Attacking and fixing Helios: An analysis of ballot secrecy

Véronique Cortier
Loria, CNRS & INRIA Nancy Grand Est, France

Ben Smyth
Loria, CNRS & INRIA Nancy Grand Est, France

***Abstract*—Helios 2.0 is an open-source web-based end-to-end verifiable electronic voting system, suitable for use in low-coercion environments. In this paper, we analyse ballot secrecy and discover a vulnerability which allows an adversary to compromise the privacy of voters. This vulnerability has been successfully exploited to break privacy in a mock election using the current Helios implementation. Moreover, the feasibility of an attack is considered in the context of French legislative elections and, based upon our findings, we believe it constitutes a real threat to ballot secrecy in such settings. Finally, we present a fix and show that our solution satisfies a formal definition of ballot secrecy using the applied pi calculus.**

***Keywords*-Applied Pi Calculus, Attack, Ballot Independence, Ballot Secrecy, Electronic Voting, Helios, Privacy.**

I. Introduction

Paper-based elections derive privacy properties from physical characteristics of the real-world, for example, the indistinguishability of an individual's ballot from an arbitrary ballot, and the inability of a coercer to collaborate with a voter inside a polling booth. Replicating these attributes in a digital setting has proven to be difficult and, hence, the provision of electronic voting systems which ensure the privacy of voters is an active research topic [1], [2], [3].

Informally, privacy for electronic voting systems is characterised by the following properties [4], [5], [6]:

- *Ballot secrecy.* A voter's vote is not revealed to anyone.
- *Receipt freeness.* A voter cannot gain information which can be used to prove, to a coercer, how she voted.
- *Coercion resistance.* A voter cannot collaborate, with a coercer, to gain information which can be used to prove how she voted.

Other desirable properties of electronic voting systems include [3], [7], [8]:

- *Fairness:* All votes are independently cast.
- *Individual verifiability:* A voter can check that her own ballot is published on the election's bulletin board.
- *Universal verifiability:* Anyone can check that all the votes in the election outcome correspond to ballots published on the election's bulletin board.

The fairness property prohibits the voting system from influencing a voter's vote; more formally, this requires that observation of the voting system (that is, observing interaction between participants) does not leak information that may affect a voter's decision. One aspect of fairness is *ballot independence*, which based upon [9, §1.1] can be informally stated as: observing another voter's interaction with the election system does not allow a voter to cast a *related* vote. The individual and universal verifiability properties (also called *end-to-end verifiability* [3], [10], [11], [7], [12]) allow voters and election observers to verify – independently of the hardware and software running the election – that votes have been recorded, tallied and declared correctly. In this paper, we analyse ballot secrecy in Helios 2.0 [13].

Helios is an open-source web-based electronic voting system. The scheme is claimed to satisfy ballot secrecy [13], but the nature of remote voting makes the possibility of satisfying stronger privacy properties difficult, and Helios does not satisfy receipt freeness nor coercion resistance. In addition to ballot secrecy, the system provides end-to-end verifiability (cf. [8], [14] and [15, Chapter 3] for an analysis of end-to-end verifiability in Helios). Helios is particularly significant due to its real-world deployment: the International Association of Cryptologic Research used Helios to elect its board members [16], following a successful trial in a non-binding poll [17]; the Catholic University of Louvain adopted the system to elect the university president [13]; and Princeton University used Helios to elect the student vice president [18].

Formal definitions of ballot secrecy have been introduced in the context of the applied pi calculus by Delaune, Kremer & Ryan [4], [5], [19], [20] and Backes, Hriţcu & Maffei [6]. These privacy definitions consider two voters $\mathcal{A}$, $\mathcal{B}$ and two candidates t, t'. Ballot secrecy is captured by the assertion that an adversary (controlling arbitrary many dishonest voters) cannot distinguish between a situation in which voter $\mathcal{A}$ votes for candidate t and voter $\mathcal{B}$ votes for candidate t', from another situation in which $\mathcal{A}$ votes t' and $\mathcal{B}$ votes t. This can be expressed by the following equivalence.

$$\mathcal{A}(t) \mid \mathcal{B}(t') \approx_l \mathcal{A}(t') \mid \mathcal{B}(t)$$

These formal definitions of ballot secrecy have been used by their respective authors to analyse the electronic voting protocols due to: Fujioka, Okamoto & Ohta [1], Okamoto [2], Lee *et al.* [21], and Juels, Catalano & Jakobsson [3], [22], [23]. It therefore seems natural to check whether Helios satisfies ballot secrecy as well.

Contribution: Our analysis of Helios reveals an attack which violates ballot secrecy. The attack exploits the system's lack of ballot independence, and works by replaying a voter's ballot or a variant of it (without knowing the vote contained within that ballot). Replaying a voter's ballot immediately violates ballot secrecy in an election with three voters. For example, consider the electorate Alice, Bob, and Mallory; if Mallory replays Alice's ballot, then Mallory can reveal Alice's vote by observing the election outcome and checking which candidate obtained at least two votes. The practicality of this attack has been demonstrated by violating privacy in a mock election using the current Helios implementation. Furthermore, the vulnerability can be exploited in more realistic settings and, as an illustrative example, we discuss the feasibility of the attack in French legislative elections. This case study suggests there is a plausible threat to ballot secrecy. We also propose a variant of the attack which abuses the malleability of ballots to ensure replayed ballots are distinct: this makes identification of replayed ballots non-trivial (that is, checking for exact duplicates is insufficient). Nonetheless, we fix the Helios protocol by identifying and discarding replayed ballots. We believe this solution is particular well-suited because it maintains Benaloh's principle of ballot casting assurance [24], [25] and requires a minimal extension to the Helios code-base. Finally, we show that the revised scheme satisfies a formal definition of ballot secrecy using the applied pi calculus.

Related work: The concept of independence was introduced by Chor *et al.* [26] and the possibility of compromising security properties due to lack of independence has been considered, for example, by [27], [28], [29], [30]. In the context of electronic voting, Gennaro [9] demonstrates that the application of the Fiat-Shamir heuristic in the Sako-Kilian electronic voting protocol [31] violates ballot independence, and Wikström [32], [33] studies non-malleability for mixnets to achieve ballot independence. By comparison, we focus on the violation of ballot secrecy rather than fairness, and exploit the absence of ballot independence to compromise privacy. Similar results have been shown against mixnets [34].

Estehghari & Desmedt [35] claim to present an attack which undermines privacy and end-to-end verifiability in Helios. However, their attack is dependent on compromising a voter's computer, a vulnerability which is explicitly acknowledged by the Helios specification [13]: *"a specifically targeted virus could surreptitiously change a user's vote and mask all of the verifications performed via the same computer to cover its tracks."* Accordingly, [35] represents an exploration of known vulnerabilities rather than an attack.

Langer *et al.* [36], [37] and Volkamer & Grimm [38] also study privacy in Helios. Langer *et al.* propose a taxonomy of informal privacy requirements [36], [37], [39] to facilitate a more fine-grained comparison of electronic voting systems; this framework is used to analyse Helios and the authors claim ballot secrecy is satisfied if the adversary only has access to public data [36], [37]. Volkamer & Grimm introduce the *k-resilience* metric [38], [40] to calculate the number of honest participants required for ballot secrecy in particular scenarios; this framework is used to analyse Helios and the authors claim ballot secrecy is satisfied if the software developers are honest and the key holders do not collude [38]. Contrary to these results, we show an attack against privacy. Our work highlights the necessity for rigorous mathematical analysis techniques for security protocols; in particular, we believe the erroneous results reported by Langer *et al.* were due to the use of informal methods, and the approach by Volkamer & Grimm failed because only some particular scenarios were considered.

Structure of this paper: Section II presents the Helios electronic voting scheme. (We remark that this is the first cryptographic description of the Helios protocol in the literature and, hence, is an additional contribution of this paper.) Section III describes our attack and some variants, in addition to a study of its feasibility in the context of French legislative elections. We propose several solutions for recovering privacy in Section IV and prove that our adopted solution formally satisfies ballot secrecy in Section V.

II. Background: Helios 2.0

Helios exploits the additive homomorphic [41], [42], [43] and distributed decryption [44], [45] properties of ElGamal [46]. We will recall these cryptographic details before presenting the Helios protocol.

A. Additive homomorphic ElGamal

Given cryptographic parameters (p, q, g) and a number $n \in \mathbb{N}$ of trustees, where p and q are large primes such that $q \mid p-1$ and g is a generator of the multiplicative group $\mathbb{Z}_p^*$ of order q, the following operations are defined by ElGamal.

Distributed key generation: Each trustee $i \in n$ selects a private key share $x_i \in_R \mathbb{Z}_q^*$ and computes a public key share $h_i = g^{x_i} \bmod p$. The public key is $h = h_1 \cdot \ldots \cdot h_n \bmod p$.

Encryption: Given a message m and a public key h, select a random nonce $r \in_R \mathbb{Z}_q^*$ and derive the ciphertext $(a, b) = (g^r \bmod p,\ g^m \cdot h^r \bmod p)$.

Re-encryption: Given a ciphertext (a, b) and public key h, select a random nonce $r' \in_R \mathbb{Z}_q^*$ and derive the re-encrypted ciphertext $(a', b') = (a \cdot g^{r'} \bmod p,\ b \cdot h^{r'} \bmod p)$.

Homomorphic addition: Given two ciphertexts (a, b) and (a', b'), the homomorphic addition of plaintexts is computed by multiplication $(a \cdot a' \bmod p,\ b \cdot b' \bmod p)$.

Distributed decryption: Given a ciphertext (a, b), each trustee $i \in n$ computes the partial decryption $k_i = a^{x_i}$. The plaintext $m = \log_g M$ is recovered from $M = b/(k_1 \cdot \ldots \cdot k_n) \bmod p$.

The computation of a discrete logarithm $\log_g M$ is hard in general. However, if M is chosen from a restricted domain, then the complexity is reduced; for example, if M is an integer such that $0 \leq M \leq n$, then the complexity is $O(n)$ by linear search or $O(\sqrt{n})$ using the baby-step giant-step algorithm [47] (see also [48, §3.1]).

For secrecy, each trustee $i \in n$ must demonstrate knowledge of a discrete logarithm $\log_g h_i$, that is, they proof that h_i has been correctly constructed; this prevents, for example, a trustee constructing their public key share $h_i = h$. For integrity of decryption, each trustee $i \in n$ must demonstrate equality between discrete logarithms $\log_g h_i$ and $\log_a k_i$; this prevents, for example, a trustee constructing the public key share $h_i = g^{m+x_i}$ and providing the partial decryption $k_i = a^{x_i}$. In addition, the voter must demonstrate that a valid vote has been encrypted. These proofs can be achieved using signatures of knowledge (see Appendix A for details).

B. Protocol description

An election is created by naming an election officer, selecting a set of trustees, and generating a distributed public key pair. The election officer publishes, on the bulletin board, the public part of the trustees' key (and proof of correct construction), the candidate list $\tilde{t} = (t_1, \ldots, t_l) \cup \{\epsilon\}$ (where ϵ represents a vote of abstention), and the list of eligible voters $\widetilde{id} = (id_1, \ldots, id_n)$; the officer also publishes the *election fingerprint*, that is, the hash of these parameters. Informally, the steps that participants take during a run of Helios are as follows.

