Using Selene to Verify your Vote in JCJ

Vincenzo Iovino, Alfredo Rial, Peter B. Rønne, and Peter Y. A. Ryan

University of Luxembourg
vinciovino@gmail.com, {alfredo.rial, peter.roenne, peter.ryan}@uni.lu

Abstract. We show how to combine the individual verification mechanism of Selene with the coercion-resistant e-voting scheme from Juels, Catalano and Jakobsson (JCJ). This results in an e-voting scheme which allows the voter to check directly that her vote is counted as intended, but still allows her to mitigate coercion.
We also construct variants of the protocol which provide everlasting privacy or better verifiability. Further, both improvements of JCJ and Selene are discussed.

1 Introduction

Remote e-voting gives voters the opportunity to conveniently vote from home, work or even abroad. However, it also presents cryptographers with the difficult task of integrating both verifiability and privacy properties in a secure, efficient and usable e-voting protocol. One of the hardest problems of leaving the reassuring frame of a voting booth is to protect voters against coercion attempts. Juels, Catalano and Jakobsson (JCJ) [JCJ05] found a way to provide coercion-resistance across multiple elections, assuming only a single coercion-free registration. The registration provides the voters with credentials which they use for voting. Coerced voters can provide the coercer with a fake credential, and a vote cast using this will not be counted. The system was later implemented as Civitas [CCM08].

The JCJ-mechanism might be worrisome to the normal user. Was the credential entered correctly? Did someone else manage to override my vote? In the end, it would be reassuring for the voters to be able to directly check that their votes were counted correctly. However, providing voters with such a service endangers the receipt-freeness and coercion-resistance if not done carefully. Fortunately, Selene [RRI16] provides us with a mechanism for individual tallied-as-intended verifiability while being able to mitigate the coercion threat. This is done by giving each vote a unique tracking number, but first revealing this to the voter after the tally has been published. Unfortunately, Selene was developed for Helios style protocols, but in this paper we will show that the construction can also be applied to the coercion-resistant vote casting system from JCJ/Civitas. Indeed we will consider different variants of JCJ and show how Selene can be added to JCJ even in the case when we want to provide everlasting privacy via pseudonyms, or when we offer better verifiability properties. We will also see how to address the secure platform problem with the extra verifiability gained from Selene. Along the way we will discuss some problems and solutions of the JCJ construction with cross-election and dynamic coercion. Further, we will give a more efficient construction of the zero-knowledge proofs needed in Selene.

1.1 Related work

Since the seminal paper defining coercion-resistance [JCJ05], there have been numerous paper analyzing the JCJ protocol and providing alternatives, see e.g. [NFVK13a] and references therein.

Selene [RRI16] is based on the idea of having trackers for the votes, an idea already suggested in Schneier's book [Sch94], which later independently also appeared in a scheme used for ANR (Agence National de la Recherche) funding committee meetings. Recently, sElect [KMST16] uses trackers to achieve good accountability. However, in all of these cases the tracker directly represents a receipt, whereas Selene mends this by delaying when the voter can obtain the tracker.

The idea of everlasting privacy goes back to Moran and Naor [MN06] and have been studied in several works, see e.g. [CPP13] for how to make perfectly private audit trails in general election schemes, or [ACKR13] for how to do automated verification of everlasting protocols. Here we focus on pseudonymity rather than anonymity. However, if we follow JCJ closely, this is the best we can do since the credentials themselves will be like pseudonyms to a future adversary.

The secure platform problem is one of the main problems in e-voting. One solution is to use out-of-band channels and code-voting, see [Cha01,RT09]. In e.g. Helios [Adi08] Benaloh challenges [Ben06] should help to detect malware, but are unfortunately not often used [KOKV11]. Relying on hardware tokens is yet another possibility, see [HK14,GRCC15], but is not always unproblematic, see [KR16].

2 Building Blocks

Our construction uses the following building blocks: a non-interactive zero-knowledge proof system (NIZK) [BFM88] in the random oracle model [BR93], the ElGamal public key encryption scheme [Gam85], threshold encryption with a plaintext equivalence test [JJ00], a verifiable re-encryption mixnet [SK95], the Pedersen commitment scheme [Ped91], a web bulletin board [HL09], untappable channels [HH07] and anonymous channels [Fre00].

3 System Model and Setup

We first describe the parties involved in an e-voting scheme.

Voters. The voters V_i ($i = 1, \dots, n$) register for voting, cast ballots, obtain trackers and verify the voting results.

Tally Tellers. The Tally Tellers T_j tally the cast ballots and publish the results.

Registration Tellers. The Registration Tellers RT_k register voters.

Tracker Tellers. The Tracker Tellers TT_l process trackers. They could be the same parties as the Tally Tellers or the Registration Tellers, but they are kept separate here due to the different trust assumptions.

Our e-voting scheme consists of the following phases.

Setup. In the setup phase, the parties generate secret and public keys. Each voter creates a designated verifier key. The Tally Tellers generate a public key pk_T for a threshold encryption scheme.

Registration. In the registration phase, a voter V_i and the registration tellers run a protocol. The designated verifier key dvk_i of V_i and pk_T are used as inputs. As a result of the protocol, the voter obtains a credential C_i. Additionally, the voter identifier V_i, the key dvk_i and an encryption of C_i under pk_T are published on the web bulletin board (BB).

