opens up for a coercion possibility: the coercer can demand to observe the receipt of the $g^{r_{j,i}}$. Of course the voter can always create a fake term $g^{r'_{j,i}}$ and pretend to the coercer that this is the term that was sent to her, see section 5.4. Further, the terms are sent at randomized times and the coercer will thus have to intensively follow the voter. However, the possibility of receiving a wrong term while the coercer is present, might be discouraging for the voter. A possibility to circumvent this is to allow voters to secretly contact the voting authorities to request that only

the fake $g^{r_{j,i}}$ term that the voter has calculated be communicated back to her. They are now safe from the coercion threat, but a coerced voter have lost the individual verifiability. This suggests a novel form of coercion resistance, distinct from the conventional one in which the voter gets to cast her intended vote and to verify it, or *coercion evidence*, [17], in which she gets to verify her vote but it might be nullified. Here she gets to cast her vote but if coerced may lose the possibility to verify it.

The coercion problem might escalate if the coercer is colluding (or pretends to be) with one of the Tellers. The voter then has to guess which $g^{r_{j,i}}$ to fake (this is incidentally also a problem in Civitas [10]). In the BeleniosRF construction there is a voting authority which is trusted for the receipt-freeness, and in this case we can circumvent this danger by letting this authority receive the $g^{r_{j,i}}$ terms and only forward the g^{r_i} to the voter.

True coercion-resistant vote schemes often work with credentials, e.g. Civitas [10]. The voting authority knows the true credential, and the voter can provide the coercer(s) with fake credential(s). Where Civitas is not directly compatible with Selene, one can imagine to combine its credential construction and the extra null votes of [22] to create a true coercion-resistant scheme compatible with the tracker construction. In this case the extra credentials can also be used to make the tracker retrieval coercion-resistant. A scheme could be as follows. After the tally board is created we allow a certain time for the voters to note the trackers, construct fake α-terms and contact the voting authorities privately with these terms. After this time the voter can log in to the voting system to get the α term, however the credential is also used in this process. The voting authority provide the true α term if the correct credential is used. If a fake credential is used, the system outputs the corresponding faked α-term which has been provided by the voter.

7.5 Dispute Resolution

Dispute resolution, the ability for a judge to determine the cheating or malfunctioning component or party when an error is reported, is quite hard to achieve, especially in the internet voting context. In Selene this could be tricky. If a voter claims that the vote corresponding to their tracker is not what they cast, it is hard to determine if it is the voter who is lying or mis-remembering, or the system that cheated. But this is a problem with the tracking number approach anyway.

If a voter insists that the vote on the WBB is wrong, we could resolve this if the voter is prepared to sacrifice their ballot privacy by allowing threshold decryptions of their ballot for example. This has to be performed with great care and suitable controls, and presumably *in camera* to avoid introducing coercion opportunities.

8 Selene II

The drawback of Selene is that a coerced voter might have the misfortune of choosing the coercer's tracking number, or the coercer simply claims, falsely, that this is his tracker. If the threats of the coercer are sufficiently unpleasant this possibility could be enough to deflect the voter from voting her intent. The goal of this construction is to provide voters with a set of alternative trackers, each pointing to one of the candidates, in such a way that these trackers are unique to her. If coerced she simply points to the tracker from this set that points to the coercers requested candidate, and now the coercer cannot claim ownership of this tracker. The tally board will now contain $c{\cdot}v$ additional tracking numbers, where c is the number of the candidates and v is the number of the voters. These will give one extra vote per candidate per voter which has to be subtracted in the tally. This is ok for simple plurality style elections, but not for more elaborate social choice functions, at least not without some adaptation. This aspect of the scheme is reminiscent of Rivest's *ThreeBallot* [26].

The necessary extra assumption for this construction is that the encryption scheme used to encrypt the cast votes must support Plaintext Equivalence Tests (PETs) (see Section 3). This happens to hold for both the schemes [11] and [22] discussed above. For the uncoerced voter this alternative scheme will be just like standard Selene. For the coerced voter, the difference is that she can request c extra trackers corresponding to a vote for each of the candidates, and these trackers are unique to her and cannot coincide with the one of the coercer. Of course, care is needed to ensure that a coercer cannot detect such a request, but in fact, as we will see shortly, it is possible to make public the c trackers assigned to each voter. This removes the need for a coerced voter to specifically request them.