1) The voter launches a browser script that downloads the election parameters and recomputes the election fingerprint. The voter should verify that the fingerprint corresponds to the value published on the bulletin board. (This ensures that the script is using the trustees' public key; in particular, it helps prevent encrypting a vote with an adversary's public key. Such attacks have been discussed in the context of Direct Anonymous Attestation by Rudolph [49]; although, the vulnerability was discounted, in the trusted computing setting, by Leung, Chen & Mitchell [50].)
2) The voter inputs her vote $v \in \tilde{t}$ to the browser script, which creates a ballot consisting of her vote encrypted by the trustees' public key, and a proof that the ballot represents a permitted vote (this is needed because the ballots are never decrypted individually, in particular, it prevents multiple votes being encoded as a single ballot). The ballot is displayed to the voter.
3) The voter can audit the ballot to check if it really represents a vote for her chosen candidate; if she decides to do this, then the script provides her with the random data used in the ballot creation. She can then independently reconstruct her ballot and verify that it is indeed well-formed. The script provides some practical resistance against vote selling by refusing to cast audited ballots. See Benaloh [24], [25] for further details on ballot auditing.
4) When the voter has decided to cast her ballot, the script submits it to the election officer. The election officer authenticates the voter and checks that she is eligible to vote. The election officer also verifies the proof and publishes the ballot, appended with the voter's identity id, on the bulletin board. (In practice, the election officer also publishes the hash of the ballot, we omit this detail for brevity.)
5) Individual voters can check that their ballots appear on the bulletin board and, by verifying the proof, observers are assured that ballots represent permitted votes.
6) After some predefined deadline, the election officer homomorphically combines the ballots and publishes the encrypted tally on the bulletin board. Anyone can check that tallying is performed correctly.
7) Each of the trustees publishes a partial decryption of the encrypted tally, together with a signature of knowledge proving the partial decryption's correct construction. Anyone can verify these proofs.
8) The election officer decrypts the tally and publishes the result. Anyone can check this decryption.

Figure 1 Ballot construction by the browser script

Input: Cryptographic parameters (p, q, g), public key h, candidate list $\tilde{t} = (t_1, \ldots, t_l) \cup \{\epsilon\}$ and vote v.

Output: Encrypted vote $(a_1, b_1), \ldots, (a_l, b_l)$, signatures of knowledge $(\bar{a}_1, \bar{b}_1, \bar{c}_1, \bar{s}_1, \bar{a}'_1, \bar{b}'_1, \bar{c}'_1, \bar{s}'_1), \ldots, (\bar{a}_l, \bar{b}_l, \bar{c}_l, \bar{s}_l, \bar{a}'_l, \bar{b}'_l, \bar{c}'_l, \bar{s}'_l)$ and signature of knowledge $(\bar{a}, \bar{b}, \bar{c}, \bar{s}, \bar{a}', \bar{b}', \bar{c}', \bar{s}')$.

1) If $v \notin \tilde{t}$ then the script terminates.
2) Encode the vote v as a bitstring. For all $1 \leq i \leq l$, let
$$m_i = \begin{cases} 1 & \text{if } v = t_i \\ 0 & \text{otherwise} \end{cases}$$
3) The bitstring representing the vote is encrypted. For all $1 \leq i \leq l$, let
$$(a_i, b_i) = (g^{r_i} \bmod p,\ g^{m_i} \cdot h^{r_i} \bmod p)$$
where $r_i \in_R \mathbb{Z}_q^*$.
4) For all $1 \leq i \leq l$, let $(\bar{a}_i, \bar{b}_i, \bar{c}_i, \bar{s}_i, \bar{a}'_i, \bar{b}'_i, \bar{c}'_i, \bar{s}'_i)$ be a signature of knowledge demonstrating that the ciphertext (a_i, b_i) contains either 0 or 1, that is, each candidate can receive at most one vote.
5) Let $(\bar{a}, \bar{b}, \bar{c}, \bar{s}, \bar{a}', \bar{b}', \bar{c}', \bar{s}')$ be a signature of knowledge demonstrating that the ciphertext $(a_1 \cdot \ldots \cdot a_l,\ b_1 \cdot \ldots \cdot b_l)$ contains either 0 or 1, that is, at most one candidate receives one vote.

Formally, Step 2 is defined in Figure 1. (For simplicity the ballot construction algorithm in Figure 1 considers a vote $v \in \tilde{t}$, this can be generalised [13] to consider a vote $\tilde{v} \subseteq \tilde{t}$.) Checking voter eligibility (Step 4) is beyond the scope of Helios and Adida *et al.* [13] propose the use of existing infrastructure. The remaining steps follow immediately from the application of cryptographic primitives (see Section II-A for details).

C. Software implementation

Helios 3.0 is an extension of Helios 2.0 which adds numerous practical features, including: integration of authentication with various web-services (for example, Facebook, GMail and Twitter), bulk voter registration using pre-existing electoral rolls, and simplification of administration with multiple trustees. Helios 3.0 has been implemented and is publicly available: http://heliosvoting.org/.

III. Attacking ballot secrecy

Ballot secrecy means a voter's vote is not revealed to anyone. We show that the Helios protocol does not satisfy this definition of ballot secrecy, by presenting an attack which allows an adversary to reveal a voter's vote. Moreover, we will show that formal definitions of ballot secrecy [4], [19], [6] are also violated.

Intuitively, an adversary may identify a voter's ballot on the bulletin board (using the voter's id) and recast this ballot by corrupting dishonest voters. The multiple occurrences of the voter's ballot will leak information in the tally and the adversary can exploit this knowledge to violate the voter's privacy. An informal description of the attack will now be presented in the case of three eligible voters and Section III-C considers a more realistic setting. (A formal analysis appears in Section IV.)

A. Attack description

Let us consider an election with candidates $t_1, \ldots, t_l$ and three eligible voters who have identities id_1, id_2 and id_3. Suppose that voters id_1, id_2 are honest and id_3 is a dishonest voter controlled by the adversary. Further assume that the honest voters have cast their ballots. The bulletin board entries are as follows:

$$\begin{gathered} id_1, ciph_1, spk_1, spk'_1 \\ id_2, ciph_2, spk_2, spk'_2 \end{gathered}$$

where for $i \in \{1, 2\}$ we have

$$\begin{array}{rcl} ciph_i & = & (a_{i,1}, b_{i,1}), \ldots, (a_{i,l}, b_{i,l}) \\ spk_i & = & (\bar{a}_{i,1}, \bar{b}_{i,1}, \bar{c}_{i,1}, \bar{s}_{i,1}, \bar{a}'_{i,1}, \bar{b}'_{i,1}, \bar{c}'_{i,1}, \bar{s}'_{i,1}), \\ & & \ldots, (\bar{a}_{i,l}, \bar{b}_{i,l}, \bar{c}_{i,l}, \bar{s}_{i,l}, \bar{a}'_{i,l}, \bar{b}'_{i,l}, \bar{c}'_{i,l}, \bar{s}'_{i,l}) \\ spk'_i & = & (\bar{a}_i, \bar{b}_i, \bar{c}_i, \bar{s}_i, \bar{a}'_i, \bar{b}'_i, \bar{c}'_i, \bar{s}'_i) \end{array}$$

The value $ciph_i$ is the ith voter's encrypted vote, spk_i demonstrates that ciphertexts $(a_{i,1}, b_{i,1}), \ldots, (a_{i,l}, b_{i,l})$ contain either 0 or 1, and spk'_i demonstrates that $(a_{i,1} \cdot \ldots \cdot a_{i,l},\ b_{i,1} \cdot \ldots \cdot b_{i,l})$ contains either 0 or 1.

Exploiting the absence of ballot independence: The adversary observes the bulletin board and selects $ciph_k$, spk_k, spk'_k such that id_k is the voter whose privacy will be compromised, where $k \in \{1, 2\}$. The adversary submits the ballot $ciph_k, spk_k, spk'_k$ and it immediately follows that the bulletin board is composed as follows:

$$\begin{gathered} id_1, ciph_1, spk_1, spk'_1 \\ id_2, ciph_2, spk_2, spk'_2 \\ id_3, ciph_k, spk_k, spk'_k \end{gathered}$$

It is trivial to see that each bulletin board entry represents a permitted vote; that is, $spk_1, spk'_1, spk_2, spk'_2, spk_k, spk'_k$ all contain valid signatures of knowledge.

We have informally shown that Helios does not satisfy ballot independence (observing another voter's interaction with the election system allows a voter to cast the *same* vote), and this will now be exploited to violate privacy.

Violating privacy: The homomorphic addition of ballots reveals the encrypted tally $(a_{1,1} \cdot a_{2,1} \cdot a_{k,1},\ b_{1,1} \cdot b_{2,1} \cdot b_{k,1}), \ldots, (a_{1,l} \cdot a_{2,l} \cdot a_{k,l},\ b_{1,l} \cdot b_{2,l} \cdot b_{k,l})$ and, given the partial decryptions, these ciphertexts can be decrypted to reveal the number of votes for each candidate. Since there will be at least two votes for the candidate voter id_k voted for, the voter's vote can be revealed and hence privacy is not preserved. Moreover, the vote of the remaining honest voter will also be revealed.
A video demonstrating the attack against the Helios 3.0 implementation has been produced [51].

B. Variants exploiting malleability and key reuse

In the aforementioned attack description, the ballots cast by two voters are identical which may result in the detection of an attack. For a covert attack, the adversary may prefer to cast a distinct ballot. This can be achieved by exploiting the malleability of ballots. In particular, given a valid ballot

$$\begin{aligned} &(a_1, b_1), \ldots, (a_l, b_l), \\ &\quad (\bar{a}_1, \bar{b}_1, \bar{c}_1, \bar{s}_1, \bar{a}'_1, \bar{b}'_1, \bar{c}'_1, \bar{s}'_1), \ldots, \\ &\quad\quad (\bar{a}_l, \bar{b}_l, \bar{c}_l, \bar{s}_l, \bar{a}'_l, \bar{b}'_l, \bar{c}'_l, \bar{s}'_l), \\ &\quad\quad\quad (\bar{a}, \bar{b}, \bar{c}, \bar{s}, \bar{a}', \bar{b}', \bar{c}', \bar{s}') \end{aligned} \tag{B1}$$

the following ballots are also valid

$$\begin{aligned} &(a_1, b_1), \ldots, (a_l, b_l), \\ &\quad (\bar{a}_1, \bar{b}_1, \bar{c}_1, \bar{s}_1 + q, \bar{a}'_1, \bar{b}'_1, \bar{c}'_1, \bar{s}'_1 + q), \ldots, \\ &\quad\quad (\bar{a}_l, \bar{b}_l, \bar{c}_l, \bar{s}_l + q, \bar{a}'_l, \bar{b}'_l, \bar{c}'_l, \bar{s}'_l + q), \\ &\quad\quad\quad (\bar{a}, \bar{b}, \bar{c}, \bar{s} + q, \bar{a}', \bar{b}', \bar{c}', \bar{s}' + q) \end{aligned} \tag{B2}$$

$$\begin{aligned} &(a_{\pi(1)}, b_{\pi(1)}), \ldots, (a_{\pi(l)}, b_{\pi(l)}), \\ &(\bar{a}_{\pi(1)}, \bar{b}_{\pi(1)}, \bar{c}_{\pi(1)}, \bar{s}_{\pi(1)}, \bar{a}'_{\pi(1)}, \bar{b}'_{\pi(1)}, \bar{c}'_{\pi(1)}, \bar{s}'_{\pi(1)}), \ldots, \\ &\quad (\bar{a}_{\pi(l)}, \bar{b}_{\pi(l)}, \bar{c}_{\pi(l)}, \bar{s}_{\pi(l)}, \bar{a}'_{\pi(l)}, \bar{b}'_{\pi(l)}, \bar{c}'_{\pi(l)}, \bar{s}'_{\pi(l)}), \\ &\quad\quad (\bar{a}, \bar{b}, \bar{c}, \bar{s}, \bar{a}', \bar{b}', \bar{c}', \bar{s}') \end{aligned} \tag{B3}$$

where π is an arbitrary permutation over $\{1, \ldots, l\}$. Ballot B2 adds q to the response components of Ballot B1, this changes the ballot but not the vote. (It is also possible to modify a subset of the response components.) However, this might be considered an implementation bug in Helios

3.0, rather than a theoretical attack, because the ballots are identical if considered as group elements. Replaying Ballot B3 has the advantage of casting a theoretical distinct ballot, since Ballot B3 represents a vote for a different candidate (with the exception of an abstention vote), and it is possible to compromise ballot secrecy in elections with three voters without abstention votes; however, more than one (modified) ballot may be required in elections with abstention votes. This variant of our attack also demonstrates a further violation of ballot independence in Helios: observing another voter's interaction with the election system allows a voter to cast a *different* vote (for example, a voter can cast a distinct vote from their boss). Both variants of the attack are particularly useful when the bulletin board includes the hash of the ballot (for example, in the Helios 3.0 implementation), rather than the complete ballot, because the hashes will be distinct.

