Towards a Mechanized Proof of Selene Receipt-Freeness and Vote-Privacy

Alessandro Bruni, Eva Drewsen, and Carsten Schürmann*

IT University of Copenhagen
{albr, evdr, carsten}@itu.dk

Abstract. Selene is a novel voting protocol that supports individual verifiability, Vote-Privacy and Receipt-Freeness. The scheme provides tracker numbers that allow voters to retrieve their votes from a public bulletin board and a commitment scheme that allows them to hide their vote from a potential coercer. So far, however, Selene was never studied formally. The Selene protocol was neither completely formalized, nor were the correctness proofs for Vote-Privacy and Receipt-Freeness.
In this paper, we give a formal model for a simplified version of Selene in the symbolic model, along with a machine-checked proof of Vote-Privacy and Receipt-Freeness. All proofs are checked with the Tamarin theorem prover.

1 Introduction

The original motivation of the Selene voting protocol [17] was to design a voting protocol that is verifiable, usable, and guarantees Vote-Privacy (VP) and Receipt-Freeness (RF). Selene's hallmark characteristics is that it does not require voters to check their votes on a bulletin board using long hashes of encrypted ballots, but instead works with readable and memorisable tracker numbers.

Selene is a voting protocol that could, at least in theory, be used in binding elections. One way to increase the confidence in its correctness is to use formal methods. The more complex a protocol the more likely are design mistakes, and the earlier such mistakes can be found and fixed, the better it is for all stakeholders involved. Selene uses an ElGamal crypto system, two independent phases of mixing, Pedersen style trap-door commitments and zero-knowledge proofs of knowledge.

In this paper we apply Tamarin to mechanize the proofs of correctness for VP and RF for Selene. First, we model Selene in the Tamarin language. The first model corresponds to the original Selene protocol described in Section 2. Using

* This work was funded in part through the Danish Council for Strategic Research, Programme Comission on Strategic Growth Technologies under grant 10-092309. This publication was also made possible by NPRP grant NPRP 7-988-1-178 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

Tamarin, we prove VP and RF, but only under the assumptions that coercer and voters do not collude. Tamarin constructs a counter example otherwise. We then strengthen the model according to a fix that was already described in the original Selene paper and then show VP and RF. Tamarin can no longer find a counter example.

For the purpose of mechanization, we also develop a precise message sequence chart for full Selene, which we then simplify further to become suitably representable in Tamarin. Tamarin is described in Section 3. While working with Tamarin, we also discovered a few completeness issues with the implementation of Tamarin that are currently being worked on by the Tamarin team.

Contributions We describe a formalisation of Selene. A description of the full formalization can be found in [9]. We propose a simplified model of Selene, where explicit mixing is replaced by random multi-set reductions. We formalise our simplified model of Selene in Tamarin, and express the properties of VP and RF in our model. We describe the counter example and the modified model for which show VP and RF. All Tamarin proof scripts can be found at `https://github.com/EvaSleva/selene-proofs`.

Related work The extended version of the original Selene paper [17] includes in the appendix a partial argument of correctness of the main construction, however it does not provide a formal proof of the scheme. Other voting protocols have undergone formal analysis, such as the FOO [11], Okamoto [15] and Lee et al. [13], which have been analysed in [7]. An analysis of Helios [2] is presented in [5] and of the Norwegian e-voting protocol [12] in [6]. The arguments are partly formalized, for example in the applied π-calculus [1] and the theorem prover ProVerif [4]. Recently the Dreier et al. [8] extended the equational reasoning of multiset rewrite rules in Tamarin, which have been pivotal for our development, and applied this technique to the analyses of the FOO and Okamoto protocols.

Organization This paper is organized as follows. In Section 2 we describe the full Selene voting protocol as described in [17] as message sequence charts. In Section 3 we give a brief introduction into the Tamarin tool and explain syntactic categories and the Tamarin rewrite engine. In Section 4, we describe then the two Tamarin models, and present the result of mechanizing the proofs of VP and RF. Finally, we conclude and assess results in Section 7.

2 The Selene voting protocol

The purpose of Selene is to construct a receipt-free scheme ensuring individual verifiability, i.e. voters can check that their vote is tallied in the final result. Selene achieves this by giving each user a *tracker number* that will point to their vote on a public bulletin board, containing all cast votes. The scheme maintains *vote-privacy*, since none of the involved parties—besides the voters themselves—learns who cast each vote; *individual verifiability*, since the protocol gives a proof

to the voters that their vote has been tallied; and *receipt-freeness*, since voters have no way of proving how they voted to a potential coercer, because they can fake the proof that should convince the coercer how they voted.

The involved parties in the protocol are the *Voters* (V_i); an *Election Authority* (EA) responsible for checking their identities and issuing the tracking numbers; a *Web Bulletin Board* (WBB) that publishes the intermediate stages of the voting process, as well as the anonymized, decrypted votes; a *Mixnet* (M) performing distributed re-encryption of the tracking numbers and the votes; and a number of *Tellers* (T), performing distributed threshold decryption of the votes.

Tracker numbers must remain unlinkable to the voter from the perspective of the various parties involved; at the same time, one must be assured that each voter is given a distinct tracker number. This problem is solved by the distributed Mixnet carrying along proofs of re-encryption. The Tellers produce a Pedersen-style commitment for each voter to their tracker number, also in a distributed fashion; they also decrypt the votes and tracker numbers, which are finally posted publicly on the Bulletin Board. To ensure that the computations are performed correctly, non-interactive zero knowledge proofs are carried throughout the protocol.

2.1 Re-encryption Mixnets and Pedersen-style Commitments

At the heart of the Selene protocol are two useful properties of the ElGamal cryptosystem, which we now briefly review: it can perform randomized re-encryptions and can act as a commitment scheme.

The ElGamal encryption scheme operates under a cyclic group $\mathbb{G}$ of order q and generator g. For any given private key $sk \in \mathbb{Z}_q$; the corresponding public key is $pk = g^{sk}$; the encryption of a message $m \in \mathbb{G}$ intended for the owner of sk is the ElGamal tuple $(\alpha, \beta) = (g^r, m \cdot pk^r)$ given a uniformly random choice of $r \in \mathbb{Z}_q$; finally, decryption is performed by computing $m = \frac{\beta}{\alpha^{sk}}$.