Thus we start by constructing a total of $v \cdot (c+1)$ tracking numbers which are exponentiated and encrypted trivially as in Selene I:

$$n_a,\ g^{n_a},\ \{g^{n_a}\}_{PK_T}\,, \quad a = 1, \ldots, c \cdot v + v$$

As before, the (Mix) Tellers put the last encrypted terms through a sequence of verifiable, re-encryption mixes and get:

$$\{g^{n_{\pi(a)}}\}'_{PK_T}$$

These are now assigned to the voters' PKs in groups of $c+1$. Let us for simplicity denote the trackers given to voter i as $n_{i,k}$ with $i = 1, \ldots, v$, $k = 1, \ldots, c+1$

$$(PK_i, \{g^{n_{i,k}}\}'_{PK_T})$$

Nobody knows the permutation relating the original n_as and the $n_{i,k}$s. Just as in Selene, the Tellers construct a trapdoor commitment, in this case to the last tracker $n_{i,c+1}$

$$C_i := h_i^{r_i} \cdot g^{n_{i,c+1}}$$

where $r_i := \sum_{j=1}^{t} r_{i,j}$

Now, for each voter there will be a tuple of terms posted to the WBB:

$$\big(PK_i,\, (\{g^{n_{i,1}}\}_{PK_T},\, \{\mathsf{Cand_1}\}_{\mathsf{PK_T}}) \ldots, (\{\mathsf{g}^{\mathsf{n_{i,c}}}\}_{\mathsf{PK_T}},\, \{\mathsf{Cand_c}\}_{\mathsf{PK_T}}),\\ (\{g^{n_{i,c+1}}\}_{PK_T}, h_i^{r_i} \cdot g^{n_{i,c+1}},\ )\big).$$

The last entry of this tuple is left blank awaiting the voter's encrypted vote. The first c trackers are assigned to a vote for each of the candidates, named $\mathsf{Cand_1}$ to $\mathsf{Cand_c}$ here. The encryptions of these candidates are done with trivial randomness for verifiability.

The voter now cast her vote $\{\mathsf{Vote_i}\}_{\mathsf{PK_T}}$ which is added to the last entry with a corresponding signature and a non-interactive zero knowledge proof. Let us denote this Π_i. The WBB now have the following tuple for each voter

$$\big(PK_i, (\{g^{n_{i,1}}\}_{PK_T},\, \{\mathsf{Cand_1}\}_{\mathsf{PK_T}}), \ldots, (\{\mathsf{g}^{\mathsf{n_{i,c}}}\}_{\mathsf{PK_T}}, \{\mathsf{Cand_c}\}_{\mathsf{PK_T}}),\\ (\{g^{n_{i,c+1}}\}_{PK_T}, h_i^{r_i} \cdot g^{n_{i,c+1}}, \{\mathsf{Vote_i}\}_{\mathsf{PK_T}}, \Pi_{\mathsf{i}})\big)\,.$$

The tally board is then created by running all pairs of trackers and corresponding votes through the Mix-net and decrypting to get

$$(g^{n_a}, \mathsf{Vote_a}), \quad \mathsf{a} = 1, \ldots, \mathsf{c} \cdot \mathsf{v} + \mathsf{v}$$

By construction this contains an extra vote for each candidate for all voters.

8.1 Notification of Tracker Numbers

Finally the voters are notified of their tracker $n_{i,c+1}$ as in Selene, and they can use this to check their vote on the tally board. Of course, the verification is not as transparent as for the basic Selene scheme: as we will see each voter has revealed to them c trackers each of which points to a distinct candidate or option. The basic Selene construction notifies them of their "real" tracker, which should point to the vote that they cast. Furthermore, the construction guarantees that there will be another vote on the WBB identical to the one their tracker points to. But this requires a degree of understanding or at least confidence in the crypto. It seems that this loss of transparency is an inevitable consequence of seeking a higher level of coercion resistance.