An adversary may replay ballots in different elections, when the trustees' public key is reused and the candidate lists for each election are of equal length. This variant of the attack can be avoided if distinct keys are used for each election.

The variants of our attack in this section have all been successfully launched against the Helios 3.0 implementation.

C. Generalised attack and French election case study

Our attack demonstrates that the ballot of an arbitrary voter can be replayed by any other voter. In general, this does not reveal the voter's vote; but, some information is leaked, and colluding voters can replay sufficiently many ballots to leak the voter's vote. We will now discuss the feasibility of compromising ballot secrecy in a real-world election, focusing on the cost of an attack in French legislative elections, where each district elects a representative for the French National Assembly. Districts have several polling stations and each polling station individually announces its tally [52]; these tallies are published in local newspapers. The publication of tallies is typical of French elections at all levels; for example, from the election of mayor, to the presidential election.

In this (standard) voting configuration, an adversary can violate the ballot secrecy of a given voter by corrupting voters registered at the same polling station (for example, a coalition of neighbours or a family). The corrupted voters replay the ballot of the voter under attack, as previously explained. The motivation for restricting the selection of corrupted voters to the same polling station is twofold. Firstly, fewer corrupt voters are required to significantly influence the tally of an individual polling station (in comparison to influencing the election outcome). Secondly, it is unlikely to change the district's elected representative, because a candidate will receive only a few additional votes in the district; it follows that coercing voters to sacrifice their vote, for the purposes of the attack, should be easier. In the remainder of this section, we discuss how many corrupt voters are required to violate ballot secrecy – by making a significant change in the tally of a polling station – in an arbitrary district of Aulnay-sous-Bois and a rural district in Toul.

Party	Tally
PS	4120
UMP	3463
FN	1933
Europe Eco.	1921
Front de gauche	880
NPA	697
MODEM	456
Debout la République	431
Alliance école	193
LO	156
Émergence	113
Liste chrétienne	113

Table I
2010 LEGISLATIVE ELECTION RESULTS IN AULNAY-SOUS-BOIS [53]

1) Ballot secrecy in Aulnay-sous-Bois: Using historic data and/or polls, it is possible to construct the expected distribution of votes. For simplicity, let us assume the distribution of votes per polling station is the average of the 2010 tally (Table I), and that if the adversary can increase the number of votes for a particular candidate by more than σ (by replaying a voter's ballot), then this is sufficient to determine that the voter voted for that candidate. In addition, suppose that the adversary corrupts abstaining voters and therefore we do not consider the redistribution of votes. We remark that corrupting abstaining voters may be a fruitful strategy, since abstaining voters do not sacrifice their vote by participating in an attack.

Table II presents the expected distribution of votes, and includes the number of voters that an adversary must corrupt to determine if a voter voted for a particular candidate, for various values of σ. We shall further assume that participation in the region is consistent with 2010; that is, 291 of the 832 eligible voters are expected to participate. It follows that 50 voters corresponds to approximately 6% of the Aulnay-sous-Bois electorate, and 10 voters corresponds to approximately 1%. Our results therefore demonstrate that the privacy of a voter can be compromised by corrupting a small number of voters. In particular, for medium-size parties (in terms of votes received) – including, for example, FN and Europe Ecologie – it is sufficient to corrupt 19 voters to see the number of votes increase by 50%. Furthermore, given the low turn-out (541 voters are expected to abstain), it seems feasible to corrupt abstaining voters, and therefore an attack can be launched without any voter sacrificing their vote.

Limitations: For such an attack based on a statistical model, we acknowledge that this model is rather naïve, but believe it is sufficiently indicative to illustrate the real threat of an attack against privacy. A definitive mathematical analysis should be considered in the future.

Party	Expected tally	$\sigma = 200\%$	$\sigma = 150\%$	$\sigma = 50\%$	$\sigma = 20\%$
PS	81	162	122	41	17
UMP	68	136	102	34	14
FN	38	76	57	19	8
Europe Eco.	38	76	57	19	8
Front de gauche	17	34	26	9	4
NPA	14	28	21	7	3
MODEM	9	18	14	5	2
Debout la République	8	16	12	4	2
Alliance école	4	8	6	2	1
LO	3	6	5	2	1
Émergence	2	4	3	1	1
Liste chrétienne	2	4	3	1	1

Table II
NUMBER OF DUPLICATE BALLOTS FOR A SIGNIFICANT CHANGE IN THE TALLY

Cases of complete privacy breach: The probabilistic nature of these attacks may introduce sufficient uncertainty to prevent privacy violations, and we will consider voting configurations where an adversary can definitively learn a voter's vote. Observe that if an attacker can corrupt half of the voters at a polling station, then the vote of an arbitrary voter can be revealed. Moreover, the cost of this attack can be reduced. In particular, if n dishonest voter's replay voter $\mathcal{V}$'s ballot, then it is possible to deduce that $\mathcal{V}$ did not vote for any candidate that received strictly less than $n+1$ votes. This leaks information about voter $\mathcal{V}$'s chosen candidate and in cases where exactly one candidate received more than n votes, the voter's vote can be deduced.

2) Ballot secrecy in small polling stations: The difficulties of large scale corruption may prohibit our attack in the majority of polling stations; however, our attack is feasible in small polling stations found in rural districts. For example, let us consider the 2007 legislative elections in the district of Toul [54]. This district has 75350 eligible voters registered at 193 polling stations. Accordingly, the average polling station has 390 registered voters, but the variance is large. Indeed, 33 polling stations have between 50 and 99 voters, 9 polling stations have less then 50 voters, and the smallest two polling stations have 8, respectively 16, voters. Moreover, the attack is simplified by non-participating voters. In these small polling stations it is thus sufficient to corrupt a very small number of voters to reveal a voter's vote while the final outcome of the election would not change as it is based on 75350 eligible voters.

IV. SOLUTION: WEEDING REPLAYED BALLOTS

Our attack exploits the possibility of replaying a voter's ballot without detection, and can be attributed to the lack of ballot independence in Helios. This section sketches some possible solutions to ensure ballot independence.

A. Weeding replayed ballots

The ballots replayed in our attacks can all be identified. First, ciphertexts and signatures of knowledge should have a unique representation as group elements, for example, by requiring that the response component of signatures of knowledge is in the interval $[0, q-1]$. Second, a ballot should not contain a ciphertext that already exists on the bulletin board. The election officer should reject ballots that do not satisfy these conditions. This solution is simple and can easily be implemented in a future version of Helios.

B. Binding ballots to voters

The previous approach requires a special mechanism to handle replayed ballots. We now propose a technique that makes such actions futile. In essence, based upon inspiration from [9, §4.2] and [42], we ensure that proofs associated with replayed ballots are considered invalid; that is, we bind the link between a voter and her ballot. This is achieved by adding the identity of the voter in the construction of challenges used by signatures of knowledge. More precisely, for voter id, the sign algorithm (defined in Appendix A) is modified as follows: on input (a, b), such that $a \equiv g^r \bmod p$ and $b \equiv h^r \cdot g^m \bmod p$, let challenge $c_m = \mathcal{H}(a_{\mathsf{min}}, b_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}, id) - \sum_{i \in \{\mathsf{min},\ldots,m-1,m+1,\ldots,\mathsf{max}\}} c_i \pmod q$, where values a_{min}, $b_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}$ and $c_1, \ldots, c_{m-1}, c_{m+1}, \ldots, c_m$ are defined as before. For correctness, the verification algorithm must also be modified. In particular, for candidate signatures constructed by voter id, the verifier should check $\mathcal{H}(a_{\mathsf{min}}, b_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}, id) \equiv \sum_{\mathsf{min} \leq i \leq \mathsf{max}} c_i \pmod q$.

In a similar direction, the electronic voting protocol proposed by Juels, Catalano & Jakobsson [22] – which has been implemented by Clarkson, Chong & Myers [55], [56] as Civitas – requires ballots to be bound to private voter credentials. This provides *eligibility verifiability* [8]: anyone can check that each ballot published on the bulletin board was cast by a registered voter and at most one ballot is tallied per voter. It is likely that eligibility verifiability enforces ballot independence, but the provision of eligibility verifiability appears to be expensive, in particular, Juels, Catalano & Jakobsson and Clarkson, Chong & Myers assume the existence of an infrastructure for voter credentials.

C. Discussion

Our *weeding replayed ballots* solution is particular attractive because it adheres to Benaloh's notion of *ballot casting assurance* [24], [25] which asserts that the ballot encryption device (the browser script in this instance) does not know the voter's identity. (We remark that neither the original Helios scheme nor our proposed fix strictly satisfy Benaloh's notion of ballot casting assurance if a voter decides to use her own computer.) The ballot casting assurance principle is important because knowledge of the voter's identity could be used to infer the likelihood of auditing and this information can be used to influence the behaviour of the ballot encryption device; in particular, if a ballot is unlikely to be audited, then the device may act maliciously, for example, by encrypting a different vote. By comparison, the *binding ballots to voters* solution would necessarily require that the voter's identity is revealed to the ballot encryption device. Moreover, for privacy purposes, the election officer may chose to allocate voters with pseudo-identities when casting ballots (rather than associate ballots with actual voter identities); since these pseudo-identities are unknown to voters in advance, an additional interaction with the election officer would be required. (Note that the use of pseudo-identities does not prevent the attack by breaking the link between ballots and voters, because the link is known by the election officer.) Finally, extending Helios to provide eligibility verifiability would require a considerable extension to the Helios code-base and, furthermore, finding a suitable solution is an open problem. Accordingly, we adopt the weeding replayed ballot solution and, in the next section, we show that this is sufficient to ensure ballot secrecy, in the formal setting.

V. Formal proof of ballot secrecy

We formally prove that weeding duplicate ballots ensures ballot secrecy. We make use of the applied pi calculus [57], [58], due to its proven suitability for evaluating security properties of electronic voting protocols (see, for example, [19], [6], [8]).