Tracker Preparation. In this phase, the Tracker Tellers and the voters run a protocol. A set of trackers $\{n_i\}_{i=1,...,n}$, the designated verifier keys of the voters and pk_T are used as inputs. As a result of the protocol, each voter obtains a Pedersen commitment to its tracker. Additionally, an encryption under pk_T of the tracker associated with a voter V_i is appended to the row for voter V_i on the BB. In this protocol, the association between trackers and voters is not revealed to any party.

Vote Casting. In this phase, a voter V_i computes a ballot and publishes it on the BB. In our construction, the ballot contains an encryption under pk_T of the credential C_i and of the vote vote_i.

Tallying. In this phase, the Tally Tellers take as input the ballots published on the BB and run a protocol to output pairs (vote_a, n_a), which associate each valid vote with the tracker of the voter that cast that vote. Those pairs are published on the BB.

Tracker Retrieval. In this phase, a voter V_i and the tracker tellers run a protocol as a result of which V_i learns the tracker n with which it became associated in the tracker preparation phase.

Once a voter learns her tracker n_a, the voter can verify on the BB that the pair (vote_a, n_a) is correct.

Setup. Let G be a cyclic group of prime order q and g be a generator of G. In the setup phase, each voter creates designated verifier key $\mathsf{dvk}_i = g^{x_i}$. The designated verifier keys are used to provide deniability in the registration phase in JCJ and the implementation Civitas. Additionally, we use the same designated verifier key in the Selene construction as the public key for the ElGamal encryption scheme. JCJ also suggest an alternative registration with an erasure function. In that case we need a PKI as in Selene where dvk_i is the voter's public key.

The Tally Tellers run the distributed key generation algorithm of the threshold encryption scheme to generate a public key pk_T and obtain each a private share of the secret key.

4 Description of the e-voting protocol

In this section, we describe the protocol combining JCJ and Selene in detail.

4.1 Registration

The registration is quite similar to JCJ/Civitas. Each voter has a designated verifier key dvk_i. For each eligible voter V_i, each Registration Teller RT_j randomly picks $C_{ij} \leftarrow_{\$} G$ and publishes $\{C_{ij}\}_{\mathsf{pk}_T}$ on BB in a row marked for voter V_i. As discussed in Section 7.1, we could instead use pseudonyms PV_i if everlasting privacy is desired.

For each voter, the encryptions are multiplied together to homomorphically obtain a single credential C_i. On BB, we now have the following row for each voter

$$V_i \ , \mathsf{dvk}_i \ , \{C_{ij}\}_{\mathsf{pk}_T} \ , \prod_j \{C_{ij}\}_{\mathsf{pk}_T} = \{C_i\}_{\mathsf{pk}_T}$$

Here we deviate from JCJ/Civitas by also including the public key dvk_i in the row. This key will be the public key used by the voter in Selene.

The voter now receives the credential shares and designated verifier proofs from the Registration Tellers

$$RT_i \rightarrow V_i : \, C_{ij} \, , \pi_{ij}$$

where π_{ij} is a designated proof to the key dvk_i proving that $\{C_{ij}\}_{\mathsf{pk}_T}$, appearing on BB, is an encryption of C_{ij}. The voter can now calculate C_i and check the proofs.

If the voter is coerced, she chooses at random an alternative value C_i' in G and shows this to the coercer. The proofs can be faked with her designated verifier key. It is here of course essential that the voter knows the secret key, but a coerced voter can even reveal this secret key to the coercer, as long as the coercer does not cooperate with the registration tellers. It is important that the coercer not be present by the reception of all C_{ij}'s. In general, we assume that the coercer does not interfere in the registration process.

The credential can be reused for several elections, and could, in principle, be obtained in booth by the registration authorities.

4.2 Tracker Preparation

Whereas the previous part was very similar to JCJ/Civitas, we now add the main ingredient of Selene, namely, the personal voting trackers that each voter can use to check her tallied vote.

The trackers $\{n_i\}_{i=1,\ldots,n}$ should be a negligible set of $\mathbb{Z}_q$ (i.e. the chance of a random element in $\mathbb{Z}_q$ being a tracker is negligible).

The Tracker Tellers first publish

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

on BB, where the encryption is with trivial randomness. The trackers are sent through a re-encryption mix and one anonymised tracker is added to each of the voters' rows to obtain

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

where π is the permutation used for mixing. In the following we will suppress π for easier notation. Note that, whereas credentials can be used for several elections, this tracker mixing needs to be renewed for each election.

Further each Tracker Teller TT_j randomly chooses $r_{ij} \leftarrow_{\$} \mathbb{Z}_q$ for each voter and publishes

$$\{\mathsf{dvk}_i^{r_{ij}}\}_{\mathsf{pk}_T} \, , \{g^{r_{ij}}\}_{\mathsf{pk}_T} \, , \varPi_{ij}$$

where $\varPi_{ij}$ is a non-interactive zero-knowledge proof that this is done correctly. The proof is presented in the Selene protocol, see [RRI15], App. A. As in Selene, the terms from each Teller are now homomorphically combined with the encryption of the tracker, and we obtain a trapdoor commitment to the tracker

$$\{g^{n_i}\}_{\mathsf{pk}_T} \prod_j \{\mathsf{dvk}_i^{r_{ij}}\}_{\mathsf{pk}_T} = \{g^{n_i} \, \mathsf{dvk}_i^{r_i}\}_{\mathsf{pk}_T}$$

with $r_i = \sum_j r_{ij}$. This is appended to each voter's row. Finally, the Tally Tellers decrypt the trapdoor commitment to the tracker, $g^{n_i} \, \mathsf{dvk}_i^{r_i}$, for each voter.