Re-encryption Given an encryption pair $(\alpha, \beta) = (g^r, m \cdot pk^r)$ with $m \in \mathbb{G}$ and uniformly random $r \in \mathbb{Z}_q$, one can compute a randomized re-encryption of m without knowing the secret key sk. Computing the pair $(\alpha \cdot g^{r'}, \beta \cdot pk^{r'})$ with uniformly random $r' \in \mathbb{Z}_q$, is equal to $(g^{(r+r')}, m \cdot pk^{(r+r')})$ and hence decrypts to the same value of m while being indistinguishable from the former encryption without having the secret key sk. Shuffling mixnets chain this sort of encryption on a set of encrypted values, such that if at least one link in the mixnet is kept secret, the final cipher texts will be unlinkable to the original input, albeit encrypting the same values.

Pedersen-style commitments ElGamal cryptosystems also allow to commit to a message without revealing it right away. Given a message $m \in \mathbb{M}$, with $\mathbb{M} \subseteq \mathbb{G}$ and small $|\mathbb{M}|$ (e.g. the number of random tracker numbers chosen for the election), one computes the commitment of m as $\beta = g^m \cdot pk^r$ for some randomly uniform $r \in \mathbb{Z}_q$. To reveal the message m, simply output $\alpha = g^r$ and compute $\gamma = \frac{\beta}{\alpha^{sk}} = g^m$, then check against all $m' \in \mathbb{M}$ to find the matching $g^{m'} = \gamma$.

The Pedersen-style commitment scheme constitutes the core of the individual verifiability and receipt-freeness in Selene. First, the voters can be convinced that the protocol behaved correctly by committing in advance to the tracking number that will be linked to their vote, which they can publicly check. It is in fact believed to be hard to compute a reveal message α' that decrypts to a different m' knowing only m, pk and r, as it reduces to solving the discrete logarithm problem. Most importantly, the voters (knowing sk and α) can construct the fake α' by computing $\alpha' = g^{\frac{m-m'}{sk}} \cdot \alpha$. This makes it practically impossible for anyone but the voter to construct a fake receipt, thus the voters can trust that the protocol behaved correctly, whereas a potential coercer cannot trust any receipt from the voter. For a more detailed explanation of Pedersen-style commitment schemes we refer to the original paper [16].

2.2 Protocol steps

We will now explain the main steps of Selene. Before voting begins, each voter V_i must be given a tuple on the WBB containing their public key, the encrypted tracker number and the trap door commitment to the tracker number: $(pk_i, \{g^{n_i}\}_{pk_T}, \beta)$. This process is as follows:

Set up All voters are assumed to have their public/secret key pairs: $(pk_i = g^{sk_i}, sk_i)$. The election authority publicly creates unique tracker numbers n_i for each voter, computes g^{n_i} and the ElGamal encryption under the teller's public key: $\{g^{n_i}\}_{pk_T}$. These terms are posted on the WBB:

$$(n_i, g^{n_i}, \{g^{n_i}\}_{pk_T})$$

These are put through a sequence of verifiable re-encryption mixes and the shuffled numbers are assigned to the voters and posted on the WBB:

$$(pk_i, \{g^{n_{\pi(i)}}\}_{pk_T})$$

Since the numbers have gone through multiple mixes, no single teller knows this assignment. The shuffling is verifiable however, so it preserves the original tracker numbers.

Creation of trap-door commitments The trapdoor commitments are created in a distributed fashion among the tellers. For each voter i, each teller j creates a fresh random $r_{i,j}$, computes $\{g^{r_{i,j}}\}_{pk_T}$ and $\{pk_i^{r_{i,j}}\}_{pk_T}$. For each voter, the product of the second elements are formed:

$$\{pk_i^{r_i}\}_{pk_T} = \prod_{j=1}^{t} \{pk_i^{r_{i,j}}\}_{pk_T}$$

where by exploiting the multiplicative homomorphic property of ElGamal:

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

Then the product of $\{pk_i^{r_i}\}_{pk_T}$ and $\{g^{n_{\pi(i)}}\}_{pk_T}$ is formed to obtain the encrypted trapdoor commitment: $\{pk_i^{r_i} \cdot g^{n_{\pi(i)}}\}_{pk_T}$. The commitments are decrypted and posted on the WBB along with the voter's identity and the encrypted tracker number:

$$(pk_i, \{g^{n_{\pi(i)}}\}_{pk_T}, (pk_i^{r_i} \cdot g^{n_{\pi(i)}}), \qquad)$$

All these steps with proofs and audits are also posted. The last entry is left blank for the vote. The tellers keep their $g^{r_{i,j}}$ terms secret for now.

Voting Each voter encrypts and signs their vote $Sign_{V_i}(\{Vote_i\}_{pk_T})$ and sends it along with a proof of knowledge of the plaintext. The signature and proof are needed to ensure "ballot independence" [19] and to prevent an attacker copying the vote as their own. The server checks for duplication, checks proofs and pairs off the vote with the key corresponding to the private key which it was signed. The entry on the WBB now looks like this:

$$(pk_i, \{g^{n_{\pi(i)}}\}_{pk_T}, (pk_i^{r_i} \cdot g^{n_{\pi(i)}}), Sign_{V_i}(\{Vote_i\}_{pk_T}))$$

Decryption and tabulation For each row on the WBB, the second and fourth terms (which are the tracker and vote) are taken out and put through verifiable, parallel, re-encryption mixes, and threshold decrypted. We then have a list of pairs: $(g^{n_{\pi(i)}}, Vote_i)$ from which the tracker can be derived:

$$(n_{\pi(i)}, Vote_i)$$

Revealing the trackers After the trackers and votes have been available for a suitable amount of time, the voter receives the $g^{r_{i,j}}$ terms from all the tellers through a private channel and combines them to form g^{r_i}, which is the α term of the ElGamal encryption under the voters' PKs. The $g^{n_{\pi(i)}} \cdot pk_i^{r_i}$ posted earlier is the β component, and the voter can now form the ElGamal cryptogram: $(g^{r_i}, g^{n_{\pi(i)}} \cdot pk_i^{r_i})$, which they can decrypt with their secret key to reveal $g^{n_{\pi(i)}}$ and hence $n_{\pi(i)}$.