A. Applied pi calculus

We first recall the applied pi calculus setting [57]. We assume an infinite set of *names* $a, b, c, \ldots, k, \ldots, m, n, \ldots, s, \ldots$, an infinite set of *variables* $x, y, z, \ldots$, and a *signature* Σ consisting of a finite set of *function symbols*, each with an associated arity. We use metavariables u, w to range over both names and variables. *Terms* L, M, N, T, U, V are built by applying function symbols to names, variables, and other terms. We write $\{^M/_x\}$ for the *substitution* that replaces the variable x with the term M. Arbitrarily large substitutions can be written as $\{^{M_1}/_{x_1}, \ldots, ^{M_l}/_{x_l}\}$ and the letters σ and τ range over substitutions. We write $N\sigma$ for the result of applying σ to the free variables of term N. A term is *ground* when it does not contain variables.

The signature Σ is equipped with an *equational theory* E, that is, a set of equations of the form $M = N$, where the terms M, N are defined over the signature Σ. We define equality modulo the equational theory, written $=_E$, as the smallest equivalence relation on terms that contains E and is closed under application of function symbols, substitution of terms for variables and bijective renaming of names. We write $M =_E N$ when the equation $M = N$ is in the theory E, and keep the signature implicit. When E is clear from its usage, we may abbreviate $M =_E N$ as $M = N$. The negation of $M =_E N$ is denoted $M \neq_E N$ (and similarly abbreviated $M \neq N$).

Figure 2 Syntax for processes

$P, Q, R ::=$	(plain) processes
0	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu\, n.P$	name restriction
if ϕ then P else Q	conditional
$u(x).P$	message input
$\overline{u}\langle M\rangle.P$	message output
$A, B, C ::=$	extended processes
P	plain process
$A \mid B$	parallel composition
$\nu\, n.A$	name restriction
$\nu\, x.A$	variable restriction
$\{^M/_x\}$	active substitution

Processes and *extended processes* are defined in the usual way (Figure 2). We write $\nu\, \tilde{u}$ for the (possibly empty) series of pairwise-distinct binders $\nu\, u_1. \cdots .\nu\, u_l$. The active substitution $\{^M/_x\}$ can replace the variable x for the term M in every process it comes into contact with and this behaviour can be controlled by restriction, in particular, the process $\nu\, x.(\{^M/_x\} \mid P)$ corresponds exactly to let $x = M$ in P. Arbitrarily large active substitutions can be obtained by parallel composition and we occasionally abbreviate $\{^{M_1}/_{x_1}\} \mid \ldots \mid \{^{M_l}/_{x_l}\}$ as $\{^{M_1}/_{x_1}, \ldots, ^{M_l}/_{x_l}\}$ or $\{^{\tilde{M}}/_{\tilde{x}}\}$. We also use σ and τ to range over active substitutions, and write $N\sigma$ for the result of applying σ to the free variables of N. Extended processes must have at most one active substitution for each variable and there is exactly one when the variable is under restriction. The only minor change compared to [57] is that conditional branches now depend on formulae $\phi, \psi ::= M = N \mid M \neq N \mid \phi \wedge \psi$. If M and N are ground, we define $[\![M = N]\!]$ to be true if $M =_E N$ and false otherwise. The semantics of $[\![\ ]\!]$ is then extended to formulae in the standard way.

The *scope* of names and variables are delimited by binders $u(x)$ and $\nu\, u$. The set of bound names is written $\mathsf{bn}(A)$ and the set of bound variables is written $\mathsf{bv}(A)$; similarly we define the set of free names $\mathsf{fn}(A)$ and free variables $\mathsf{fv}(A)$. Occasionally, we write $\mathsf{fn}(M)$ (and $\mathsf{fv}(M)$ respectively) for the set of names (and respectively variables) which appear in term M. An extended process is *closed* when every variable x is either bound or defined by an active substitution.

We define a *context* $C[\_]$ to be an extended process with a hole. We obtain $C[A]$ as the result of filling $C[\_]$'s hole with the extended process A. An *evaluation context* is a context whose hole is not in the scope of a replication, a conditional, an input, or an output. A context $C[\_]$ closes A when $C[A]$ is closed.

A *frame*, denoted φ or ψ, is an extended process built from the null process 0 and active substitutions $\{^M/_x\}$, which are composed by parallel composition and restriction. The *domain* $\mathrm{dom}(\varphi)$ of a frame φ is the set of variables that φ exports, that is, the set of variables x for which φ contains an active substitution $\{^M/_x\}$ such that x is not under restriction. Every extended process A can be mapped to a frame $\varphi(A)$ by replacing every plain process in A with 0.

1) Operational semantics: The operational semantics are defined by three relations: *structural equivalence* ($\equiv$), *internal reduction* ($\rightarrow$), and *labelled reduction* ($\xrightarrow{\alpha}$). These relations satisfy the rules in Figure 3 and are defined such that: structural equivalence is the smallest equivalence relation on extended processes that is closed by α-conversion of both bound names and bound variables, and closed under application of evaluation contexts; internal reduction is the smallest relation on extended processes closed under structural equivalence and application of evaluation contexts; and for labelled reductions α is a *label* of the form $c(M)$, $\overline{c}\langle u\rangle$, or $\nu\, u.\overline{c}\langle u\rangle$ such that u is either a channel name or a variable of base type.

2) Equivalence: The definition of observational equivalence [57] quantifies over all contexts which makes proofs difficult, therefore we adopt labelled bisimilarity in this paper. Labelled bisimilarity relies on an equivalence relation between frames, called static equivalence.

Definition 1 (Static equivalence): Two closed frames φ and ψ are statically equivalent, denoted $\varphi \approx_s \psi$, if $\mathrm{dom}(\varphi) = \mathrm{dom}(\psi)$ and there exists a set of names $\tilde{n}$ and substitutions σ, τ such that $\varphi \equiv \nu\,\tilde{n}.\sigma$ and $\psi \equiv \nu\,\tilde{n}.\tau$ and for all terms M, N such that $\tilde{n} \cap (\mathrm{fn}(M) \cup \mathrm{fn}(N)) = \emptyset$, we have $M\sigma =_E N\sigma$ holds if and only if $M\tau =_E N\tau$ holds. Two closed extended processes A, B are statically equivalent, written $A \approx_s B$, if their frames are statically equivalent; that is, $\varphi(A) \approx_s \varphi(B)$.

The relation $\approx_s$ is called *static* equivalence because it only examines the current state of the processes, and not the processes' dynamic behaviour. The following definition of labelled bisimilarity captures the dynamic part.

Definition 2 (Labelled bisimilarity): Labelled bisimilarity ($\approx_l$) is the largest symmetric relation $\mathcal{R}$ on closed extended processes such that $A\ \mathcal{R}\ B$ implies:

1) $A \approx_s B$;
2) if $A \rightarrow A'$, then $B \rightarrow^* B'$ and $A'\ \mathcal{R}\ B'$ for some B';
3) if $A \xrightarrow{\alpha} A'$ such that $\mathrm{fv}(\alpha) \subseteq \mathrm{dom}(A)$ and $\mathrm{bn}(\alpha) \cap \mathrm{fn}(B) = \emptyset$, then $B \rightarrow^* \xrightarrow{\alpha} \rightarrow^* B'$ and $A'\ \mathcal{R}\ B'$ for some B'.

Figure 3 Semantics for processes

PAR-0	A	$\equiv$	$A \mid 0$
PAR-A	$A \mid (B \mid C)$	$\equiv$	$(A \mid B) \mid C$
PAR-C	$A \mid B$	$\equiv$	$B \mid A$
REPL	$!P$	$\equiv$	$P \mid !P$
NEW-0	$\nu\, n.0$	$\equiv$	0
NEW-C	$\nu\, u.\nu\, w.A$	$\equiv$	$\nu\, w.\nu\, u.A$
NEW-PAR	$A \mid \nu\, u.B$	$\equiv$	$\nu\, u.(A \mid B)$ where $u \notin \mathrm{fv}(A) \cup \mathrm{fn}(A)$
ALIAS	$\nu\, x.\{^M/_x\}$	$\equiv$	0
SUBST	$\{^M/_x\} \mid A$	$\equiv$	$\{^M/_x\} \mid A\{^M/_x\}$
REWRITE	$\{^M/_x\}$	$\equiv$	$\{^N/_x\}$ where $M =_E N$

COMM	$\overline{c}\langle x\rangle.P \mid c(x).Q \rightarrow P \mid Q$	
THEN	if ϕ then P else $Q \rightarrow P$	if $[\![\phi]\!] = \mathsf{true}$
ELSE	if ϕ then P else $Q \rightarrow Q$	otherwise

IN $\quad c(x).P \xrightarrow{c(M)} P\{^M/_x\}$

OUT-ATOM $\quad \overline{c}\langle u\rangle.P \xrightarrow{\overline{c}\langle u\rangle} P$

OPEN-ATOM $\quad \dfrac{A \xrightarrow{\overline{c}\langle u\rangle} A' \qquad u \neq c}{\nu\, u.A \xrightarrow{\nu\, u.\overline{c}\langle u\rangle} A'}$

SCOPE $\quad \dfrac{A \xrightarrow{\alpha} A' \qquad u \text{ does not occur in } \alpha}{\nu\, u.A \xrightarrow{\alpha} \nu\, u.A'}$

PAR $\quad \dfrac{A \xrightarrow{\alpha} A' \qquad \mathrm{bv}(\alpha) \cap \mathrm{fv}(B) = \mathrm{bn}(\alpha) \cap \mathrm{fn}(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$

STRUCT $\quad \dfrac{A \equiv B \qquad B \xrightarrow{\alpha} B' \qquad B' \equiv A'}{A \xrightarrow{\alpha} A'}$

B. Modelling Helios in applied pi

We start by constructing a suitable signature Σ to capture the cryptographic primitives used by Helios and define an equational theory E to capture the relationship between these primitives.

1) Signature: We adopt the following signature.

$$\Sigma = \{\mathsf{ok}, \mathsf{zero}, \mathsf{one}, \bot, \mathsf{fst}, \mathsf{snd}, \mathsf{pair}, *, +, \circ,\\ \mathsf{partial}, \mathsf{checkspk}, \mathsf{penc}, \mathsf{spk}, \}$$

Functions $\mathsf{ok}, \mathsf{zero}, \mathsf{one}, \bot$ are constants; $\mathsf{fst}, \mathsf{snd}$ are unary functions; $\mathsf{dec}, \mathsf{pair}, \mathsf{partial}, *, +, \circ$ are binary functions;

$\mathsf{checkspk}, \mathsf{penc}$ are ternary functions; and spk is a function of arity four. We adopt infix notation for $*, +$, and $\circ$.