For the corresponding voter, the Election Authority takes the pairs

$$(\{g^{n_{i,1}}\}_{PK_T},\, \{\mathsf{Cand_1}\}_{\mathsf{PK_T}}), \ldots, (\{\mathsf{g}^{\mathsf{n_{i,c}}}\}_{\mathsf{PK_T}}, \{\mathsf{Cand_c}\}_{\mathsf{PK_T}}), (\{\mathsf{g}^{\mathsf{n_{i,c+1}}}\}_{\mathsf{PK_T}}, \{\mathsf{Vote_i}\}_{\mathsf{PK_T}})$$

and re-encrypts these for anonymity of the coerced voter. Then it sends these re-encrypted terms to a Mix-net that mixes the first c pairs to get:

$$(\{g^{n_{i,\pi'(1)}}\}_{PK_T},\, \{\mathsf{Cand}_{\pi'(1)}\}_{\mathsf{PK_T}}), \ldots, (\{\mathsf{g}^{\mathsf{n_{i,\pi'(c)}}}\}_{\mathsf{PK_T}}, \{\mathsf{Cand}_{\pi'(\mathsf{c})}\}_{\mathsf{PK_T}}),\\ (\{g^{n_{i,c+1}}\}_{PK_T}, \{\mathsf{Vote_i}\}_{\mathsf{PK_T}}).$$

These pairs are then sent to the Tellers which performs a PET between the first c encrypted votes and the last actually cast vote $\mathsf{Vote_i}$. Only one of these should match assuming a valid vote has been cast. Let us denote the corresponding tracker $n_{i,\star}$, that is $\mathsf{Vote_i} = \mathsf{Cand}_\star$. The corresponding pair is removed, then the remaining c encrypted tracking numbers are run through a further Mix-net (to ensure ballot privacy) and decrypted by the tellers. The result is a set of c trackers

$$\{n_{i,k} | \, k = 1, \ldots, c+1\} \setminus \{n_{i,\star}\}$$

in which each points to a different candidate. These are posted to the WBB against the voter's ID. Since these are unique to the voter, she can use these safely as fake trackers. Of course, the tracker corresponding to the chosen candidate is precisely the tracker that the voter will get via the standard Selene approach.

8.2 Alternative Approach

We will now briefly describe an alternative way of decreasing the chance of being caught lying about a faked tracker. The main advantage compared to the previous approach is that the tally board this time contains precisely all the votes and no extra votes. The idea is simply to remove some of the trackers from this final tally on the WBB, say at least two for each candidate, the precise number can be changed according to the coercion level. The coerced voter can now claim that she is one of the unlucky participants who do not have a tracker on the tally board. This could also have happened to the coercer, but since there is at least two blanks for every candidate, this is still a possible scenario. If we want protection against a coercer who is controlling more voters even more trackers need to be removed.

Unlike standard Selene, the real trackers, which in this case need to be random, cannot be shown before the tally board creation. Instead the trackers g^{n_i} are created secretly by the EA, i.e. they are directly posted on WBB as

$$\{g^{n_i}\}_{\mathsf{pk}_T} .$$

Further, the trackers are drawn from a domain M which has a cardinality being polynomial in the security parameter. We think of this set as being known to the participants and there has to be an efficient algorithm searching it.

The scheme now follows the basic scheme until after the parallel shuffle. The decryption of the voter/tracker pairs is now done such that the vote part is decrypted first. In the first two instances of a vote for a given candidate the corresponding tracker is not decrypted, but the remaining are. The EA assists in retrieving the trackers from the exponentiated trackers. Alternatively a version of the scheme without exponentiation can also be used. The uniqueness of the (revealed) trackers can directly be checked.

All voters can retrieve their trackers as in basic Selene, however, a few will not find their tracker because it was not decrypted. These voters can check whether their tracker belong to M. Since this set is polynomial in size, Theorem 2 in

Appendix B will also hold in this case, i.e. it is with overwhelming probability impossible for an adversary, even colluding with all Tellers, to construct a faked α term opening to an element in the set M.

The major drawback is than some voters now have lost their ability to check their cast vote in the tally. These might protest and even reveal their tracker number which they obtain in the Selene construction, and which will be one of the removed trackers. However, they cannot prove that this was really their tracker. This is important because the coercer can ask the coerced voter what her tracker number is, but she can then simply choose a random number from M different from all the published trackers. The main danger is if the coerced voter by chance then chooses the actual tracker of the coercer (in case he really also have one of the missing trackers). However, this probability is

$$\frac{2}{L} \cdot \frac{1}{|M| - 2c}$$

if we removed $2c$ trackers and in the case that L votes were cast for the coercer's candidate. This is not negligible, but can be controlled and kept very low by adjusting the size of M and the security parameter.