4.3 Vote Casting

Vote casting is done like in JCJ. Here we follow Civitas. If voter V_i wants to vote "vote$_i$", she anonymously sends to BB

$$(\{C_i\}_{\mathsf{pk}_T} \, , \{\text{vote}_i\}_{\mathsf{pk}_T}, \pi)$$

where π is a zero-knowledge proof that the vote is well-formed together with a proof that C_i and vote$_i$ are simultaneously known, which prevents vote copying. To cast a vote in presence of a coercer, the fake credential given to the coercer is simply used in place of the real one.

4.4 Improving the Coercion Resistance of JCJ

The JCJ protocol has a tally procedure which leaves room for certain coercion attacks. Let us first remind ourselves how the tally procedure works. It relies heavily on the Tally Tellers performing Plaintext Equivalence Tests (PETs) on the encryption of the credentials, see e.g. [CCM08].

1. Zero-knowledge proofs of the cast ballots are checked, and invalid ballots are removed.
2. Duplicates,i.e., ballots that use the same credential, are removed according to the existing vote update policy. This is done using PETs among the ciphertexts of the credentials in the cast ballots. This means that the coercer cannot mark the vote with a chosen number of duplicates.
3. The list of ciphertexts of registered credentials is anonymized using a mix-net. Further, from the list of valid ballots after duplicate removal, we likewise use a parallel mix-net to anonymize the pairs of ciphertexts of credentials and votes.
4. Unauthorized votes, i.e., ballots that do not use a registered credential, are removed by performing PETs of the credentials from the cast votes with the list of registered credentials.
5. The remaining valid votes can now be decrypted to reveal the tally.

The duplicate removal can in certain quite special situations give the coercer unwanted information and correspondingly hinders coercion-resistance, as we will now see. This was discovered, but not analyzed, in [Roe16]. The problem appears when the coercion happens dynamically or across elections. Consider an uncoerced voter who has already voted. The coercer now detects this somehow, say by overhearing this or seeing this in the browsing history of the voter.[1] The coercer can now coerce the voter just before voting ends. The coerced voter now gives the coercer a fake credential, and they can sit down and cast an, in fact, invalid vote. However, in the duplicate removal phase, it will then be evident that the credential was fake, since no duplicates are detected for the fake vote. To circumvent this, all voters should start by casting fake votes if they want to be prepared for later coercion threats, which seems pretty complicated. Note that the protocol in [KHF11] actually does something similar to prevent board flooding attacks on JCJ, but the cost is a statistical coercion-resistance.

Another case is a voter which was coerced in an earlier election and gave the coercer a fake credential. At a later election, the coercer can now cast a vote using this credential and check whether this will have duplicates in the duplicate removal phase. If this does not happen, the coercer can conclude that either the credential was fake, or that the voter did not vote in the latter election, which might be improbable. This means that the coerced voter also needs to cast votes using the fake credential even at elections after being coerced to be on the safe side.

Note that it does not help to do a mix before performing the duplicate elimination since the groups of ballots could still be marked by a certain number of duplicates.

If vote updating is not intended, we can sidestep the issue by simply dropping the step of duplicate removal. After anonymizing both the registered credentials and the cast ballots, PETs are performed for each registered credential against

[1] We can assume that coerced voters are careful to use only devices out of the reach of the coercer or to delete browsing history, but this is more unlikely for uncoerced voters.

the cast ballots until the first match comes up. We then pick this as the vote for the given credential. For the set of cast votes for a given valid credential this will pick one in the set at random. The method thus reveals a minimum amount of information, but makes vote updating harder to implement. Further it also decreases verifiability as discussed below, but Selene helps here.

4.5 Tallying with Selene

Tallying with Selene requires a minor modification. First, all proofs are checked and invalid votes are discarded. Then all cast pairs

$$(\{C_a\}_{\mathsf{pk}_T}, \{\text{vote}_a\}_{\mathsf{pk}_T}) \mapsto (\{C_{\pi(a)}\}_{\mathsf{pk}_T}, \{\text{vote}_{\pi(a)}\}_{\mathsf{pk}_T})$$

are re-encryption mixed.

Further the pairs of registered credentials and tracking numbers

$$(\{C_i\}_{\mathsf{pk}_T}, \{g^{n_i}\}_{\mathsf{pk}_T}) \mapsto (\{C_{\pi'(i)}\}_{\mathsf{pk}_T}, \{g^{n_{\pi'(i)}}\}_{\mathsf{pk}_T})$$

from each voter's column are re-encryption mixed in parallel. From each entry in this anonymised list of credential-tracker pairs, the Tally Tellers do PETs against the credentials from the anonymised list of cast votes. The first time we get a positive match, the corresponding vote is decrypted (verifiably) together with the corresponding tracker. If wanted, one can also do more elaborate PETs (like in JCJ-Civitas), first removing all duplicate votes, possibly with some vote update policy, as explained in Section 4.4.

The end result (after taking the discrete log of the trackers) is the Tally Board of valid vote-tracker pairs

$$(\text{vote}_a, n_a) \ .$$

4.6 Tracker Retrieval

Finally, the tracker retrieval happens like in Selene. Each Tracker Teller provides each voter with their share $g^{r_{ij}}$.

$$TT_j \to V_i : g^{r_{ij}} \ .$$

This happens according to some random time distribution a suitable time after the tally has been published, see [RRI16].