The term $\mathsf{penc}(T, N, M)$ denotes the encryption of plaintext M, using random nonce N and key T. The term $U * U'$ denotes the homomorphic combination of ciphertexts U and U', the corresponding operation on plaintexts is written $M + M'$ and $N \circ N'$ on nonces. The partial decryption of ciphertext U using key L is denoted $\mathsf{partial}(L, U)$. The term $\mathsf{spk}(T, N, M, U)$ represents a signature of knowledge that proves U is a ciphertext under public key T on the plaintext M using nonce N and such that M is either the constant zero or one. We introduce tuples using pairings and, for convenience, $\mathsf{pair}(M_1, \mathsf{pair}(\ldots, \mathsf{pair}(M_n, \perp)))$ is occasionally abbreviated as $(\!|M_1, \ldots, M_n|\!)$, and $\mathsf{fst}(\mathsf{snd}^{i-1}(M))$ is denoted $\pi_i(M)$, where $i \in \mathbb{N}$. We use the equational theory E that asserts functions $+, *, \circ$ are commutative and associative, and includes the equations:

$$\mathsf{fst}(\mathsf{pair}(x, y)) = x \qquad \text{(E1)}$$

$$\mathsf{snd}(\mathsf{pair}(x, y)) = y \qquad \text{(E2)}$$

$$\mathsf{zero} + \mathsf{one} = \mathsf{one} \qquad \text{(E3)}$$

$$\mathsf{dec}(x_{\mathsf{sk}}, \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{plain}})) = x_{\mathsf{plain}} \qquad \text{(E4)}$$

$$\mathsf{dec}(\mathsf{partial}(x_{\mathsf{sk}}, \mathit{ciph}), \mathit{ciph}) = x_{\mathsf{plain}} \qquad \text{(E5)}$$
$$\text{where } \mathit{ciph} = \mathsf{penc}(\mathsf{pk}(x_{\mathsf{sk}}), x_{\mathsf{rand}}, x_{\mathsf{plain}})$$

$$\mathsf{penc}(x_{\mathsf{pk}}, y_{\mathsf{rand}}, y_{\mathsf{plain}}) * \mathsf{penc}(x_{\mathsf{pk}}, z_{\mathsf{rand}}, z_{\mathsf{plain}})$$
$$= \mathsf{penc}(x_{\mathsf{pk}}, y_{\mathsf{rand}} \circ z_{\mathsf{rand}}, y_{\mathsf{plain}} + z_{\mathsf{plain}}) \qquad \text{(E6)}$$

$$\mathsf{checkspk}(x_{\mathsf{pk}}, \mathit{ball}, \mathsf{spk}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, \mathsf{zero}, \mathit{ball})) = \mathsf{ok} \qquad \text{(E7)}$$
$$\text{where } \mathit{ball} = \mathsf{penc}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, \mathsf{zero})$$

$$\mathsf{checkspk}(x_{\mathsf{pk}}, \mathit{ball}, \mathsf{spk}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, \mathsf{one}, \mathit{ball})) = \mathsf{ok} \qquad \text{(E8)}$$
$$\text{where } \mathit{ball} = \mathsf{penc}(x_{\mathsf{pk}}, x_{\mathsf{rand}}, \mathsf{one})$$

Equation E5 allows plaintext M to be recovered from ciphertext $\mathsf{penc}(\mathsf{pk}(L), N, M)$ given partial decryption $\mathsf{partial}(L, \mathsf{penc}(\mathsf{pk}(L), N, M))$, when the partial decryption is constructed using the private key L. Equation E6 represents the homomorphic combination of ciphertexts. The Equations E7 and E8 allow the verification of signatures of knowledge $\mathsf{spk}(T, N, M, \mathsf{penc}(T, N, M))$, when $M \in \{\mathsf{zero}, \mathsf{one}\}$. The remaining equations are standard.

Example 1: Given randomness N, N', plaintexts $(M, M') \in \{(\mathsf{zero}, \mathsf{zero}), (\mathsf{zero}, \mathsf{one}), (\mathsf{one}, \mathsf{zero})\}$, and public key T, one can construct a signature of knowledge $L = \mathsf{spk}(T, N \circ N', M + M', \mathsf{penc}(T, N, M) * \mathsf{penc}(T, N', M'))$. Then $\mathsf{checkspk}$ applied to the public key T, the homomorphically combined ciphertexts $\mathsf{penc}(T, N, M) * \mathsf{penc}(T, N', M')$, and the signature L is equal to ok using Equations E3, E6, E7, and E8

2) Helios process specification: In the applied pi calculus, it is sufficient to model the parts of the voting system which need to be trusted for ballot secrecy; all the remaining parts of the system are controlled by the adversarial environment. Accordingly, we assume the existence of at least two honest voters $\mathcal{A}$, $\mathcal{B}$; since this avoids the scenario where ballot secrecy of an individual voter is compromised by collusion amongst all the remaining voters. In addition, the following trust assumptions are required.

- At least one trustee is honest
- The election officer runs the bulletin board honestly:
  - Voters $\mathcal{A}$, $\mathcal{B}$ have authentic channels with the bulletin board
  - Signatures of knowledge are checked for dishonest voters*
  - Replays of honest ballots (that is, those cast by $\mathcal{A}$ or $\mathcal{B}$) are rejected*
  - The tally is correctly computed*
  - The trustees have an authentic channel with the bulletin board
- The browser script is trusted and has the correct public key of the election

(Assumptions marked with * could be performed by an honest trustee, rather than the bulletin board.) Although neither voters nor observers can verify that there exists an honest trustee, an assurance of trust is provided by distribution. The necessity to trust the election officer to run the bulletin board is less desirable and work-in-progress [59] aims to weaken this assumption; moreover, to further distribute trust assumptions, the trustees could also check signatures and tallying. Finally, trust in the browser script can be obtained by using software written by a reputable source or writing your own code.

In an election with two candidates, the trusted components are modelled by the administration process A_n^ϕ and voting process V defined in Figure 4. For generality, the administration process A_n^ϕ is parametrised by the number of voters n and a formula ϕ; the latter corresponds to the checks performed by the bulletin board before accepting a ballot. We will consider several variants of Helios (including the original Helios 2.0 protocol and our fixed scheme) by considering suitable formula that we call *Helios process specifications*.

Definition 3 (Helios process specification): A formula ϕ is a *Helios process specification*, if $\mathrm{fv}(\phi) \subseteq \{y_1, y_2, y_{\mathsf{ballot}}, z_{\mathsf{pk}}\}$.

The voting process V contains free variables $x_{\mathsf{vote}}, x'_{\mathsf{vote}}$ to represent the voter's vote (which is expected to be encoded as constants zero and one), and the free variable x_{auth} represents the channel shared by the voter and the bulletin board. The definition of the process V corresponds to the description of the browser script (Figure 1). The administration process A_n^ϕ is parametrised by the number of voters n and Helios process specification ϕ. The restricted

Figure 4 Helios process specification

Given the number of voters $n \geq 2$ and Helios process specification ϕ, the administration process A_n^ϕ and voting process V are defined below

$$
\begin{array}{rl}
V = & \nu\, r\ . \\
& \text{let } ciph = \mathsf{penc}(z_{\mathsf{pk}}, r, x_{\mathsf{vote}}) \text{ in} \\
& \text{let } spk = \mathsf{spk}(z_{\mathsf{pk}}, r, x_{\mathsf{vote}}, ciph)) \text{ in} \\
& \nu\, r'\ . \\
& \text{let } ciph' = \mathsf{penc}(z_{\mathsf{pk}}, r', x'_{\mathsf{vote}}) \text{ in} \\
& \text{let } spk' = \mathsf{spk}(z_{\mathsf{pk}}, r', x'_{\mathsf{vote}}, ciph')) \text{ in} \\
& \text{let } \widehat{spk} = \mathsf{spk}(z_{\mathsf{pk}}, r \circ r', x_{\mathsf{vote}} + x'_{\mathsf{vote}}, ciph * ciph') \text{ in} \\
& \overline{x_{\mathsf{auth}}}\langle (\!| ciph, ciph', spk, spk', \widehat{spk} |\!) \rangle
\end{array}
$$

$$
A_n^\phi = \nu\, sk_T, a_1, a_2, d\ .\ (\_ \mid BB_n^\phi \mid T \mid \{^{\mathsf{pk}(sk_T)}/_{z_{\mathsf{pk}}}\})
$$

$$
\begin{array}{rl}
BB_n^\phi = & a_1(y_1)\ .\ \overline{c}\langle y_1\rangle\ .\ a_2(y_2)\ .\ \overline{c}\langle y_2\rangle\ . \\
& a_3(y_3)\ .\ \text{if } \phi\{^{y_3}/_{y_{\mathsf{ballot}}}\} \text{ then} \\
& \cdots\ a_n(y_n)\ .\ \text{if } \phi\{^{y_n}/_{y_{\mathsf{ballot}}}\} \text{ then} \\
& \text{let } tally = \pi_1(y_1) * \cdots * \pi_1(y_n) \text{ in} \\
& \text{let } tally' = \pi_2(y_1) * \cdots * \pi_2(y_n) \text{ in} \\
& \overline{d}\langle (\!| tally, tally' |\!) \rangle\ . \\
& d(y_{\mathsf{partial}})\ . \\
& \overline{c}\langle y_{\mathsf{partial}}\rangle\ . \\
& \overline{c}\langle (\!| \mathsf{dec}(\pi_1(y_{\mathsf{partial}}), tally), \mathsf{dec}(\pi_2(y_{\mathsf{partial}}), tally')|\!)\rangle
\end{array}
$$

$$
\begin{array}{rl}
T = & d(y_{\mathsf{tally}})\ . \\
& \overline{d}\langle (\!| \mathsf{partial}(sk_T, \pi_1(y_{\mathsf{tally}})), \mathsf{partial}(sk_T, \pi_2(y_{\mathsf{tally}}))|\!)\rangle
\end{array}
$$

name sk_T models the tallier's secret key and the public part $\mathsf{pk}(sk_T)$ is included in the process's frame. The restricted names a_1, a_2 model authentic channels between the two honest voters and the bulletin board, and the channel name d captures the authentic channel with the honest trustee. To ensure the adversary has access to messages sent on private channels, communication is relayed on the public channel c. The sub-process BB_n^ϕ represents the bulletin board and T represents the tallier. The bulletin board accepts ballots from each voter and checks they are valid using the Helios process specification ϕ (this predicate will be discussed in more detail below). Once all ballots have been submitted, the bulletin board homomorphically combines the ciphertexts and sends the encrypted tallies to the tallier for decryption. (The necessity for all voters to participate is included for simplicity, in particular, our bulletin board does not weed ballots containing invalid proofs.) The tallier receives the homomorphic combinations of ballots y_{tally} and derives a partial decryption for each candidate; these partial decryptions are sent to the bulletin board and the election result is published.

Example 2: Given a Helios process specification ϕ, an election with voters $\mathcal{A}$ and $\mathcal{B}$ who select votes (m_1, m'_1), $(m_2, m'_2) \in \{(\mathsf{zero}, \mathsf{zero}), (\mathsf{zero}, \mathsf{one}), (\mathsf{one}, \mathsf{zero})\}$ and such that the other $n - 2$ voters are controlled by the adversary, can be modelled by the process $A_n^\phi[V\{^{a_1}/_{x_{\mathsf{auth}}}\}\sigma \mid V\{^{a_2}/_{x_{\mathsf{auth}}}\}\tau]$, where $\sigma = \{^{m_1}/_{x_{\mathsf{vote}}}, ^{m'_1}/_{x'_{\mathsf{vote}}}\}$ and $\tau = \{^{m_2}/_{x_{\mathsf{vote}}}, ^{m'_2}/_{x'_{\mathsf{vote}}}\}$.