9 Conclusions

We present a new voting protocol, based on the idea of tracking numbers but with the twist that voters do not learn their number until after voting has finished and the tracker/vote pairs have been posted to the bulletin board. This prevents the usual coercer attack on such tracking number systems: the coercer demands that the voter hand over her tracking number before the results are posted. We also provide a mix net construction that ensures that each voter gets a unique tracking number, preventing the attack of assigning the same tracker to voters likely to vote the same way. Furthermore, the construction ensures a high level of assurance that the voter receives the correct tracker while ensuring that this is deniable to a third party.

The resulting scheme provides a good level of verifiability and coercion resistance while at the same time providing a very direct and simple to understand mechanism for voter verification. The protocol is not crypto free, but the crypto is used in a way that is quite transparent to the voter it is all under the bonnet as it were. In particular the voter verification step involves just tracking numbers and votes in the clear. Voters do not have to handle encrypted ballots as is the case for previous E2E verifiable schemes. A further advantage appears to be that we avoid the need to audit the ballots created by the voter's device. Typically this necessitates the introduction of some kind of cut-and-chose protocol into the voting ceremony, significantly complicating the voter experience. Now, because the voter gets to check her vote in the clear we can sidestep this complication, but at the cost of a more complex dispute resolution procedure.

In is interesting to note that the Selene construction can be thought of as an add-on to an existing non-verifiable scheme, or indeed a conventional E2E

verifiable scheme for which people want a greater degree of transparency in the verification. Indeed Selene could even be retrofitted to a cryptographic election that has been contested. Note further that an option is to run the basic Selene I scheme but if a significant level of coercion is reported before and during the vote casting period, the Selene II constructions could be dynamically added to the WBB give the higher degree of coercion resistance.

References

1. Mathilde Arnaud, Véronique Cortier, and Cyrille Wiedling. Analysis of an electronic boardroom voting system. In *4th International Conference on e-Voting and Identity (VoteID'13)*, volume 7985 of *Lecture Notes in Computer Science*, Surrey, UK, July 2013. Springer.
2. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993.
3. Jonathan Ben-Nun, Niko Fahri, Morgan Llewellyn, Ben Riva, Alon Rosen, Amnon Ta-Shma, and Douglas Wikström. A new implementation of a dual (paper and cryptographic) voting system. In *5th International Conference on Electronic Voting, (EVOTE)*, 2012.
4. Josh Benaloh. Simple verifiable elections. In Dan S. Wallach and Ronald L. Rivest, editors, *2006 USENIX/ACCURATE Electronic Voting Technology Workshop, EVT'06, Vancouver, BC, Canada, August 1, 2006*. USENIX Association, 2006.
5. Josh Cohen Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 544–553. ACM, 1994.
6. David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 626–643. Springer, December 2012.
7. Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Workshop on Theory and Practice in Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 403–422. Springer, March 2011.
8. Sergiu Bursuc, Gurchetan S. Grewal, and Mark Dermot Ryan. Trivitas: Voters directly verifying votes. In Aggelos Kiayias and Helger Lipmaa, editors, *E-Voting and Identity - Third International Conference, VoteID 2011, Tallinn, Estonia, September 28-30, 2011, Revised Selected Papers*, volume 7187 of *Lecture Notes in Computer Science*, pages 190–207. Springer, 2011.
9. David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer, August 1993.
10. Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: A secure voting system. In *In IEEE Symposium on Security and Privacy*, 2008.