In case of coercion, it is easy for the voter to compute an alternative $(g^{r'_i})$, which will open the encryption to any tracker number they would like. However, this is hard to do without the secret key, so it would not be feasible for an attacker to reveal the wrong tracker to the voter. Due to space limitations, we refer to [9] for the full formalization of Selene.

3 Tamarin

Tamarin is a specialised theorem prover for security protocols based on labelled multiset rewriting. It can prove both trace properties and equivalence properties on labelled transition systems. Tamarin supports convergent equational theories and the AC theories for Diffie-Hellman and multisets [8, 18]. In Tamarin models, multiset rewriting rules encode both the protocol specification and the Dolev-Yao attacker. Because of its multiset semantics, the tool supports precise analysis of protocols with state and unbounded numer of sessions, however at the cost of non-termination, since the problem is undecidable in general.

Definition 1 (Rules, facts, terms, equational theories and semantics). *A* term *is either a variable x or a function $f(t_1, \ldots, t_n)$ of fixed arity n, with $t_1, \ldots, t_n$ terms. Equality of terms $=_E$ is defined as the smallest reflexive, symmetric and transitive closure of a user-defined* equational theory *E. Variables are annotated with a* sort system*, with a top generic sort msg and two incompatible sub-sorts: ˜ for nonces and $ for public messages.* Facts *are of the form $\mathsf{F}(t_1, \ldots, t_n)$, with fixed arity n and $t_1, \ldots, t_n$ terms. There are six* reserved fact symbols*: Fr for fresh nonces; In and Out for protocol input and output; KU, KD and K for attacker knowledge. All other facts are* user-defined*.* Persistent facts *are prefixed with the ! (bang) modality, and can be consumed arbitrarily often.*

A labelled multiset rewrite rule *is a rule of the form $l -[a]\!\!\rightarrow r$, where the multisets of facts l, a and r represent the rule premise, label and conclusion. We omit the brackets when $a = \emptyset$. A* state *S is a multiset of facts. We define the semantics of a rule $l -[a]\!\!\rightarrow r$ as the relation $S \xrightarrow{\sigma(a)} S'$ with substitution σ where $\sigma(l) \subseteq_E S$ and $S' = S \setminus_E \sigma(l) \uplus \sigma(r)$.*

Functions and equations model cryptography symbolically: for example asymmetric encryption and decryption with the equation $adec(aenc(m, pk(sk)), sk) =_E m$, saying that the encryption of m using $pk(sk)$ only succeeds with the corresponding secret key sk. In Section 4.2 we present a more advanced equational theory that covers the commitment and re-encryption schemes used in Selene.

Observational Equivalences Tamarin supports both trace-based properties and observational equivalence in the models. Trace-based peoperties suffice to model secrecy and authentication. However in this work we focus on observational equivalences, i.e. show that an adversary cannot distinguish between two systems, specified by the left and the right projections of special terms $diff(t_1, t_2)$ occurring in the model. To define observational equivalence we split the rules into *system rules* (*Sys*), *environment rules* (*Env*) and *interface rules* (*IF*). In the following, $\mathcal{F}^\#$ and $\mathcal{G}^\#$ are finite multisets of facts and ground facts, while ρ is the set of all rule recipes. For a detailed definition of these concepts see [3].

Definition 2 (Observational Equivalence). *Two sets of multiset rewrite rules S_A and S_B are observational equivalent with respect to an environment Env, written $S_A \approx_{Env} S_B$, if, given the labelled transition system defined by the rules $S_A \cup IF \cup Env$ and $S_B \cup IF \cup Env$, there is a relation $\mathcal{R}$ containing the initial states, such that for all states $(S_A, S_B) \in \mathcal{R}$ we have:*

- *If $S_A \xrightarrow[a]{r} S'_A$ and r is the recipe of a rule in $Env \cup IF$, then there exists action $a' \in \mathcal{F}^\#$, and $S'_B \in \mathcal{G}^\#$, such that $S_B \xrightarrow[a']{r} S'_B$, and $(S'_A, S'_B) \in \mathcal{R}$.*
- *If $S_A \xrightarrow[a]{r} S'_A$ and r is the recipe of a rule in S_A, then there exist recipes $r_1, \ldots, r_n \in \rho$ of rules in S_B, actions $a_1, \ldots, a_n \in \mathcal{F}^\#$, $n \geq 0$, and $S'_B \in \mathcal{G}^\#$, such that $S_B \xrightarrow[a_1]{r_1} \ldots \xrightarrow[a_n]{r_n} S'_B$, and $(S'_A, S'_B) \in \mathcal{R}$.*
- *We have the same in the other direction.*

Tamarin does not directly prove observational equivalences: it rather proves a stronger notion of equivalence, called *mirroring of dependency graphs*. A *dependency graph* is defined as a graph where the nodes are ground rule instances, and there is a directed arc from a rule r_1 to r_2 iff r_1 produces a fact that is consumed by r_2. Furthermore, it is required that all input facts have incoming edges, that non-persistent facts are consumed at most once, and that exactly one rule has no outgoing edges (the goal rule). *Mirroring* is defined as an isomorphism between two graphs modulo the equational theory. Let $mirrors(dg)$ denote the mirrors of a dependency graph dg.

To prove mirroring, Tamarin constructs all possible instantiations of dependency graphs for the left- and right-hand side systems, and shows that for each dependency graph on one side, there exists one on the other that mirrors it. If such construction is possible, then we say that the two systems are *dependency graph equivalent*. We will not dive into the details however, but rather present the essential result of the proof technique that this paper uses.