The voter (or rather her device) combines these shares to get g^{r_i}. Together with the public trapdoor commitment $g^{n_i}\,\mathsf{dvk}_i^{r_i}$, the term g^{r_i} forms an ElGamal encryption of the tracker under the key dvk_i. The voter can now decrypt and directly check that her vote appears correctly on the Tally Board.

Trackers can be faked in the case of coercion, just like in Selene. That is, the voter finds the wanted fake tracker, n^*, on BB for the coercer's choice of vote and calculates

$$\left(g^{-n^*} g^{n_i}\,\mathsf{dvk}_i^{r_i}\right)^{x_i^{-1}}$$

as the fake term to give to the coercer instead of g^{r_i}. Here x_i is the secret key of $\mathsf{dvk}_i = g^{x_i}$.

A potential attack would raise if an adversary, possibly colluding with all the Tracker Tellers, could make a voter get a fake g^{r_i} term that the voter decrypts to a valid tracker different from the true tracker of the voter with non-negligible probability. In [RRI15] it is proven that this is hard under a standard computational assumption.

5 More Efficient Zero-Knowledge Proofs in Selene

In the tracker preparation phase, the Tracker Tellers publish

$$\{\mathsf{dvk}_i^{r_{ij}}\}_{\mathsf{pk}_T}, \{g^{r_{ij}}\}_{\mathsf{pk}_T}, \Pi_{ij}$$

where the zero-knowledge proof was of the correctness of this construction, i.e. that the two generators are raised to the same known power. However, the term $\{g^{r_{ij}}\}_{\mathsf{pk}_T}$ is not really needed. In principle, it could be used for accountability if the Tracker Teller tries to send a wrong $g^{r_{ij}}$ to the voter. However, for deniability, the Tracker Teller sends this term without any proof to the voter. This means that there is no proof that the Teller sent a wrong message to the voter. Thus we suggest to only publish

$$\{\mathsf{dvk}_i^{r_{ij}}\}_{\mathsf{pk}_T}, \Pi'_{ij}$$

where Π'_{ij} is a shorter zero-knowledge proof, showing that the ciphertext indeed encrypts the key dvk_i to a known power. In a long version of this notem we present this proof in details; it consists of 8 group elements in some group of prime order p and of 6 elements of $\mathbb{Z}_p^\star$. We also prove in the long version that the adversary, also in this case, even when colluding with all Tellers, only has a negligible chance of constructing a fake term $g^{r_{ij}}$ that makes the voter decrypt to a valid tracker different from her real tracker.

6 Security assumptions and arguments for security

In this section we will briefly mention the trust assumptions for the voting authorities and give brief explanations of why the different security properties hold.

6.1 Trust assumptions for the Tellers

- The Registration Tellers are trusted individually for coercion-resistance and collectively for verifiability. For everlasting privacy via pseudonyms (see section 7.1) they are individually trusted for everlasting privacy.
- The Tally Tellers are trusted collectively for privacy (and hence coercion-resistance) and verifiability. A threshold version follows directly. We will here assume that the verifiable reencryption mixes done it the protocol are performed by the Tally Tellers, and that these are private if at least one Teller is honest.

– The Tracker Tellers are trusted collectively for privacy. They are trusted individually for coercion-resistance since the voter needs to know which $g^{r_{ij}}$ to fake for the coercer (like for the Registration Tellers).

6.2 Verifiability

We assume that the voters keep their private designated verifier keys secret. An adversary colluding with all the Registration Tellers can however still obtain the credential of a voter, and cast votes on her behalf, violating at least eligibility verifiability. The same can happen if the adversary and all the Tally Tellers collude, see also section 7.2 below how to mitigate this risk.

However, if such grand collusion do not happen, the only ballots on BB with a given voters correct credential are with overwhelming probability cast by the voter herself. That the correct vote is now chosen in the tally is secured by checking the zero-knowledge proofs of the verifiable PETs and verifying the correctness of the mixes. Finally, the actual decryption of the vote can also be verified.

The correctness of the individual verifiability of the Selene trackers, is very similar to the original Selene construction. The verification of the first mix of the trackers ensure that each voter gets a unique tracker, from the set of trackers. The pairwise mix of registered credentials and trackers, together with verification of the PETs ensure that this tracker is assigned to the voter's cast vote. Again, the correct decryption of the trackers can be verified. That the voters receive the correct trackers with overwhelming probability is discussed above.

6.3 Vote privacy

If the Tally Tellers or Tracker Tellers collude they can easily break privacy. Otherwise privacy of the mixes and encryptions will ensure privacy. In general, ballot independence is ensured by the construction (at least if we do not do the duplicate weeding) if we check the proofs of the PETs. This also means that even if the Registration Tellers collude and can cast valid votes on behalf of voters, this does not violate privacy.

6.4 Coercion-resistance vs coercion-mitigation

Coercion-resistance and, related, receipt-freeness is a harder problem. The point is that even in the ideal version of the scheme, the voters will know exactly which vote is theirs in the final tally by checking their unique tracker. This is intended and gives the voter a reassurance of the correctness of the vote. However, each voter knowing their unique tracker, does constitute a piece of information, not obtainable in standard voting schemes, and which is not foreseen in standard definitions of coercion-resistance and receipt-freeness.

Coerced voter however still have good options to *mitigate* coercion. They have algorithms to both fake their credential and the term to obtain their tracker number. The difference to standard coercion-resistance crystallizes when the voter shows a fake tracking number to the coercer, and it turns out to be the coercer's own tracker. This was analyzed in Selene [RRI15] where also several alternative versions without this drawback were discussed, but at the cost of a less clear Tally Board.