11. Véronique Cortier, Georg Fuchsbauer, and David Galindo. Beleniosrf: A strongly receipt-free electronic voting scheme. *IACR Cryptology ePrint Archive*, 2015:629, 2015.
12. Véronique Cortier and Ben Smyth. Attacking and fixing helios: An analysis of ballot secrecy. In *Proceedings of the 24th IEEE Computer Security Foundations Symposium, CSF 2011, Cernay-la-Ville, France, 27-29 June, 2011*, pages 297–311, 2011.
13. Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118. Springer, May 1997.
14. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, August 1987.
15. Rosario Gennaro. Achieving independence efficiently and securely. In James H. Anderson, editor, *14th ACM Symposium Annual on Principles of Distributed Computing*, pages 130–136. Association for Computing Machinery, August 1995.
16. Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.
17. Gurchetan S. Grewal, Mark Dermot Ryan, Sergiu Bursuc, and Peter Y. A. Ryan. Caveat coercitor: Coercion-evidence in electronic voting. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 367–381. IEEE Computer Society, 2013.
18. Gurchetan S. Grewal, Mark Dermot Ryan, Liqun Chen, and Michael R. Clarkson. Du-vote: Remote electronic voting with untrusted computers. In Cédric Fournet, Michael W. Hicks, and Luca Viganò, editors, *IEEE 28th Computer Security Foundations Symposium, CSF 2015, Verona, Italy, 13-17 July, 2015*, pages 155–169. IEEE, 2015.
19. Rolf Haenni and Oliver Spycher. Secure internet voting on limited devices with anonymized DSA public keys. In *2011 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '11, San Francisco, CA, USA, August 8-9, 2011*, 2011.
20. James Heather and David Lundin. The append-only web bulletin board. In *Formal Aspects in Security and Trust, 5th International Workshop, FAST 2008, Malaga, Spain, October 9-10, 2008, Revised Selected Papers*, pages 242–256, 2008.
21. Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, WPES 2005, Alexandria, VA, USA, November 7, 2005*, pages 61–70, 2005.
22. Oksana Kulyk, Vanessa Teague, and Melanie Volkamer. Extending helios towards private eligibility verifiability. In Rolf Haenni, Reto E. Koenig, and Douglas Wikström, editors, *E-Voting and Identity - 5th International Conference, VoteID 2015, Bern, Switzerland, September 2-4, 2015, Proceedings*, volume 9269 of *Lecture Notes in Computer Science*, pages 57–73. Springer, 2015.
23. Birgit Pfitzmann and Ahmad-Reza Sadeghi. Anonymous fingerprinting with direct non-repudiation. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 401–414. Springer, December 2000.
24. Kim Ramchen and Vanessa Teague. Parallel shuffling and its application to prêt à voter. In *2010 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '10, Washington, D.C., USA, August 9-10, 2010*, 2010.

25. Brian Randell and Peter Y A Ryan. Voting technologies and trust. In *In IEEE Symposium on Security and Privacy*, pages 50–56, 2006.
26. Ronald L. Rivest. The ThreeBallot Voting System. https://people.csail.mit.edu/rivest/Rivest-TheThreeBallotVotingSystem.pdf.
27. Peter Y A Ryan and Vanessa Teague. Pretty good democracy. In *IN: WORKSHOP ON SECURITY PROTOCOLS*, 2009.
28. P.Y.A. Ryan and S. A. Schneider. Prêt à voter with re-encryption mixes. Technical Report CS-TR-956, University of Newcastle, 2006.
29. Scantegrity Team. Scantegrity. http://www.scantegrity.org/papers/whitepaper.pdf.
30. Bruce Schneier. *Applied cryptography - protocols, algorithms, and source code in C (2. ed.)*. Wiley, 1996.
31. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer, August 1990.
32. Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 1–16. Springer, May / June 1998.
33. Hoeteck Wee. Zero knowledge in the random oracle model, revisited. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 417–434. Springer, December 2009.

A Efficient NIZKPoK for the well-formedness of the Tellers' ciphertexts

NIZKPok in the RO. In section 5.1 we need ZK proofs to ensure that the Tellers construct the commitments correctly. We will use non-interactive zero-knowledge proofs of knowledge (NIZKPoK) in the RO model. These can be constructed taking a sigma protocol and making it non-interactive using the Fiat-Shamir heuristic [14].

Precisely a NIZKPoK in the RO consists of two algorithms

$$\mathsf{NIZK} = (\mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$$

with the usual syntax and semantics of NIZKs [16] except that they have oracle access to a function $O(\cdot)$ that acts like a random oracle. A NIZKPoK in the RO has to satisfy the properties of completeness, soundness, zero-knowledge, and proof of knowledge (that implies soundness), where the latter are stated in the programmable random oracle.