Ballot validity: In Helios 2.0, the election officer considers a ballot to be valid if the signature proofs of knowledge hold. Accordingly, we can model the Helios administration by the process $A_n^{\phi_{\mathsf{orig}}}$ where the Helios process specification ϕ_{orig} is defined as follows.

$$
\begin{aligned}
\phi_{\mathsf{orig}} \triangleq\ & \mathsf{checkspk}(z_{\mathsf{pk}}, \pi_1(y_{\mathsf{ballot}}), \pi_3(y_{\mathsf{ballot}})) = \mathsf{ok} \\
& \wedge \mathsf{checkspk}(z_{\mathsf{pk}}, \pi_2(y_{\mathsf{ballot}}), \pi_4(y_{\mathsf{ballot}})) = \mathsf{ok} \\
& \wedge \mathsf{checkspk}(z_{\mathsf{pk}}, \pi_1(y_{\mathsf{ballot}}) * \pi_2(y_{\mathsf{ballot}}), \pi_5(y_{\mathsf{ballot}})) = \mathsf{ok}
\end{aligned}
$$

We have shown that these checks are insufficient to ensure ballot secrecy. Our weeding replayed ballots solution proposed in Section IV-A additionally requires that the ciphertexts inside the ballot do not appear on the bulletin board. This revised scheme can be modelled using the Helios process specification ϕ_{sol} is defined as follows.

$$
\begin{aligned}
\phi_{\mathsf{sol}} \triangleq\ & \phi_{\mathsf{orig}} \wedge \pi_6(y_{\mathsf{ballot}}) = \perp \wedge \\
& \bigwedge_{i,j \in \{1,2\}} \pi_i(y_j) \neq \pi_1(y_{\mathsf{ballot}}) \wedge \pi_i(y_j) \neq \pi_2(y_{\mathsf{ballot}})
\end{aligned}
$$

We can also model a naïve solution that would consist in weeding only identical ballots by considering the Helios process specification ϕ_{ident} defined below.

$$
\phi_{\mathsf{ident}} \triangleq \phi_{\mathsf{orig}} \wedge \pi_6(y_{\mathsf{ballot}}) = \perp \wedge\ y_{\mathsf{ballot}} \neq y_1 \wedge y_{\mathsf{ballot}} \neq y_2
$$

We have already shown that removing exact duplicates is insufficient because it would fail to detect variants of our attack whereby the contents of a ballot are permuted. In the next section, we formally show that Helios 2.0 (modelled using ϕ_{orig}) and the naïve solution (modelled using ϕ_{ident}) do not satisfy ballot secrecy, and that our proposed solution (modelled using ϕ_{sol}) does satisfy ballot secrecy.

C. Formal analysis: Ballot secrecy

Based upon Delaune, Kremer & Ryan [4], [5], [19], and as previous discussed (see *related work* in Section I), we formalise ballot secrecy for two voters $\mathcal{A}$, $\mathcal{B}$ and two candidates t, t' with the assertion that an adversary cannot distinguish between a situation in which voter $\mathcal{A}$ votes for candidate t and voter $\mathcal{B}$ votes for candidate t', from another situation in which $\mathcal{A}$ votes t' and $\mathcal{B}$ votes t. Formally, this is captured by Definition 4.

Definition 4 (Ballot secrecy): Given a Helios process specification ϕ, we say *ballot secrecy is satisfied* if for all $(m_1, m'_1), (m_2, m'_2) \in \{(\mathsf{zero}, \mathsf{zero}), (\mathsf{zero}, \mathsf{one}), (\mathsf{one}, \mathsf{zero})\}$ and integers $n \geq 2$, we have

$$
A_n^\phi[V\{^{a_1}/_{x_{\mathsf{auth}}}\}\sigma \mid V\{^{a_2}/_{x_{\mathsf{auth}}}\}\tau] \approx_l A_n^\phi[V\{^{a_1}/_{x_{\mathsf{auth}}}\}\tau \mid V\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma]
$$

where $\sigma = \{^{m_1}/_{x_{\mathsf{vote}}}, {}^{m'_1}/_{x'_{\mathsf{vote}}}\}$ and $\tau = \{^{m_2}/_{x_{\mathsf{vote}}}, {}^{m'_2}/_{x'_{\mathsf{vote}}}\}$.

The ballot secrecy definition proposed by Delaune, Kremer & Ryan considered a vote to be an arbitrary name, whereas a vote in our setting must be a pair $(m, n) \in \{(\mathsf{zero}, \mathsf{zero}), (\mathsf{zero}, \mathsf{one}), (\mathsf{one}, \mathsf{zero})\}$; it follows that Definition 4 is a straightforward variant of the original.

The Helios 2.0 protocol does not satisfy our privacy definition (Lemma 1) and naïve ballot weeding solutions are also insufficient (Lemma 2).

Lemma 1: The Helios process specification ϕ_{orig} does not satisfy ballot secrecy.

Intuitively, the proof of Lemma 1 is due to the environment's ability to replay $\mathcal{A}$'s ballot, therefore introducing an observable difference: the result will include two instances of $\mathcal{A}$'s vote. Formally, this follows immediately from the proof Lemma 2.

Lemma 2: The Helios process specification ϕ_{ident} does not satisfy ballot secrecy.

Proof: Consider $n = 3$, $(m_1, m'_1) = (\mathsf{zero}, \mathsf{one})$ and $(m_2, m'_2) = (\mathsf{one}, \mathsf{zero})$. Let $\sigma = \{^{m_1}/_{x_{\mathsf{vote}}}, {}^{m'_1}/_{x'_{\mathsf{vote}}}\}$ and $\tau = \{^{m_2}/_{x_{\mathsf{vote}}}, {}^{m'_2}/_{x'_{\mathsf{vote}}}\}$. We consider a sequence of transitions where the two voters output their ballots and then the adversary chooses its ballots to be a permutation of the first voter's ballot. Namely, if the first voter's ballot is $(\!| ciph, ciph', spk, spk', \widehat{spk} |\!)$ then the adversary outputs $(\!| ciph', ciph, spk', spk, \widehat{spk} |\!)$. Formally, this corresponds to the transitions.

$$A_n^{\phi}[V\{^{a_1}/_{x_{\mathsf{auth}}}\}\sigma \mid V\{^{a_2}/_{x_{\mathsf{auth}}}\}\tau] \rightarrow \xrightarrow{\nu\, x.\overline{c}\langle x\rangle} \rightarrow \xrightarrow{\nu\, y.\overline{c}\langle y\rangle} \xrightarrow{c((\!|\pi_2(x),\pi_1(x),\pi_4(x),\pi_3(x),\pi_5(x)|\!))} \rightarrow^{*} \xrightarrow{\nu\, z.\overline{c}\langle z\rangle} \nu\, \tilde{n}.\tau_1$$

for some names $\tilde{n}$ and substitution τ_1 such that:

$$\mathsf{dec}(\pi_1(z), \pi_1(x) * \pi_1(y) * \pi_2(x))\tau_1 \quad =_E \quad \mathsf{one} + \mathsf{one}$$

Then this labeled transition has to matched:

$$A_n^{\phi}[V\{^{a_1}/_{x_{\mathsf{auth}}}\}\tau \mid V\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma] \rightarrow \xrightarrow{\nu\, x.\overline{c}\langle x\rangle} \rightarrow \xrightarrow{\nu\, y.\overline{c}\langle y\rangle} \xrightarrow{c((\!|\pi_2(x),\pi_1(x),\pi_4(x),\pi_3(x),\pi_5(x)|\!))} \rightarrow^{*} \xrightarrow{\nu\, z.\overline{c}\langle z\rangle} \nu\, \tilde{n}.\tau_2$$

for some names $\tilde{n}$ and substitution τ_2, such that:

$$\mathsf{dec}(\pi_1(z), \pi_1(x) * \pi_1(y) * \pi_2(x))\tau_2 \quad =_E \quad \mathsf{one}$$

It follows immediately that $\nu\, \tilde{n}.\tau_1 \not\approx_s \nu\, \tilde{n}.\tau_2$ and, hence, ϕ_{ident} does not satisfy ballot secrecy. ■

In contrast, removing duplicates up to permutation ensures ballot secrecy.

Theorem 1: The Helios process specification ϕ_{sol} satisfies ballot secrecy.

ProVerif is an automatic tool that can check equivalence in the applied pi calculus [60]. Although ProVerif has been successfully used to prove ballot secrecy (for example, in the Fujioka, Okamoto & Ohta protocol [61]), it cannot prove Theorem 1 for two main reasons. First, ProVerif cannot prove equivalences under the homomorphic equation (Equation E6). Second, our theorem states ballot secrecy for any number n of participants and ProVerif cannot handle parametrised processes. We proceed by constructing a relation that relates $A_n^{\phi}[V\{^{a_1}/_{x_{\mathsf{auth}}}\}\sigma \mid V\{^{a_2}/_{x_{\mathsf{auth}}}\}\tau]$ and $A_n^{\phi}[V\{^{a_1}/_{x_{\mathsf{auth}}}\}\tau \mid V\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma]$, and all their successors, such that it satisfies the three properties of Definition 2. In particular, the two final frames (containing the result of the election) should be statically equivalent. A key step is to show that ballots accepted by the bulletin board must have a particular form due to the checks performed by ϕ_{sol}.

Definition 5 (Valid ballot): A term N is said to be a *valid ballot* if $N = (\!| N_1, N_2, N_3, N_4, N_5 |\!)$ for some N_i such that $N_1 = \mathsf{penc}(z_{\mathsf{pk}}, N_1^1, N_1^2)$ and $N_1 = \mathsf{penc}(z_{\mathsf{pk}}, N_2^1, N_2^2)$ with $N_1^2, N_2^2 \in \{\mathsf{zero}, \mathsf{one}\}$.

Having shown that the bulletin board accepts only valid ballots, we can deduce that the outcome of the election at the end of the execution of $A_n^{\phi}[V\{^{a_1}/_{x_{\mathsf{auth}}}\}\sigma \mid V\{^{a_2}/_{x_{\mathsf{auth}}}\}\tau]$ is exactly the same as in $A_n^{\phi}[V\{^{a_1}/_{x_{\mathsf{auth}}}\}\tau \mid V\{^{a_2}/_{x_{\mathsf{auth}}}\}\sigma]$. We can then conclude the proof of Theorem 1 by showing that the partial decryptions and the encrypted ballots of honest voters do not leak any extra information to the adversary. The full proof appears in the long version of this paper [62].

D. Limitations

The limitations of our model, which we introduced to simplify the presentation and proof, are detailed below; we believe a full security proof should follow using similar reasoning. We consider a model with only two candidates and, moreover, we make use of the (standard) definition of ballot secrecy which is limited to elections with two honest voters [4], [5], [19]. In addition, the definition of ballot secrecy does not consider parallel composition of protocol executions and we therefore recommend using distinct keys for each election (although we believe it should be sufficient to include an election identifier – for example, the election fingerprint – within the challenge hashes included within signatures of knowledge, similar to the methodology in Section IV-B). The administrative process A_n^{ϕ} enforces an ordering on voters (namely, the voter using private channel a_1 must vote first, followed by the voter using private channel a_2, and then any remaining voters – controlled by the adversarial environment – can vote) and A_n^{ϕ} does not permit revoting. The signature and equational theory do not capture low-level technical details surrounding the correct construction of public keys; in particular, we do not use signatures of knowledge to verify correct key construction. We also omit signatures of knowledge that demonstrate correct construction of partial decryptions. Finally, we offer the usual caveat to formal analysis and acknowledge that our result does not imply the absence of real-world attacks (see, for example, [63], [64], [65], [66], [67]). It may, therefore, be possible to modify the ballot in a way that would not be captured by our analysis. (In particular, it is important to notice that the scheme used for signatures of knowledge is not provably non-malleable.) Accordingly, we encourage a thorough cryptographic analysis of our solution in the provable security model.