7 Extensions and Alternative Protocols

7.1 Everlasting Privacy via Pseudonyms

Privacy is easy to break for a future adversary who is able to break the employed encryption, e.g. because the DDH assumption happens to be broken or simply by the expected increase in computational power over time. In general, we think about the future adversary as having unlimited computational power, but only being active after the election using the data from BB.

A quick and dirty way to obtain everlasting privacy is to use pseudonyms. I.e., instead of labelling the rows on the bulletin board with the voter IDs, we use pseudonyms. We assume that only the Registration Tellers and the Tracker Tellers know the relation between the pseudonyms, designated verifier keys and the actual voter IDs. Especially, this information will not be public and not available to the future adversary.

Of course, pseudonyms are not the best way to preserve privacy, especially across elections. However, they are easy to implement with not too big usability costs. In particular, the JCJ construction works with credentials, which to the future adversary are just like pseudonyms labelling the voters, even though they only appear under encryption. As we show now, we can also use the Selene mechanism in this case with some modifications.

Registration with pseudonyms. In the registration phase, we mark the voter's row on BB with the pseudonym PV_i instead of V_i

$$PV_i \ , (\mathsf{dvk}_i)^{s_{ij}} \ , \ \{C_{ij}\}_{\mathsf{pk}_T} \ , \prod_j \{C_{ij}\}_{\mathsf{pk}_T} = \{C_i\}_{\mathsf{pk}_T}$$

Note that each Registration Teller also takes the public key of the voter dvk_i and raises it to the random power s_{ij} before publishing it. For each voter, we can now collect the terms

$$PV_i \ , (\mathsf{dvk}_i)^{s_i} = \prod_j (\mathsf{dvk}_i)^{s_{ij}} \ , \prod_j \{C_{ij}\}_{\mathsf{pk}_T} = \{C_i\}_{\mathsf{pk}_T} \ ,$$

with $s_i = \sum_j s_{ij}$. The Registration Tellers now send both the credential shares C_{ij}, the random exponents r_{ij}, the pseudonym and the designated verifier proofs to the voter

$$RT_i \rightarrow V_i : \ C_{ij} \, , s_{ij} \, , \pi_{ij} \, , PV_i$$

where π_{ij} is a designated proof to the key dvk_i proving that $\{C_{ij}\}_{\mathsf{pk}_T}$, appearing on BB, is an encryption of C_{ij}. The voter checks the proofs and the validity of the values s_{ij}. Further, the voter can now calculate C_i and s_i. For internal purposes the voter can update her key to be $(\mathsf{dvk}_i)^{s_i}$.

The reason for raising dvk_i to s_i is two-fold. The first reason is to blind the public key from the future adversary. From $(\mathsf{dvk}_i)^{s_i}$, it is information-theoretically impossible to infer dvk_i. The second reason is to prevent the following verifiability attack. Suppose that all registration tellers collude. They could then point two or more voters to the same pseudonym and credential, which would only be detected if the attacked voters unlikely compare pseudonyms. This would only give one vote to the two voters. Note that this verifiability is outside the scope of the JCJ assumptions, assuming at least one Registration Teller is honest. However, in [Roe16], it was shown that we can do better (see also below). However, by knowing the exponentials, the registration tellers would need to know a discrete logarithm relation between the attacked voters, which is infeasible by the hardness of the discrete logarithm problem, if we assume that the PKI has been set up properly.

The remainder of the protocol can now proceed as above with dvk_i replaced by $(\mathsf{dvk}_i)^{s_i}$. The future adversary will be able to relate a vote to the pseudonym and $(\mathsf{dvk}_i)^{s_i}$, but not directly to the voter. Note that the Tracker Tellers need to know the relation between pseudonyms and voters to return the random terms $g^{r_{ij}}$ to the voters. Like the Registration Tellers they are thus also assumed not to be colluding with the future adversary. This trust is one of the reasons to distinguish them from the Tally Tellers.

7.2 Stronger Verifiability

In [Roe16], a version of JCJ-Civitas was presented which has stronger security guarantees, and only changes the registration and voting procedure slightly. The main point is that the voters know the discrete logarithm of their credential, and this can be seen as a secret key. The cast ballots containing the encrypted credential are basically anonymously signed using this secret key. This prevents verifiability attacks where either all Tally Tellers or Registration Tellers are corrupted. In that case, they know the secret credentials, and could cast valid votes on behalf of any voter. If we use the duplicate removal step, which had slight coercion-resistance problems, as discussed above, this attack could be detectable by alert voters. However, even so, it could lead to unsolvable disputes about the validity of the election, see [Roe16].

Selene can also be added to this version of JCJ just as for standard JCJ. However, we can also create a new combination of JCJ and Selene where, post-registration, the voters only have to handle a single key (actually, coerced voters, of course, also need to handle the fake keys).

The registration works as follows. For a given voter V_i, all Registration Tellers RT_j choose random values $c_{ij} \in \mathbb{Z}_q$ and publish $\{g^{c_{ij}}\}_{\mathsf{pk}_T}$ on BB. The voter gets c_{ij} from RT_j together with a designated zero-knowledge proof to dvk_i, proving the correct encryption of $g^{c_{ij}}$.

The ciphertexts of the credential shares can now be multiplied together, but are further multiplied by $\{\mathsf{dvk}_i\}_{\mathsf{pk}_T}$, which for verifiability is encrypted with trivial randomness. Since ElGamal is homomorphic, the final ciphertext is an encryption of the voter credential $C_i = g^{c_i} := g^{\sum_j c_{ij}+x_i}$. However, in this case the voter, and only the voter, knows the discrete logarithm, since the Registration Tellers do not know the secret key of dvk_i.