Precisely, the simulator for the zero-knowledge can program the RO at some points and the distinguisher between the real experiment and the simulated experiment will be given access to an oracle modified at those points when it is fed with the output of the simulated experiment or with the original experiment otherwise [33]. Recall that in a sigma protocol, special soundness means that from two executions of the protocol (a, b_1, c_1) and (a, b_2, c_2) accepted by the verifier and with $b_1 \neq b_2$, it is possible to extract the witness.

When a sigma protocol is made non-interactive in the RO model with the Fiat-Shamir heuristic, the proof of knowledge property is modeled by stating

that if a prover can convince a verifier with some probability p then there exist an extractor that, having oracle access to the prover and having the capability of programming the RO, can extract the witness with probability polynomially related to p.

Our relation. Fix a DH-group[2] $\mathcal{G}$ of order p and two group elements $g, h \in \mathcal{G}$ and a public-key pk_T for the ElGamal cryptosystem. We want to prove the well-formedness of $\{g^r\} := (A_1, A_2)$ and $\{h^r\} := (B_1, B_2)$. Precisely, we are given two ciphertexts $c_1 := (A_1, A_2)$ and $c_2 := (B_1, B_2)$ and we want to construct a NIZKPoK system that allows to prove knowledge of values r, s_1 and s_2 in $\mathsf{Z_p}$ such that $a := g^r, b := h^r$, $c_1 := (A_1, A_2) = (g^{s_1}, a \cdot \mathsf{pk}_T^{s_1})$ and $c_2 := (B_1, B_2) = (g^{s_2}, b \cdot \mathsf{pk}_T^{s_2})$.

Our NIZKPoK. We construct a NIZKPoK system (in the RO) $\mathsf{NIZK} := (\mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ that uses an underlying NIZKPoK system $\mathsf{NIZK}^1 := (\mathsf{NIZK}^1.\mathsf{Prove}, \mathsf{NIZK}^1.\mathsf{Verify})$ for Chaum-Pedersen's proofs of knowledge of discrete log equality [9] and a NIZKPoK system $\mathsf{NIZK}^2 := (\mathsf{NIZK}^2.\mathsf{Prove}, \mathsf{NIZK}^2.\mathsf{Verify})$ (in the RO) for Schnorr's proofs of knowledge of an exponent [31]. NIZK^1 and, respectively, NIZK^2 can be constructed taking a Chaum-Pedersen's sigma protocol and, respectively, a Schnorr's sigma protocol, and making them non-interactive using the Fiat-Shamir heuristic (actually we will need the *strong* Fiat-Shamir heuristic [6] as we will elaborate at the end of the section).

First we describe the prover.

1. $\mathsf{NIZK.Prove}$ has embedded $\mathcal{G}, p, g, h$ and pk_T and takes as input the statement (A_1, A_2, B_1, B_2) satisfying the above relation and the corresponding witness r, s_1 and s_2 for it. The algorithm starts setting $a := g^r$ and $b := h^r$.
2. $\mathsf{NIZK.Prove}$ computes $\pi_{1,1} := \mathsf{NIZK}^2.\mathsf{Prove}(A_1, s_1)$, i.e., it computes a proof of knowledge of the randomness in A_1, and $\pi_{1,2} := \mathsf{NIZK}^2.\mathsf{Prove}(B_1, s_1)$, i.e., it computes a proof of knowledge of the randomness in B_1.
3. $\mathsf{NIZK.Prove}$ chooses random element $t \in \mathsf{Z_p}$ and computes $(A'_1, A'_2) := (A_1^t, A_2^t)$ and $(B'_1, B'_2) := (B_1^t, B_2^t)$.
4. $\mathsf{NIZK.Prove}$ shows that this is done correctly, by publishing 3 proofs $\pi_{2,1}$, $\pi_{2,2}$, $\pi_{2,3}$ of the following equations: (1) $\log_{A_1} A'_1 = \log_{A_2} A'_2$, (2) $\log_{B_1} B'_1 = \log_{B_2} B'_2$ and (3) $\log_{A_1} A'_1 = \log_{B_1} B'_1$. Such proofs are computed using $\mathsf{NIZK}^1.\mathsf{Prove}$ using the witness t.
5. $\mathsf{NIZK.Prove}$ computes $C := a^t$ and $D := b^t$.
6. $\mathsf{NIZK.Prove}$ proves that (A'_1, A'_2) and (B'_1, B'_2) encrypt these by two proofs $\pi_{3,1}$ and $\pi_{3,2}$ for the following facts: (4) $\log_g A'_1 = \log_{PK_T} A'_2/C$ and (5) $\log_g B'_1 = \log_{PK_T} A'_2/D$. These proofs are computed using $\mathsf{NIZK}^1.\mathsf{Prove}$ with witness $s_1 t$ for $\pi_{3,1}$ and $s_2 t$ for the proof $\pi_{3,2}$.
7. $\mathsf{NIZK.Prove}$ uses $\mathsf{NIZK}^1.\mathsf{Prove}$ with witness t to compute a proof π_4 for the fact that the two exponents in C and D are the same: (6) $\log_g C = \log_h D$.