VI. Conclusion and further discussion

This paper identifies a vulnerability in Helios 2.0 which can be used to violate ballot secrecy. Critics may argue that an attack is unrealistic due its high cost; indeed, in some cases, the attack may change the outcome of an election (that is, the votes introduced for the purposes of violating privacy may swing the result), and large scale privacy invasions would be expensive in terms of the required number of dishonest voters. However, if the views of these critics are to be entertained, then we must revise the standard definitions of ballot secrecy in the literature (for example, [4], [5], [6]) because Helios cannot satisfy them. Furthermore, we believe all voters should be considered equally and, hence, the preservation of ballot secrecy should be universal. But, for elections using Helios, our case study demonstrates the contrary: in French legislative elections a coalition of voters can gain some information about a voter's vote in an arbitrary polling station and, moreover, if the number of voters registered at a particular polling station is small (for example, in a rural setting), then a voter's privacy can be violated by a few dishonest voters. It follows that privacy of individual voters can be compromised by a few dishonest voters and, accordingly, we believe our attack is significant. We also believe the absence of ballot independence can be similarly exploited in other electronic voting protocols to violate privacy (indeed, our preliminary results support this hypothesis). To address the problem, we have introduced a variant of the Helios protocol which has been shown to satisfy definitions of ballot secrecy in the applied pi calculus and work in progress aims to provide a full security proof in a cryptographic setting. Finally, Adida & Pereira have acknowledged the vulnerability [68], [69] and a fix has been proposed for future Helios releases.

Acknowledgements

We are grateful to Ben Adida and Olivier Pereira for their constructive comments, and hope this research will enhance future Helios releases. Discussion with Mark D. Ryan helped clarify the presentation of this paper, and Ben Adida informed us that Douglas Wikström is the contemporaneous discoverer of this attack. David Bernhard gave useful feedback on our attack variants. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement n° 258865, project ProSecure, and the ANR-07-SeSur-002 AVOTÉ project.

Appendix

A. *Signatures of knowledge*

Helios is reliant on signatures of knowledge to ensure secrecy and integrity of the ElGamal scheme, and to ensure voters encrypt valid votes. This appendix presents suitable cryptographic primitives. Let $\mathcal{H}$ denote a hash function. In Helios, $\mathcal{H}$ is defined to be SHA-256.

1) Knowledge of discrete logs: Given the aforementioned cryptographic parameters (p, q, g), a signature of knowledge demonstrating knowledge of a discrete logarithm $h = \log_g g^x$ can be derived, and verified, as defined by [70], [71], [72].

Sign: Given x, select a random nonce $w \in_R \mathbb{Z}_q^*$. Compute witness $g' = g^w \bmod p$, challenge $c = \mathcal{H}(g') \bmod q$ and response $s = w + c \cdot x \bmod q$.

Verify: Given h and signature g', s, check $g^s \equiv g' \cdot h^c \pmod{p}$, where $c = \mathcal{H}(g') \bmod q$.

A valid proof asserts knowledge of x such that $x = \log_g h$; that is, $h \equiv g^x \bmod p$.

2) Equality between discrete logs: Given the aforementioned cryptographic parameters (p, q, g), a signature of knowledge demonstrating equality between discrete logarithms $\log_f f^x$ and $\log_g g^x$ can be derived, and verified, as defined by [44], [45].

Sign: Given f, g, x, select a random nonce $w \in_R \mathbb{Z}_q^*$. Compute witnesses $f' = f^w \bmod p$ and $g' = g^w \bmod p$, challenge $c = \mathcal{H}(f', g') \bmod q$ and response $s = w + c \cdot x \bmod q$.

Verify: Given f, g, h, k and signature f', g', s, check $f^s \equiv f' \cdot h^c \pmod{p}$ and $g^s \equiv g' \cdot k^c \pmod{p}$, where $c = \mathcal{H}(f', g') \bmod q$.

A valid proof asserts $\log_f h = \log_g k$; that is, there exists x, such that $h \equiv f^x \bmod p$ and $k \equiv g^x \bmod p$. This signature of knowledge scheme can be extended to a disjunctive proof of equality between discrete logs (see below).

For our purposes, given a ciphertext (a, b), each trustee would derive a signature on g, a, x_i, where x_i is the trustee's private key share. The ith trustee's signature g'_i, a'_i, c_i, s_i would be verified with respect to g, a, h_i, k_i, where h_i is the trustee's share of the public key and k_i is the trustee's partial decryption; that is, the proof asserts $\log_g h_i = \log_a k_i$, as required for integrity of decryption.

3) Disjunctive proof of equality between discrete logs: Given the aforementioned cryptographic parameters (p, q, g), a signature of knowledge demonstrating that a ciphertext (a, b) contains either 0 or 1 (without revealing which), can be constructed by proving that either $\log_g a = \log_h b$ or $\log_g a = \log_h b/g^m$; that is, a signature of knowledge demonstrating a disjunct proof of equality between discrete logarithms [41], [43]. Observe for a valid ciphertext (a, b) that $a \equiv g^r \bmod p$ and $b \equiv h^r \cdot g^m \bmod p$ for some nonce $r \in \mathbb{Z}_q^*$; hence the former disjunct $\log_g g^r = \log_h h^r \cdot g^m$ is satisfied when $m = 0$, and the latter $\log_g g^r = \log_h(h^r \cdot g^m)/g^m$ when $m = 1$.

This technique is generalised by [13] to allow a signature of knowledge demonstrating that a ciphertext (a, b) contains message m, where $m \in \{\mathsf{min}, \ldots, \mathsf{max}\}$ for some system parameters $\mathsf{min}, \mathsf{max} \in \mathbb{N}$. Formally, a signature of knowledge demonstrating a disjunct proof of equality between discrete logarithms can be derived, and verified, as follows [13], [41], [43].

Sign: Given ciphertext (a, b) such that $a \equiv g^r \bmod p$ and $b \equiv h^r \cdot g^m \bmod p$ for some nonce $r \in \mathbb{Z}_q^*$, where plaintext $m \in \{\mathsf{min}, \ldots, \mathsf{max}\}$. For all $i \in \{\mathsf{min}, \ldots, m-1, m+1, \ldots, \mathsf{max}\}$, compute challenge $c_i \in_R \mathbb{Z}_q^*$, response $s_i \in_R \mathbb{Z}_q^*$ and witnesses $a_i = g^{s_i}/a^{c_i} \bmod p$ and $b_i = h^{s_i}/(b/g^i)^{c_i} \bmod p$. Select a random nonce $w \in_R \mathbb{Z}_q^*$. Compute witnesses $a_m = g^w \bmod p$ and $b_m = h^w \bmod p$, challenge $c_m = \mathcal{H}(a_{\mathsf{min}}, b_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}) - \sum_{i \in \{\mathsf{min}, \ldots, m-1, m+1, \ldots, \mathsf{max}\}} c_i \pmod q$ and response $s_m = w + r \cdot c_m \bmod q$.

Verify: Given (a, b) and $(a_{\mathsf{min}}, b_{\mathsf{min}}, c_{\mathsf{min}}, s_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}, c_{\mathsf{max}}, s_{\mathsf{max}})$, for each $\mathsf{min} \leq i \leq \mathsf{max}$ check $g^{s_i} \equiv a_i \cdot a^{c_i} \pmod p$ and $h^{s_i} \equiv b_i \cdot (b/g^i)^{c_i} \pmod p$. Finally, check $\mathcal{H}(a_{\mathsf{min}}, b_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}) \equiv \sum_{\mathsf{min} \leq i \leq \mathsf{max}} c_i \pmod q$.

A valid proof asserts that (a, b) is a ciphertext containing the message m such that $m \in \{\mathsf{min}, \ldots, \mathsf{max}\}$.

References

[1] A. Fujioka, T. Okamoto, and K. Ohta, "A Practical Secret Voting Scheme for Large Scale Elections," in *AUSCRYPT'92: Workshop on the Theory and Application of Cryptographic Techniques*, ser. LNCS, vol. 718. Springer, 1992, pp. 244–251.

[2] T. Okamoto, "Receipt-Free Electronic Voting Schemes for Large Scale Elections," in *SP'97: 5th International Workshop on Security Protocols*, ser. LNCS, vol. 1361. Springer, 1998, pp. 25–35.

[3] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-Resistant Electronic Elections," Cryptology ePrint Archive, Report 2002/165, 2002.

[4] S. Kremer and M. D. Ryan, "Analysis of an Electronic Voting Protocol in the Applied Pi Calculus," in *ESOP'05: 14th European Symposium on Programming*, ser. LNCS, vol. 3444. Springer, 2005, pp. 186–200.

[5] S. Delaune, S. Kremer, and M. Ryan, "Coercion-Resistance and Receipt-Freeness in Electronic Voting," in *CSFW'06: 19th Computer Security Foundations Workshop*. IEEE Computer Society, 2006, pp. 28–42.

[6] M. Backes, C. Hriţcu, and M. Maffei, "Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-calculus," in *CSF'08: 21st Computer Security Foundations Symposium*. IEEE Computer Society, 2008, pp. 195–209.

[7] Participants of the Dagstuhl Conference on Frontiers of E-Voting, "Dagstuhl Accord," http://www.dagstuhlaccord.org/, 2007.

[8] S. Kremer, M. D. Ryan, and B. Smyth, "Election verifiability in electronic voting protocols," in *ESORICS'10: 15th European Symposium on Research in Computer Security*, ser. LNCS, vol. 6345. Springer, 2010, pp. 389–404.

[9] R. Gennaro, "Achieving independence efficiently and securely," in *PODC'95: 14th Principles of Distributed Computing Symposium*. ACM Press, 1995, pp. 130–136.

[10] D. Chaum, P. Y. Ryan, and S. Schneider, "A Practical Voter-Verifiable Election Scheme," in *ESORICS'05: 10th European Symposium On Research In Computer Security*, ser. LNCS, vol. 3679. Springer, 2005, pp. 118–139.

[11] B. Adida, "Advances in Cryptographic Voting Systems," Ph.D. dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2006.

[12] ——, "Helios: Web-based Open-Audit Voting," in *USENIX Security'08: 17th USENIX Security Symposium*. USENIX Association, 2008, pp. 335–348.

[13] B. Adida, O. Marneffe, O. Pereira, and J. Quisquater, "Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios," in *EVT/WOTE'09: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. USENIX Association, 2009.

[14] B. Smyth, M. D. Ryan, S. Kremer, and M. Kourjieh, "Towards automatic analysis of election verifiability properties," in *ARSPA-WITS'10: Joint Workshop on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security*, ser. LNCS, vol. 6186. Springer, 2010, pp. 165–182.

[15] B. Smyth, "Formal verification of cryptographic protocols with automated reasoning," Ph.D. dissertation, School of Computer Science, University of Birmingham, 2011.

[16] J. Benaloh, S. Vaudenay, and J. Quisquater, "Final Report of IACR Electronic Voting Committee," International Association for Cryptologic Research. http://www.iacr.org/elections/eVoting/finalReportHelios_2010-09-27.html, Sept 2010.

[17] S. Haber, J. Benaloh, and S. Halevi, "The Helios e-Voting Demo for the IACR," International Association for Cryptologic Research. http://www.iacr.org/elections/eVoting/heliosDemo.pdf, May 2010.