Theorem 1 (Dependency graph equivalence implies observational equivalence).
Let S be a bi-system. If $L(S) \sim_{DG,Env} R(S)$ then $L(S) \approx_{Env} R(S)$

Our proofs of vote-privacy and receipt-freeness in Selene rely on constructing two systems using *diff*-terms and then checking whether *mirroring* holds. For a full explanation of Tamarin and the techniques we just briefly covered, we refer to the official documentation [20] and research papers [18, 14, 3, 8].

4 Selene Tamarin model

Assumptions To make the protocol amenable to formal verification we have made the following assumptions in the model: firstly, we assume that all the participants in the protocol behave honestly, except the attacker and the voter being coerced. Selene claims to be secure even under partially compromised Tellers and Mixnets, by using threshold and distributed encryption schemes. The original Selene paper does not commit to specific schemes in this regard, and these are complex verification problems out of reach for current symbolic theorem provers like Tamarin. Instead of explicitly modeling them, we only model their defining features and assume a proper implementation. Furthermore, our voting system is restricted to two voters and two candidates. For modeling vote-privacy and receipt-freeness properties in voting systems two voters are enough [7], hence this restriction does not pose further limitations in the analysis.

Simplifications to the protocol Given said assumptions we have made the following simplifications to our model. Since the protocol is honest, we do not need to model zero knowledge proofs, e.g. reencryption proofs in Selene: by the model each entity executes the protocol, hence no proof of computation is needed. Only one teller is modelled, as multiple tellers in Selene are used

to perform cryptographic operations (e.g. reencryption of the tracker numbers) without having access to the decrypted data so that one dishonest teller cannot link a decrypted tracker number to the voter. Since we assume that the tellers are honest, we do not need more than one. Similarly we do not model the mixnet explicitly, and we take advantage of the non-determinism of Tamarin rules, and of the associative-commutative multiset operators, to ensure that the link is lost between the identities of the voters and their tracker numbers, at least from the perspective of the attacker. In our model, every occurrence of mixing is replaced with a synchronisation point and a multiset operation. Finally, whenever an authentic and confidential channel is required in the protocol, we model that with a linear fact that is not accessible to the attacker, whereas if the channel is only authentic, but public, the attacker receives the value separately.

4.1 The protocol

We model our protocol after the scheme presented in Figure 1. As discussed earlier, the Mixnet does not appear in our simplified scheme, whose behaviour we model with the non-deterministic semantic of rewrite rules. We assume that each voter V_i has a public key pk_i, and the teller public key is pk_T. Initially the EA generates a unique tracker number for each voter V_i, and publishes them through the WBB. With respect to the full version, we omit the zero-knowledge proofs, the re-encryption mixing and the distributed decryption, as we assume to trust the Tellers, Mixnet and the Election Authority.

The teller then creates a commitment to the tracker number for each voter with their public key, and posts them on the WBB. Voters can now cast their vote by sending to their teller the encrypted and signed choice. After voting has ended, the encrypted votes are posted on the WBB along with the voter identity and commitment. Again the model omits the second pass of re-encryption randomisation by the mixnet, and posts directly the decrypted vote and tracker number on the WBB.

After a suitable period, the randomness of the commitments is sent to the voters. The voters can then combine this with their commitment to find their tracker number, and check the corresponding vote on the WBB.

4.2 Modelling Approach

Channels Selene assumes the existence of secure and authentic communication channels. We model these by the use of linear facts that ensure a correspondence between inputs and outputs, and add a corresponding Out fact when the communication is also public, e.g.:

$$\mathsf{Fr}(\tilde{}x) \rightarrow \mathsf{SendValue}(\tilde{}x), \mathsf{Out}(\tilde{}x) \qquad \text{(AuthCh)}$$

We use this approach as an alternative to explicitly modelling encryption in public channels: this has the advantage of greatly reducing the search space.