In case of coercion, the voter will present the coercer with a random number c_i' and corresponding group element $C_i' = g^{c_i'}$ and claim this is the real credential – just like in JCJ, but now working with the discrete logarithms instead of the group elements.

After registration, BB contains

$$V_i\,, \{C_i\}_{\mathsf{pk}_T}$$

and the uncoerced voter only needs to store the discrete logarithm of C_i. We do not demand now V_i to store dvk_i separately. The Tracker Tellers can mix and add $\{g^{n_i}\}_{\mathsf{pk}_T}$ to each voter as above, but the Tracker Tellers can now only work with $\{C_i\}_{\mathsf{pk}_T}$. Due to the homomorphic property of ElGamal, this is however enough. To create the trapdoor commitment, the Tracker Teller TT_j randomly chooses $r_{ij} \leftarrow_{\$} \mathbb{Z}_q$, and publishes for each voter

$$\{C_i\}_{\mathsf{pk}_T}^{r_{ij}} = \{C_i^{r_{ij}}\}_{\mathsf{pk}_T}\,, \mathit{\Pi}_{ij}$$

where again $\mathit{\Pi}_{ij}$ is a NIZKPoK that this is done correctly. We have here chosen the version without publishing the encryption of $g^{r_{ij}}$, however this only changes for the proof.

Observe that we need a proof that an ElGamal ciphertext is raised to some known power and this accounts to a proof of knowledge of the randomness r in a DH-tuple. A NIZKPoK for it can be obtaining by appying the Fiat-Shamir's heuristic to the Chaum-Perdersen's proofs [CP93]. The coercion-resistance of the public information follows from the DH-assumption observing what follows. Let us assume for simplicity that there is only one teller. Then, the coercer can see $C_i^r g_i^n$ along with the ciphertext raised to r but not C_i and note also that the voter does *not* know r. Thus, under the DH-assumption we can conclude that this information consists of just random group elements.

By homomorphically multiplying $\{g^{n_i}\}_{\mathsf{pk}_T}$ with all the $\{C_i^{r_{ij}}\}_{\mathsf{pk}_T}$, we get the trapdoor commitment $\{g^{n_i} C_i^{r_i}\}_{\mathsf{pk}_T}$ where the trapdoor key now is c_i. The Tally Tellers decrypt these commitments verifiably.

Vote casting follows [Roe16] and works like before. The voter casts

$$(\{C_i\}_{\mathsf{pk}_T}\,, \{\text{vote}_i\}_{\mathsf{pk}_T}, \pi)$$

anonymously to BB. The difference is that the zero-knowledge proof now also contains a proof of knowledge of the discrete logarithm in the encrypted credential, i.e. like an anonymous signature.

Tallying is just like before, and retrieving the trackers likewise. However, for coerced voters, faking the random term $g^{r_{ij}}$ is now different from standard

Selene. The point is that, whereas in the standard case, the coerced voter will hand out the real secret key of dvk_i to the coercer, in this case the coercer will get a fake key $C'_i = g^{c'_i}$. The fake term g^{r_i} is thus calculated as

$$\left(g^{-n^*} g^{n_i} C_i^{r_i}\right)^{{c'}_i^{-1}}$$

since, when combining this with the commitment on BB, we get a ciphertext which decrypts to the wanted tracker n^* when we decrypt with the fake credential key given to the coercer. Actually, this construction is mildly better than standard Selene for coercion. The reason is that, if the coercer somehow manages to see the real term g^{r_i}, this will decrypt to the voter's correct tracker in standard Selene, but here it will decrypt to a random number, since the coercer is in the possession of a fake key. The voter can thus still claim that something must have gone wrong, or the system is corrupted, whereas in standard Selene the chance of this would be negligible. In real life, this is probably not a very usable defense for coerced voters.

Note that, if Tracker Tellers are corrupted, they can reveal relations on the credentials between voters from the decrypted commitments, since they know the random coins used in the commitments. This is however less of a problem in this version of JCJ since the discrete log of the credential is needed to break verifiability, and the Tracker Tellers are anyway trusted for coercion-resistance.

7.3 On the Secure Platform Problem

One of the main problems of e-voting is the secure platform problem. Very often this problem is ignored and the voter's computing platform is considered safe. An alternative useful approach is to use an out-of-band channel, e.g. using vote codes on paper, see e.g. Pretty Good Democracy [RT09].

Instead of resorting to out-of-band channels, one can also try to secure the device used by the voter, see e.g. [NV12] [NFVK13b] where simple smart cards are used. These are further used to improve usability for the voter. One drawback of dedicated hardware might be forced abstention attacks from local coercers, who simply seize the device from the coerced voter.

Instead, we can try to spread the risk of malware attacks to two independent devices, assuming that the adversary will not be able to control both. Further, we keep these devices general, i.e., it could be smartphones or laptops and not dedicated hardware. Keys could have backups on more devices if the voter is afraid of forced coercion. Due to the setup with two different credential/keys, the combination of JCJ and Selene (with two credential/keys) seems ideal for this task.

Let us assume that the voter has two computing devices D_1 and D_2. We store the secret key of the designated verifier key dvk_i on D_2. The voter now uses device D_1 for the registration where the voter gets the credential from the registration. The credential is then stored on D_1, and possibly with secure backups. Note that, during registration, only the public key dvk_i is needed, thus device D_2 can be excluded from this process.