[2] Henceforth, by DH-group we mean a group where the Decision Diffie-Hellman assumption holds.

8. NIZK.Prove computes $\pi_5 := \mathsf{NIZK}^2.\mathsf{Prove}((g, C), rt)$, i.e., it computes a proof of knowledge of the randomness rt in C.
9. Finally NIZK.Prove outputs proof $\pi := (A'_1, A'_2, B'_1, B'_2, C, D, \pi_{1,1}, \pi_{1,2}, \pi_{2,1}, \pi_{2,2}, \pi_{2,3}, \pi_{3,1}, \pi_{3,2}, \pi_4, \pi_5)$.

A verification algorithm NIZK.Verify is constructed in the obvious way by verifying in sequence all proofs in π using the verification algorithms of the two underlying systems. It is straightforward to observe that the system be complete.

Security. ZKness follows from a standard hybrid argument by observing that any proof is either the output of $\mathsf{NIZK}^1.\mathsf{Prove}$ or $\mathsf{NIZK}^2.\mathsf{Prove}$ with valid witnesses, and thus a simulator can just invoke in sequence the simulators for the latter two systems, and the proof contains $(A'_1, A'_2, B'_1, B'_2, C, D)$ that, under the Diffie-Hellman assumption, can be simulated as well.

Soundness follows from the following simple observations. Suppose towards a contradiction that there exist a false statement (A_1, A_2, B_1, B_2) and a proof $\pi = (A'_1, A'_2, B'_1, B'_2, C, D, \pi_{1,1}, \pi_{1,2}, \pi_{2,1}, \pi_{2,2}, \pi_{2,3}, \pi_{3,1}, \pi_{3,2}, \pi_4, \pi_5)$ accepted by NIZK.Verify. By construction that means that all sub-proofs $\pi_{2,1}$, $\pi_{2,2}$, $\pi_{2,3}$, $\pi_{3,1}$, $\pi_{3,2}$, π_4 and π_5 are accepted by $\mathsf{NIZK}^1.\mathsf{Verify}$. By soundness of NIZK^1 and by definition of step 4 in the construction of NIZK.Prove it follows that (1) $\log_{A_1} A'_1 = \log_{A_2} A'_2$, (2) $\log_{B_1} B'_1 = \log_{B_2} B'_2$ and (3) $\log_{A_1} A'_1 = \log_{B_1} B'_1$.

The latter three equations imply that there exist value t such that the equations (1)-(3) in step 2 in the construction of $\mathsf{NIZK}^1.\mathsf{Prove}$ hold.

By soundness of NIZK^1 and by definition of steps 6 and 7 in the construction of NIZK.Prove and by the fact that the sub-proofs $\pi_{3,1}, \pi_{3,2}, \pi_4$ are verified, it follows that the equations (4)-(6) given in step 5 of the construction of NIZK.Prove hold.

Now equation (6) implies that there exists t' such that $C = g^{t'}$ and $D = h^{t'}$.

Now equations (4)-(5) prove that (A'_1, A'_2) encrypts C and (B'_1, B'_2) encrypts D.