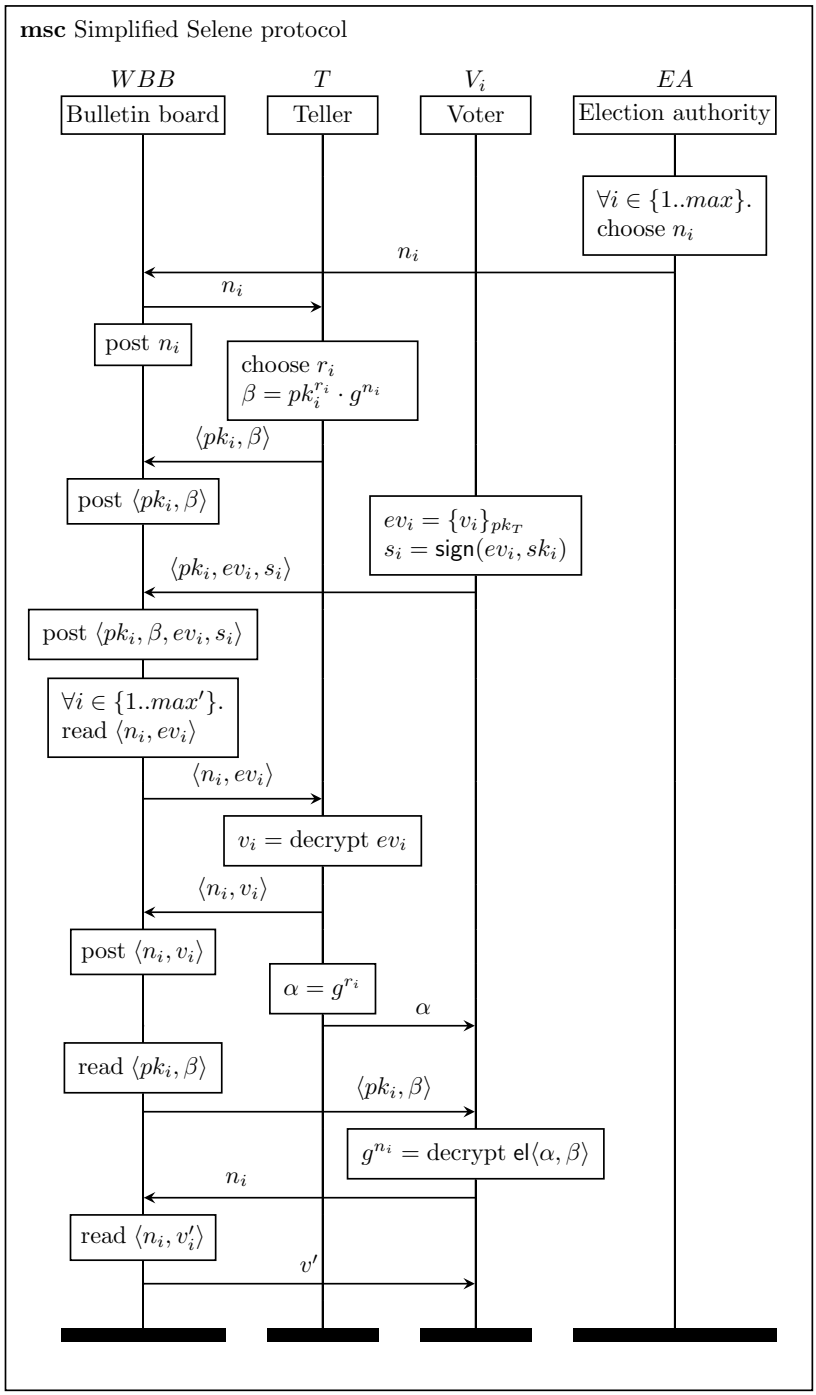

Fig. 1. Simplified Selene protocol

Equations Trap-door commitments are the central cryptographic primitive of the Selene protocol. We model them with the functions $commit/3$, $open/3$ and $fake/4$. The term $commit(m, r, pk)$ models a commitment to value m using the randomness r and the public key $pk = pk(sk)$. To open the commitment one applies $open(commit(m, r, pk(sk)), r, sk)$. Those in possession of the secret key sk can construct a receipt $fake(m, r, sk, m_2)$ for another message m_2, and it should hold that $open(commit(m, r, pk(sk)), fake(m, r, sk, m_2), sk) = m_2$. Thus we have:

$$open(commit(n_1, r, pk(sk)), r, sk) \to n_1$$
$$commit(n_2, fake(n_1, r, sk, n_2), pk(sk)) \to commit(n_1, r, pk(sk))$$

These equations do not produce a confluent rewriting system, and it is therefore not convergent, and this can cause Tamarin to produce false results. We use the Maude Church-Rosser checker to produce their Knuth-Bendix completion and get:

$$open(commit(n_1, r, pk(sk)), fake(n_1, r, sk, n_2), sk) \to n_2$$

However adding this equation still does not make it confluent, since the checker keeps finding new and larger critical pairs whenever the resulting equation is added. In order to fix this we add the equation:

$$fake(n_2, fake(n_1, r, sk, n_2), sk, n_3) \to fake(n_1, r, sk, n_3)$$

Using the Maude Church-Rosser checker this is now confirmed to yield a confluent rewriting system. [10]

Shuffling When the full Selene protocol of Section 2 requires shuffling the votes through a re-encryption mixnet, we use multisets to model the reordering, for example:

$$\begin{aligned} &\mathsf{SendTracker}(n_1), \mathsf{SendTracker}(n_2) \to \\ &\quad \mathsf{!PublishTrackers}(n_1 + n_2), \mathsf{Out}(n_1 + n_2) && \text{(Shuffle)} \\ &\mathsf{!PublishTrackers}(n_1' + n_2') \to && \text{(Receive)} \end{aligned}$$

In Tamarin, the AC symbol + denotes multiset union. It is therefore possible to match the two rules (Shuffle) and (Receive) both with the substitution $\{n_1/n_1', n_2/n_2'\}$ and $\{n_1/n_2', n_2/n_1'\}$, making irrelevant the order of inputs and outputs. Furthermore, this rule acts as a synchronization point.

4.3 Basic Model

In this section, we describe the common rules between the models used for the vote-privacy and receipt-freeness proofs of Sections 5 and 6. Our models have a fixed set of agents, that is two voters V_1 and V_2 and a teller T. Rule (Setup)

generates the keys for both agents, and outputs the corresponding public keys. The state of each agent is initialised with a predicate $\mathsf{St_{X0}}$ where X denotes the agent type and the arguments denote their knowledge, including—for the voters—their choice of candidates specified using *diff* terms. Two teller instances are created to interact with each of the voters. The fact EA starts the generation of the random tracker numbers by the Election Authority, as described by the next rule.

$$\begin{aligned}&\mathsf{Fr}(\tilde{}sk_{V_1}), \mathsf{Fr}(\tilde{}sk_{V_2}), \mathsf{Fr}(\tilde{}sk_T) \text{–}[\mathsf{OnlyOnce}]\rightarrow \\ &\quad \mathsf{Out}(pk(\tilde{}sk_{V_1})), \mathsf{Out}(pk(\tilde{}sk_{V_2})), \mathsf{Out}(pk(\tilde{}sk_T)), \\ &\quad \mathsf{St_{V0}}(\mathrm{V}_1, \mathit{diff}(\mathrm{A}, \mathrm{B}), \tilde{}sk_{V_1}, pk(\tilde{}sk_{V_1})), \\ &\quad \mathsf{St_{V0}}(\mathrm{V}_2, \mathit{diff}(\mathrm{B}, \mathrm{A}), \tilde{}sk_{V_2}, pk(\tilde{}sk_{V_2})), \\ &\quad \mathsf{St_{T0}}(\mathrm{T}, \tilde{}skT, pk(\tilde{}sk_{V_1})), \\ &\quad \mathsf{St_{T0}}(\mathrm{T}, \tilde{}skT, pk(\tilde{}sk_{V_2})), \\ &\quad \mathsf{EA}(pk(\tilde{}sk_{V_1}), pk(\tilde{}sk_{V_2}))\end{aligned} \tag{Setup}$$