A coerced voter can provide fake proofs without using device D_1, i.e., by only using the secret key on device D_2. Thus device D_2 does not learn the credential.

Vote-casting can be done on device D_1 since it holds the credential, but does not need device D_2. Finally, tracker retrieval and vote verification can be done on D_2 without using D_1.

In order to perform an undetected change of the vote, an adversary needs to infect both device D_1 to get the correct credential, and device D_2 in order to fake the verification of the final tallied vote with the Selene mechanism. Since the devices could be very independent, e.g. the check of the final vote could even be done on some public PC (with a threat of a privacy attack, of course), this seems to greatly reduce the danger from malware.

7.4 Using JCJ to Improve Selene

The combination of JCJ and Selene cannot only be used to add extra verifiability to JCJ, but can also provide a more secure tracker retrieval in Selene. The point is that the voter can authenticate herself with her credential. We can use this to make the tracker retrieval active. That is, instead of the Tracker Tellers sending out the $g^{r_{ij}}$ terms, with the risk of the coercer intercepting the message, the voter contacts the Tracker Tellers to obtain the terms. We will here briefly sketch the idea.

The voters can identify themselves to the Tracker Tellers with a ciphertext of the credential. Here and in the following such encrypted credentials should be followed by zero-knowledge proofs of plaintext knowledge of a special form that makes sure that it is not copied from e.g. already cast election ballots, or reused for ballots or authentication in later elections. For clarity we will suppress these proofs in the following. The Tally Tellers can now perform a PET with the registered credential (while also checking the zero-knowledge proof) to check the authenticity. After authentication, the terms $g^{r_{ij}}$ are handed out.

Coerced voters need to have a time window between the publication of the tally board and the start of the tracker retrieval, where they can upload a fake $g^{r_{ij}}$ term to each Tracker Teller. They do this via an anonymous channel

$$V_i \to TT_j : V_i, \{C_1\}_{\mathsf{pk}_T}, \{C_2\}_{\mathsf{pk}_T}, \{(g^{r_{ij}})_{\text{fake}}\}_{\mathsf{pk}_T} .$$

The first plaintext is supposed to be the real credential, the second plaintext the faked credential and the third plaintext is the faked term that will be shown, when someone with credential C_2 tries to retrieve their tracker share. The Tally Tellers need to be invoked to get this term. The fake term could also be sent in plain, if the channel is considered untappable for the coercer.

Now, if a coercer tries to retrieve the random term, the voter should have made a faking request beforehand, and the coercer gets the faked term.

However, we need to be careful since the coercer should not be able to use the update mechanism to discover that he is in the possession of a faked credential. We thus proceed as follows. After the time window, each Tracker Teller now has a database for each voter with rows of faking requests (which might come the coercer as well). For understandability, we assume that each voter has maximally one coercer, and we can then weed this list so that the value of the first credential C_1 can only appear once, copies are removed via PETs. A retrieval request now takes the form of $V_i\,, \{C\}_{\mathsf{pk}_T}$. The Tracker Teller now processes this request via the following algorithm which has two memory slots. Before beginning the ciphertext of the registered credential is loaded to memory slot 1 and the real value $g^{r_{ij}}$ is loaded to slot 2. For the given request it TT_j requests a PET of the submitted ciphertext with the value in memory slot one. If the PET is successful, it hands out the stored value in memory slot 2 and exits the algorithm. If not, it requests PETs against the database C_1s and the value in memory slot one. If there is no success, it hands out a random number and exits the algorithm, but if there is a success it stores the corresponding C_2 ciphertext in memory slot 1, and the fake value in memory slot 2, deletes the database row and reiterates the algorithm. The algorithm stops since the database is finite.

The coercer only has a negligible chance of guessing the real credential. Thus with overwhelming probability, if the coercer asks for tracker retrieval, the algorithm will after its first step to simulate that the credential, handed to the coercer by the voter, is the real one with a corresponding faked term. In this way the retrieval mechanism will act as if the coercer has a real credential. Note that timing might be a side channel attack here, so some default delay is required in the response time.

The advantage of the system is that also coerced voters can safely do verification of their votes, the disadvantage is a rather complicated system, and the voter still needs to be active to fake their trackers.

8 Conclusions and future work

We have shown that it is possible to use the Selene mechanism in JCJ, providing an e-voting protocol where voters can individually check that their vote was counted as intended, while still preserving a good level of coercion-resistance. Further, several alternatives were presented providing: better verifiability (while only handling a single key), everlasting privacy, a more secure tracker retrieval and better protection against malware on the voters' computing devices. Also improvements to Selene, in terms of efficiency, and JCJ, in terms of coercion-resistance, were presented.

This paper did not provide formal proofs of the security guarantees. These are currently under consideration for the classical Selene protocol in the UC framework, and should later be extended to also include this work.

Two main problems of JCJ were not touched upon, namely, efficiency and usability. Regarding usability, Selene, in some sense is a step backwards. The users (in the first version of the protocol at least) needs to handle two keys post-registration. And coerced voter have to careful when they retrieve the trackers. Further investigations are necessary to determine to which extent this can be handled by the voter assisting devices, and if the extra clarity and trust given by the check of the final vote will outweigh this. We however, also plan to increase the usability of JCJ in the future by allowing the voters to use short codes. Finally regarding efficiency, the versions of JCJ presented here still suffer from the tally time being quadratic in the number of voters, a problem we will also try to solve in future a work.