By equations (1)-(3) it follows that if (A'_1, A'_2) encrypts a plaintext C and (B'_1, B'_2) encrypts D then (A_1, A_2) encrypts $a := C^{s'}$ and (B_1, B_2) encrypts $b := D^{s'}$ for some $s' \in \mathsf{Z_p}$. Thus, by previous implications it follows that $a = g^{s't'}$ and $b = g^{s't'}$. This contradicts the hypothesis that the statement be false.

Using a similar argument and the programmability of the random oracle it is easy to construct an extractor that invoke the extractors for the underlying systems to extract the witnesses, and in particular observing that the proof of knowledge NIZK^1 used in steps 8 allow to extract rt and in step 4 to extract t and thus r, and the proofs of knowledge NIZK^2 in step 2 allow to extract s_1 and s_2.

About non-malleability. As further security guard we assume that the underlying NIZK proof systems, NIZK^1 and NIZK^2 be non-malleable[3] to prevent malleability

[3] In the case of Chaum-Pedersen or Schnorr's proofs made non-interactive with the Fiat-Shamir heuristic, this can be guaranteed by including an identifier in the hash of the statement.

attacks where a teller could wait for the proofs provided by the other tellers and thus compute a valid (sound) proof without knowing the underlying randomness. We refer the reader to Bernhard *et al.* [6] for a deeper discussion on these attacks and how to defeat them.

B About the hardness of constructing a fake α term

In Section 5.4 we claim that it is hard to calculate an alternative α term that the voter would decrypt to a valid tracker different from the unique tracker assigned to the voter. This is true even if the adversary is colluding with the Tellers. We now make this statement and its assumptions precise.

Remember that we consider a Schnorr group [31] of prime order q where we calculate modulo p and g is a generator of the group.

We will assume that the commitments $C = g^n h^r$ are correctly constructed by the tellers, where n is the tracker of the voter, h the public key and r is the randomness used in the α-term g^r. If the Tellers could write $\{g^r\}_{PK_T}$ and $\{g^a h^r\}_{PK_T}$ to the bulletin board for some a instead of $\{g^r\}_{PK_T}$ and $\{h^r\}_{PK_T}$, then the Tellers would easily be able to construct an α-term opening to a fake tracker. However, the NIZKPoK proofs given in Appendix A prevent such malicious behaviour.

Further, we assume that the set of trackers, T, has a cardinality which is at most polynomial in the security parameter (being the bit-size of p or q). This means, on one hand, that a random drawn number in the group has negligible probability of being a tracker, and on the other hand that the set $\{g^m\}_{m \in T}$ can be constructed and searched by a PPT algorithm.

Finally we assume the hardness of the 1-Diffie-Hellman Inversion Problem (1-DHI, in short) [23].

Assumption 1 (1-DHI). *Given a description of a DH-group $\mathcal{G}$ of prime order p, a generator g for it, and an element $g^x \in \mathcal{G}$, where x is randomly chosen in $\mathsf{Z_p}$, no PPT algorithm can output with non-negligible probability an element $g^{1/x}$.*

We then claim that the following theorem is true.

Theorem 2. *If the 1-DHI assumption holds there exists no PPT algorithm $\mathcal{A}$ which takes as inputs a description of a DH-group $\mathcal{G}$ along with a generator g for it, a set T of tracker number of polynomial size, two values $C = g^n h^r, h = g^x \in \mathcal{G}$ and outputs with non-negligible probability a term α such that C/α^x is of the form $g^{n'}$ where $n' \neq n$ is a valid tracker, $n' \in T$. Further, this holds true even if the algorithm is given n and r.*

Proof. The reduction is simple. Suppose towards a contradiction that there exist such algorithm $\mathcal{A}$ outputting with non-negligible probability some value α satisfying the assumed property. We can then construct another PPT algorithm $\mathcal{B}$ that first constructs and searches $\{g^m\}_{m \in T}$ to obtain n' (if succesful otherwise it halts) and then computes and outputs $\beta = (\alpha / g^r)^{1/(n-n')}$ as its guess for

$g^{1/x}$. The probability that $\mathcal{B}$ breaks the 1-DHI assumption is greater or equal to the probability that α satisfy the assumption: in fact if α satisfies the equation $C/\alpha^r = g^{n'}$, where $C = g^n h^r$, then $\alpha = g^s$ with $s := r + (n - n')/x$.