In rule (EA) the Election Authority generates the tracker numbers n_i and outputs them publicly. The trackers are shuffled using the + operator, and the pk_Vs are used to ensure that both voters don't get assigned the same tracker.

$$\begin{aligned}&\mathsf{EA}(pk_{V_1}, pk_{V_2}), \mathsf{Fr}(\tilde{}n_1), \mathsf{Fr}(\tilde{}n_2) \rightarrow \\ &\quad !\mathsf{ShuffleTrackers}(\langle \tilde{}n_1, pk_{V_1}\rangle + \langle \tilde{}n_2, pk_{V_2}\rangle), \\ &\quad \mathsf{Out}(\tilde{}n_1, \tilde{}n_2)\end{aligned} \tag{EA}$$

Rule (T1) represents the teller receiving one shuffled tracker n_i from the multiset produced by EA. The teller assigns n_i to a voter V_i by creating a commitment to its value with the voter's public key and a newly generated random value α, then stored in the state fact. The commitment is published on the WBB using both $\mathsf{PostCommitment}$ and an Out fact:

$$\begin{aligned}&\mathsf{let}\ \beta_i = \mathit{commit}(n_i, \alpha_i, pk_{V_i})\ \mathsf{in} \\ &!\mathsf{ShuffleTrackers}(\langle n_i, pk_{V_i}\rangle + y), \mathsf{Fr}(\alpha_i), \mathsf{St_{T0}}(T, \tilde{}sk_T, pk_{V_i}) \rightarrow \\ &\quad \mathsf{Out}(\langle pk_{V_i}, \beta_i\rangle), !\mathsf{PostCommitment}(pk_{V_i}, \beta_i), \\ &\quad \mathsf{St_{T1}}(T, \tilde{}sk_T, pk_{V_i}, \alpha_i, n_i, \beta_i)\end{aligned} \tag{T1}$$

Rule (V1) enacts the voting stage. To simplify the model and since we assume a trusted teller, voting is represented by a predicate $\mathsf{SendVote}$ that includes the choice and the teller's public key:

$$\mathsf{St_{V0}}(V_i, v_i, \tilde{}sk_{V_i}, pk_{V_i}) \rightarrow \mathsf{SendVote}(v_i, pk_{V_i}), \mathsf{St_{V1}}(V_i, v_i, \tilde{}sk_{V_i}, pk_{V_i}) \tag{V1}$$

Rules (T2) and (T2-Sync) represent the teller receiving the two votes cast, revealing to each voter the randomness to recover their tracker numbers, synchronising

and outputting both pairs of vote and tracker number publicly.

$$\begin{aligned}
&\mathsf{SendVote}(v_i, pk_{V_i}), \mathsf{St_{T1}}(T, \tilde{}sk_T, pk_{V_i}, \alpha_i, n_i, \beta_i) \rightarrow \\
&\quad \mathsf{SendSecretToVoter}(\alpha_i), \mathsf{PassVote}(v_i, n_i) && \text{(T2)} \\
&\mathsf{PassVote}(v_1, n_1), \mathsf{PassVote}(v_2, n_2) \rightarrow \\
&\quad !\mathsf{PublishVote}(\langle n_1, v_1\rangle + \langle n_2, v_2\rangle), \mathsf{Out}(\langle n_1, v_1\rangle + \langle n_2, v_2\rangle) && \text{(T2-Sync)}
\end{aligned}$$

Finally, rule (V2) models the checking phase, where the voter retrieves their commitment and their secret randomness to compute their tracker number. The rule also checks that the vote is posted correctly on the bulletin board by requiring the presence of the corresponding tuple in *PublishVote*.

$$\begin{aligned}
&\mathsf{let}\ n_i = open(\beta_i, \alpha_i, \tilde{}sk_{V_i})\ \mathsf{in} \\
&\mathsf{SendSecretToVoter}(\alpha_i), !\mathsf{PostCommitment}(pk_{V_i}, \beta_i), \\
&!\mathsf{PublishVote}(\langle n_i, v_i\rangle + y), \mathsf{St_{V1}}(V_i, v_i, \tilde{}sk_{V_i}, pk_{V_i}) \rightarrow && \text{(V2)}
\end{aligned}$$

5 Vote-privacy

Vote-privacy is the basic requirement to any electronic voting system, where running the protocol should not reveal the intention of each voter. Obviously one cannot simply model vote-privacy as an observational equivalence property where one voter votes in two possible ways and the rest remains unchanged, since the result would show up in the final tally. Instead, Delaune et al. [7] define vote-privacy as an equivalence between two systems where two voters V_1 and V_2 swap their choices of candidates A and B. The public outcome of the election remains unchanged, hence an attacker must observe the difference between the two systems from other information that is exchanged throughout the election.

The model introduced in Section 4.3 is sufficient to prove vote-privacy of Selene: it produces two systems where the two candidates V_1 and V_2 swap their votes A and B, using the *diff* terms. Tamarin can prove mirroring automatically, hence by Theorem 1 we conclude that they are observationally equivalent.

6 Receipt-freeness

Receipt-freeness is a stronger property than vote-privacy. To be receipt-free, a protocol must not reveal the choice of a voter even when the voter reveals all their private information to an attacker [7].

Selene claims to be receipt-free as long as the underlying vote-casting scheme is receipt-free. The extra information the voter has in Selene is a commitment to the tracking number linked to their vote, and a receipt that opens the commitment. However each voter can fake their own receipt hence the attacker cannot infer from the voter's private information whether the receipt is fake or real [17].