References

ACKR13. Myrto Arapinis, Véronique Cortier, Steve Kremer, and Mark Ryan. Practical everlasting privacy. In David A. Basin and John C. Mitchell, editors, *Principles of Security and Trust - Second International Conference, POST 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings*, volume 7796 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2013.

Adi08. Ben Adida. Helios: Web-based open-audit voting. In *Proc. 17th USENIX Security Symposium*, pages 335–348, 2008.

Ben06. 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.

BFM88. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 103–112. ACM Press, May 1988.

BR93. 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.

CCM08. Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: A secure voting system. In *In IEEE Symposium on Security and Privacy*, 2008.

Cha01. D. Chaum. Surevote: Technical overview. In *Proc. Workshop on Trustworthy Elections (WOTE'01)*, 2001.

CP93. 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.

CPP13. Edouard Cuvelier, Olivier Pereira, and Thomas Peters. Election verifiability or ballot privacy: Do we need to choose? In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*, volume 8134 of *Lecture Notes in Computer Science*, pages 481–498. Springer, 2013.

Fre00. Michael J Freedman. Design and analysis of an anonymous communication channel for the free haven project. *Online: http://www. freehaven. net/doc/comm. ps*, 2000.

Gam85. Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 31(4):469–472, 1985.

GRCC15. 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.

HH07. Delfs Hans and Knebl Helmut. Intorduction to cryptography–principles and applications, 2007.

HK14. R. Haenni and R. Koenig. *Design, Development, and Use of Secure Electronic Voting Systems*, chapter Voting over the Internet on an Insecure Platform. IGI Global, March 2014.

HL09. James Heather and David Lundin. *The Append-Only Web Bulletin Board*, pages 242–256. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

JCJ05. 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.

JJ00. Markus Jakobsson and Ari Juels. Mix and match: Secure function evaluation via ciphertexts. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 162–177. Springer, 2000.

KHF11. Reto E. Koenig, Rolf Haenni, and Stephan Fischli. Preventing board flooding attacks in coercion-resistant electronic voting schemes. In Jan Camenisch, Simone Fischer-Hübner, Yuko Murayama, Armand Portmann, and Carlos Rieder, editors, *Future Challenges in Security and Privacy for Academia and Industry - 26th IFIP TC 11 International Information Security Conference, SEC 2011, Lucerne, Switzerland, June 7-9, 2011. Proceedings*, volume 354 of *IFIP Advances in Information and Communication Technology*, pages 116–127. Springer, 2011.

KMST16. Ralf Küsters, Johannes Mueller, Enrico Scapin, and Tomasz Truderung. select: A lightweight verifiable remote voting system. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*, pages 341–354. IEEE Computer Society, 2016.

KOKV11. Fatih Karayumak, Maina M. Olembo, Michaela Kauer, and Melanie Volkamer. Usability analysis of helios - an open source verifiable remote electronic voting system. In *Proc. Electronic Voting Technology Workshop / Workshop on Trustworthy Elections (EVT/WOTE'11)*, 2011.

KR16. Steve Kremer and Peter B. Rønne. To du or not to du: A security analysis of du-vote. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 473–486. IEEE, 2016.

MN06. Tal Moran and Moni Naor. Receipt-free universally-verifiable voting with everlasting privacy. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 373–392. Springer, 2006.

NFVK13a. Stephan Neumann, Christian Feier, Melanie Volkamer, and Reto E. Koenig. Towards A practical JCJ / civitas implementation. *IACR Cryptology ePrint Archive*, 2013:464, 2013.

NFVK13b. Stephan Neumann, Christian Feier, Melanie Volkamer, and Reto E. Koenig. Towards A practical JCJ / civitas implementation. In Matthias

Horbach, editor, *Informatik 2013, 43. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Informatik angepasst an Mensch, Organisation und Umwelt, 16.-20. September 2013, Koblenz, Deutschland*, volume 220 of *LNI*, pages 804–818. GI, 2013.

NV12. Stephan Neumann and Melanie 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, 2012.

Ped91. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pages 129–140, 1991.

Roe16. Peter B. Roenne. JCJ with improved verifiability guarantees. In *The International Conference on Electronic Voting E-Vote-ID 2016 : 18–21 October 2016, Lochau/Bregenz, Austria : Proceedings*, 2016.

RRI15. Peter Y. A. Ryan, Peter B. Rønne, and Vincenzo Iovino. Selene: Voting with transparent verifiability and coercion-mitigation. *IACR Cryptology ePrint Archive*, 2015:1105, 2015.

RRI16. Peter Y. A. Ryan, Peter B. Rønne, and Vincenzo Iovino. Selene: Voting with transparent verifiability and coercion-mitigation. In Jeremy Clark, Sarah Meiklejohn, Peter Y. A. Ryan, Dan S. Wallach, Michael Brenner, and Kurt Rohloff, editors, *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, volume 9604 of *Lecture Notes in Computer Science*, pages 176–192. Springer, 2016.

RT09. Peter Y. A. Ryan and Vanessa Teague. Pretty good democracy. In Bruce Christianson, James A. Malcolm, Vashek Matyas, and Michael Roe, editors, *Security Protocols XVII, 17th International Workshop, Cambridge, UK, April 1-3, 2009. Revised Selected Papers*, volume 7028 of *Lecture Notes in Computer Science*, pages 111–130. Springer, 2009.

Sch94. Bruce Schneier. Applied cryptography, 1994.

SK95. Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 393–403. Springer, 1995.