6.1 Modelling

Like in vote-privacy, the model shows two systems in which two voters swap votes. However, V_1 always outputs a receipt for A regardless of how they voted. The coercer should not be able to determine that V_1 is producing fake receipts. We modify the model of Section 4.3 by splitting the rule (V2) into the rules (V2-1) and (V2-2). Rule (V2-2) is identical to (V2) for voter V_2 and only checks that their vote appears on the WBB, while (V2-1) outputs the secret information of V_1 (the coerced voter), including their secret key, the desired tracker number, and either a fake or a real receipt:

$$
\begin{aligned}
&\mathsf{let}\ n_1 = \mathit{open}(\beta, \alpha, \tilde{}sk_V)\ \mathsf{in}\\
&\mathsf{SendSecretToVoter}(\alpha), \mathsf{!PostCommitment}(pk_V, \beta),\\
&\mathsf{!PublishVote}(\langle n_1, v\rangle + \langle n_2, \mathit{diff}(\mathrm{B}, \mathrm{A})\rangle),\\
&\mathsf{St_{V1}}(\mathrm{V}_1, v, \tilde{}sk_V, pk_V) \rightarrow\\
&\quad \mathsf{Out}(\tilde{}sk_V), \mathsf{Out}(\beta),\\
&\quad \mathsf{Out}(\mathit{diff}(n_1, n_2)),\\
&\quad \mathsf{Out}(\mathit{diff}(\mathit{fake}(n_1, \alpha, \tilde{}sk_V, n_1), \mathit{fake}(n_1, \alpha, \tilde{}sk_V, n_2)),\\
&\quad \mathsf{Out}(\mathit{diff}(v, \mathrm{A}))
\end{aligned}
\qquad \text{(V2-1)}
$$

Here, V_1 checks for their vote, as well V_2's vote, and saves this tracker number as n_2. In the first system V_1 actually voted for A as the coercer wanted, and outputs the real receipts. In the second system V_1 voted for B, but outputs fake receipts, as if they voted for A. In the rule's conclusion V_1 outputs all the available values, which are: the private key $\tilde{}sk_V$, the commitment β, the tracker number n_1, or n_2, the secret randomness, and finally either the actual vote or the fake vote.

The randomness is a *fake* function that either opens the commitment to the voter's real tracker number n_1, or to the other tracker number n_2. As a modeling expedient we use the term $\mathit{fake}(n_1, \alpha, \tilde{}sk_V, n_1)$ to denote the real receipt for V_1, instead of simply α. This is required because Tamarin converts the directed equational theories into rewrite rules, and hence the rules produced for the *fake* constructors would not apply to the basic αs. Using the model just presented Tamarin automatically proves receipt-freeness for the protocol.

Attack The scheme poses a problem if the voter accidentally picks the coercer's own tracker number, or a tracker number of another voter under coercion. This does not reveal the voter's actual vote, but the coercer will know the voter is lying. This is a known flaw in the protocol and is also explained in the paper presenting Selene. We can reproduce this attack in our model by changing the rule (V2-2) to also output V_2's tracker number:

$$
\begin{aligned}
&\mathsf{let}\ n_1 = \mathit{open}(\beta, \alpha, \tilde{}sk_V)\ \mathsf{in}\\
&\mathsf{SendSecretToVoter}(\alpha), \mathsf{!PostCommitment}(pk_V, \beta),\\
&\mathsf{!PublishVote}(\langle n_1, v\rangle + y), \mathsf{St_{V1}}(\mathrm{V}_2, v, \tilde{}sk_V, pk_V) \rightarrow \mathsf{Out}(n_1)
\end{aligned}
\qquad \text{(V2-2)}
$$

In fact, if the adversary knows both tracker numbers, then they can compare the trackers and see that they match when V_1 chooses to fake their receipt, while they differ when V_1 behaves honestly, violating the observational equivalence.

Fix The authors of Selene [17] proposed a construction that removes the possibility of voters picking the coercer's tracker number. Each voter v gets $|C|$ additional tracker numbers that point to fake votes cast for each candidate of the possible choices C. The bulletin board contains $|C| \cdot v + v$ tracker-vote pairs, and computing the final tally amounts to removing $|C| \cdot v$ votes to each candidate. If a voter is being coerced and wants to reveal a fake receipt, they only need to pick one of their fake trackers that points to the desired candidate.

We model this fix in a simplified version of our original model, where we remove the non-determinism in the shuffling that contributes to space explosion. Tamarin could not terminate within 16 hours on the full version using a server with 16 Intel Xeon cores and 120GB of RAM. The partial model has no EA and does not output the trackers or the pairs on a public WBB. It works as follows: the teller generates three tracker numbers for each voter, but only creates a commitment to the first one, which will act as the voter's real tracker number.

$$\begin{aligned}
&\mathsf{let}\ \beta = \mathit{commit}(n_0, \alpha_i, pk_{V_i})\ \mathsf{in}\\
&\mathsf{Fr}(n_0), \mathsf{Fr}(n_1), \mathsf{Fr}(n_2), \mathsf{Fr}(\alpha_i),\\
&\mathsf{St_{T0}}(T, \tilde{}sk_T, pk_{V_i}, v_i') \rightarrow\\
&\qquad \mathsf{Out}(\langle pk_{V_i}, \beta_i \rangle), !\mathsf{PostCommitment}(pk_{V_i}, \beta_i),\\
&\qquad \mathsf{St_{T1}}(T, \tilde{}sk_T, pk_{V_i}, v_i', \alpha_i, n_0, n_1, n_2)
\end{aligned} \tag{T1'}$$

After receiving the vote, the teller assigns each tracker to a vote. The real vote is therefore assigned to the voter's real tracker number and to one other tracker number. All $|C| + 1$ trackers are published along with their corresponding vote. For each voter $|C|$ trackers are published along with the voter's identity, so the voter can use these in case of coercion. The extra tracker, that also points to the voter's actual cast vote, is removed from set, and therefore not published with the voter's identity.

$$\begin{aligned}
&\mathsf{SendVote}(v_i, pk_{V_i}), \mathsf{St_{T1}}(T, \tilde{}sk_T, pk_{V_i}, v_i', \alpha_i, n_0, n_1, n_2) \rightarrow\\
&\qquad \mathsf{SendSecretToVoter}(\alpha_i),\\
&\qquad !\mathsf{PublishTracker}(pk_{V_i}, n_0), !\mathsf{PublishTracker}(pk_{V_i}, n_2),\\
&\qquad !\mathsf{PublishVote}(n_0, v_i), !\mathsf{PublishVote}(n_1, v_i), !\mathsf{PublishVote}(n_2, v_i')
\end{aligned} \tag{T2'}$$

The difference in the checking phase is that V_1 needs to check the WBB for their personal tracker number corresponding to the coercer's desired candidate. Both votes can then be checked using the voter's two tracker numbers. The process is

unchanged for voter V_2.

$$
\begin{aligned}
&\mathsf{let}\ n_0 = \mathit{open}(\beta, \alpha, \tilde{}sk_V)\ \mathsf{in}\\
&\mathsf{SendSecretToVoter}(\alpha), \mathsf{!PostCommitment}(pk_V, \beta),\\
&\mathsf{!PublishTracker}(pk_V, n_F), \mathsf{!PublishVote}(n_0, v), \mathsf{!PublishVote}(n_F, \mathit{diff}(\mathrm{B}, \mathrm{A})),\\
&\mathsf{St_{V1}}(\mathrm{V}_1, v, \tilde{}sk_V, pk_V) \rightarrow\\
&\qquad \mathsf{Out}(\tilde{}sk_V), \mathsf{Out}(\beta),\\
&\qquad \mathsf{Out}(\mathit{diff}(n_0, n_F)),\\
&\qquad \mathsf{Out}(\mathit{diff}(\mathit{fake}(n_0, \alpha, \tilde{}sk_V, n_0), \mathit{fake}(n_0, \alpha, \tilde{}sk_V, n_F)),\\
&\qquad \mathsf{Out}(\mathit{diff}(v, \mathrm{A}))
\end{aligned}
\qquad \text{(V2-1')}
$$

As with the previous RF model, we can prove that the two systems produced by this model satisfy mirroring, therefore issuing a fake certificate for each candidate allows us to prove receipt-freeness even when the attacker knows the other voter's vote. This alternative protocol ensures a stronger type of receipt-freeness in which other voters' receipts can also be revealed.

7 Conclusions

In this work we built mechanised proofs receipt-freeness and vote-privacy for Selene, which claims to also offer individual and universal verifiability. Selene uses re-encryption mixnets and Pedersen-style commitment schemes, which lead to complex equational theories that were out of reach for many cryptographic theorem provers, including ProVerif [4]. We overcame the limitation on mixing by using the AC multiset operator of Tamarin, and built a confluent equational theory for the commitment scheme used in Selene.

Our models show that the Selene scheme preserves vote-privacy and that it is receipt-free as long as the fake receipts do not match the choice of another colluding voter. We also model the proposed fix, whereby each voter receives a fake tracker numbers for each candidate. These proofs confirm the claims of the original paper [17], albeit under the stricter condition that the Tellers, Mixnet and Election Authority are honest and not compromised. These restrictions were necessary since distributed re-encryptions and decryptions produce state explosions that makes it infeasible to find a proof, even when running on a virtual server with 16 Intel Xeon cores and 120 GB of RAM.

We believe that this study contributes to a better understanding of Selene, and to discover necessary conditions for its security, such as the synchronisation points required after the setup and the voting phases. As future work, it will be interesting to explore how to relax the assumption that all principals behave honestly and introduce zero-knowledge proofs to ensure their correct behaviour. Also, this study has not considered universal verifiability: while the protocol maintains individual verifiability—and that can be checked as a correspondence property in our current model—checking universal verifiability requires combining Selene with other receipt-free, universally verifiable schemes.

References

1. Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *ACM SIGPLAN Notices*, volume 36, pages 104–115. ACM, 2001.
2. Ben Adida. Helios: Web-based open-audit voting. In *USENIX security symposium*, volume 17, pages 335–348, 2008.
3. David Basin, Jannik Dreier, and Ralf Sasse. Automated symbolic proofs of observational equivalence. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1144–1155. ACM, 2015.
4. Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. *The Journal of Logic and Algebraic Programming*, 75(1):3–51, 2008.
5. Véronique Cortier and Ben Smyth. Attacking and fixing helios: An analysis of ballot secrecy. *Journal of Computer Security*, 21(1):89–148, 2013.
6. Véronique Cortier and Cyrille Wiedling. A formal analysis of the norwegian e-voting protocol. *Journal of Computer Security*, 25(1):21–57, 2017.
7. Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4):435–487, 2009.
8. Jannik Dreier, Charles Duménil, Steve Kremer, and Ralf Sasse. Beyond subterm-convergent equational theories in automated verification of stateful protocols. In *6th International Conference on Principles of Security and Trust (POST)*, 2017.
9. Eva Drewsen. Formal analysis of the selene voting protocol. Master's thesis, IT University of Copenhagen, June 2017.
10. Francisco Durán and José Meseguer. On the church-rosser and coherence properties of conditional order-sorted rewrite theories. *The Journal of Logic and Algebraic Programming*, 81(7-8):816–850, 2012.
11. Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *Advances in CryptologyAUSCRYPT'92*, pages 244–251. Springer, 1993.
12. Kristian Gjøsteen. The norwegian internet voting protocol. In *International Conference on E-Voting and Identity*, pages 1–18. Springer, 2011.
13. Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. *Information Security and Cryptology-ICISC 2003*, pages 245–258, 2004.
14. Simon Meier. *Advancing automated security protocol verification*. PhD thesis, ETH Zürich, 2013.
15. Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Security Protocols*, pages 25–35. Springer, 1998.
16. Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual International Cryptology Conference*, pages 129–140. Springer, 1991.
17. Peter YA Ryan, Peter B Rønne, and Vincenzo Iovino. Selene: Voting with transparent verifiability and coercion-mitigation. In *International Conference on Financial Cryptography and Data Security*, pages 176–192. Springer, 2016.
18. Benedikt Schmidt. *Formal analysis of key exchange protocols and physical protocols*. PhD thesis, Citeseer, 2012.
19. Ben Smyth and David Bernhard. Ballot secrecy and ballot independence: definitions and relations. Technical report, Cryptology ePrint Archive, Report 2013/235 (version 20141010: 082554), 2014.
20. The Tamarin Team. *Tamarin-Prover Manual*, April 2017.