[18] Princeton University, "Princeton election server," https://princeton-helios.appspot.com/, 2010.

[19] S. Delaune, S. Kremer, and M. D. Ryan, "Verifying privacy-type properties of electronic voting protocols," *Journal of Computer Security*, vol. 17, no. 4, pp. 435–487, Jul. 2009.

[20] ——, "Verifying Privacy-Type Properties of Electronic Voting Protocols: A Taster," in *Towards Trustworthy Elections: New Directions in Electronic Voting*, ser. LNCS, D. Chaum, M. Jakobsson, R. L. Rivest, and P. Y. Ryan, Eds. Springer, 2010, vol. 6000, pp. 289–309.

[21] B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang, and S. Yoo, "Providing Receipt-Freeness in Mixnet-Based Voting Protocols," in *ICISC'03: 6th International Conference on Information Security and Cryptology*, ser. LNCS, vol. 2971. Springer, 2004, pp. 245–258.

[22] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-Resistant Electronic Elections," in *WPES'05: 4th Workshop on Privacy in the Electronic Society*. ACM Press, 2005, pp. 61–70, see also http://www.rsa.com/rsalabs/node.asp?id=2860.

[23] ——, "Coercion-Resistant Electronic Elections," in *Towards Trustworthy Elections: New Directions in Electronic Voting*, ser. LNCS, D. Chaum, M. Jakobsson, R. L. Rivest, and P. Y. Ryan, Eds. Springer, 2010, vol. 6000, pp. 37–63.

[24] J. Benaloh, "Simple Verifiable Elections," in *EVT'06: Electronic Voting Technology Workshop*. USENIX Association, 2006.

[25] ——, "Ballot Casting Assurance via Voter-Initiated Poll Station Auditing," in *EVT'07: Electronic Voting Technology Workshop*. USENIX Association, 2007.

[26] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults," in *FOCS'85: 26th Foundations of Computer Science Symposium*. IEEE Computer Society, 1985, pp. 383–395.

[27] B. Chor and M. O. Rabin, "Achieving Independence in Logarithmic Number of Rounds," in *PODC'87: 6th Principles of Distributed Computing Symposium*. ACM Press, 1987, pp. 260–268.

[28] D. Dolev, C. Dwork, and M. Naor, "Non-Malleable Cryptography," in *STOC'91: 23rd Theory of computing Symposium*. ACM Press, 1991, pp. 542–552.

[29] ——, "Nonmalleable Cryptography," *Journal on Computing*, vol. 30, no. 2, pp. 391–437, 2000.

[30] R. Gennaro, "A Protocol to Achieve Independence in Constant Rounds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 7, pp. 636–647, 2000.

[31] K. Sako and J. Kilian, "Secure Voting Using Partially Compatible Homomorphisms," in *CRYPTO'94: 14th International Cryptology Conference*, ser. LNCS, vol. 839. Springer, 1994, pp. 411–424.

[32] D. Wikström, "Simplified Submission of Inputs to Protocols," Cryptology ePrint Archive, Report 2006/259, 2006.

[33] D. Wikström, "Simplified Submission of Inputs to Protocols," in *SCN'08: 6th International Conference on Security and Cryptography for Networks*, ser. LNCS, vol. 5229. Springer, 2008, pp. 293–308.

[34] B. Pfitzmann, "Breaking Efficient Anonymous Channel," in *EUROCRYPT'94: 11th International Conference on the Theory and Applications of Cryptographic Techniques*, ser. LNCS, vol. 950. Springer, 1994, pp. 332–340.

[35] S. Estehghari and Y. Desmedt, "Exploiting the Client Vulnerabilities in Internet E-voting Systems: Hacking Helios 2.0 as an Example," in *EVT/WOTE'10: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*. USENIX Association, 2010.

[36] L. Langer, "Privacy and Verifiability in Electronic Voting," Ph.D. dissertation, Fachbereich Informatik, Technischen Universität Darmstadt, 2010.

[37] L. Langer, A. Schmidt, J. Buchmann, and M. Volkamer, "A Taxonomy Refining the Security Requirements for Electronic Voting: Analyzing Helios as a Proof of Concept," in *ARES'10: 5th International Conference on Availability, Reliability and Security*. IEEE Computer Society, 2010, pp. 475–480.

[38] M. Volkamer and R. Grimm, "Determine the Resilience of Evaluated Internet Voting Systems," in *Re-Vote'09: First International Workshop on Requirements Engineering for E-Voting Systems*. IEEE Computer Society, 2010, pp. 47–54.

[39] L. Langer, A. Schmidt, J. Buchmann, M. Volkamer, and A. Stolfik, "Towards a Framework on the Security Requirements for Electronic Voting Protocols," in *Re-Vote'09: First International Workshop on Requirements Engineering for E-Voting Systems*. IEEE Computer Society, 2010, pp. 61–68.

[40] M. Volkamer, *Evaluation of Electronic Voting: Requirements and Evaluation Procedures to Support Responsible Election Authorities*, ser. Lecture Notes in Business Information Processing. Springer, 2009, vol. 30.

[41] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols," in *CRYPTO'94: 14th International Cryptology Conference*, ser. LNCS, vol. 839. Springer, 1994, pp. 174–187.

[42] R. Cramer, R. Gennaro, and B. Schoenmakers, "A Secure and Optimally Efficient Multi-Authority Election Scheme," in *EUROCRYPT'97: 16th International Conference on the Theory and Applications of Cryptographic Techniques*, ser. LNCS, vol. 1233. Springer, 1997, pp. 103–118.

[43] B. Schoenmakers, "Voting Schemes," in *Algorithms and Theory of Computation Handbook, Second Edition, Volume 2: Special Topics and Techniques*, M. J. Atallah and M. Blanton, Eds. CRC Press, 2009, ch. 15.

[44] T. P. Pedersen, "A Threshold Cryptosystem without a Trusted Party," in *EUROCRYPT'91: 10th International Conference on the Theory and Applications of Cryptographic Techniques*, ser. LNCS, no. 547. Springer, 1991, pp. 522–526.

[45] D. Chaum and T. P. Pedersen, "Wallet Databases with Observers," in *CRYPTO'92: 12th International Cryptology Conference*, ser. LNCS, vol. 740. Springer, 1993, pp. 89–105.

[46] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[47] D. Shanks, "Class number, a theory of factorization and genera," in *Number Theory Institute*, ser. Symposia in Pure Mathematics, vol. 20. American Mathematical Society, 1971, pp. 415–440.

[48] A. K. Lenstra and H. W. Lenstra Jr., "Algorithms in Number Theory," in *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, J. Leeuwen, Ed. MIT Press, 1990, ch. 12, pp. 673–716.

[49] C. Rudolph, "Covert Identity Information in Direct Anonymous Attestation (DAA)," in *SEC'07: 22nd International Information Security Conference*, ser. International Federation for Information Processing (IFIP), vol. 232. Springer, 2007, pp. 443–448.

[50] A. Leung, L. Chen, and C. J. Mitchell, "On a Possible Privacy Flaw in Direct Anonymous Attestation (DAA)," in *Trust'08: 1st International Conference on Trusted Computing and Trust in Information Technologies*, ser. LNCS, no. 4968. Springer, 2008, pp. 179–190.

[51] B. Smyth and V. Cortier, "Attacking ballot secrecy in Helios," YouTube video, linked from http://www.bensmyth.com/publications/10-attacking-helios/, 2010.

[52] "Article L65 of the French electoral code," http://www.legifrance.gouv.fr/.

[53] "Résultat par bureau du premier tour des élections régionales," http://www.monaulnay.com/wp-content/uploads/2010/03/resultat_regionale_par_bureau.pdf, 2010.

[54] "Est républicain," Daily French Newspaper, June, 18th 2007, meurthe-et-Moselle edition.

[55] M. R. Clarkson, S. Chong, and A. C. Myers, "Civitas: Toward a Secure Voting System," in *S&P'08: 29th Security and Privacy Symposium*. IEEE Computer Society, 2008, pp. 354–368.

[56] ——, "Civitas: Toward a Secure Voting System," http://hdl.handle.net/1813/7875, Cornell University, Tech. Rep. 2007-2081, May 2007, revised March 2008.

[57] M. Abadi and C. Fournet, "Mobile values, new names, and secure communication," in *POPL'01: 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. ACM Press, 2001, pp. 104–115.

[58] M. D. Ryan and B. Smyth, "Applied pi calculus," in *Formal Models and Techniques for Analyzing Security Protocols*, V. Cortier and S. Kremer, Eds. IOS Press, 2011, ch. 6.

[59] O. Pereira, B. Adida, and O. Marneffe, "Bringing open audit elections into practice: Real world uses of helios," Swiss e-voting workshop, https://www.e-voting-cc.ch/images/sevot10/slides/helios_20100906.pdf. See also http://www.uclouvain.be/crypto/electionmonitor/, 2010.

[60] B. Blanchet, M. Abadi, and C. Fournet, "Automated verification of selected equivalences for security protocols," *Journal of Logic and Algebraic Programming*, vol. 75, no. 1, pp. 3–51, Feb.–Mar. 2008.

[61] S. Delaune, M. D. Ryan, and B. Smyth, "Automatic verification of privacy properties in the applied pi-calculus," in *IFIPTM'08: 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security*, ser. International Federation for Information Processing (IFIP), vol. 263. Springer, 2008, pp. 263–278.

[62] B. Smyth and V. Cortier, "Attacking and fixing helios: An analysis of ballot secrecy," Cryptology ePrint Archive, Report 2010/625, 2010.

[63] P. Y. A. Ryan and S. A. Schneider, "An Attack on a Recursive Authentication Protocol. A Cautionary Tale," *Information Processing Letters*, vol. 65, no. 1, pp. 7–10, 1998.

[64] M. Abadi and P. Rogaway, "Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption)," in *IFIP TCS'00: 1st International Conference on Theoretical Computer Science*, ser. LNCS, vol. 1872. Springer, 2000, pp. 3–22.

[65] ——, "Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption)," *Journal of Cryptology*, vol. 15, no. 2, pp. 103–127, 2002.

[66] B. Warinschi, "A Computational Analysis of the Needham-Schröeder-(Lowe) Protocol," in *CSFW'03: 16th Computer Security Foundations Workshop*. IEEE Computer Society, 2003, pp. 248–262.

[67] ——, "A computational analysis of the Needham-Schroeder-(Lowe) protocol," *Journal of Computer Security*, vol. 13, no. 3, pp. 565–591, 2005.

[68] B. Adida, "Attacks and Defenses," Helios documentation, http://documentation.heliosvoting.org/attacks-and-defenses, 2010.

[69] B. Adida and O. Pereira, "Private email communication," November 2010.

[70] D. Chaum, J. Evertse, J. Graaf, and R. Peralta, "Demonstrating Possession of a Discrete Logarithm Without Revealing It," in *CRYPTO'86: 6th International Cryptology Conference*, ser. LNCS, vol. 263. Springer, 1987, pp. 200–212.

[71] D. Chaum, J. Evertse, and J. Graaf, "An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations," in *EUROCRYPT'87: 4th International Conference on the Theory and Applications of Cryptographic Techniques*, ser. LNCS, vol. 304. Springer, 1988, pp. 127–141.

[72] C.-P. Schnorr, "Efficient Identification and Signatures for Smart Cards," in *CRYPTO'89: 9th International Cryptology Conference*, ser. LNCS, vol. 435. Springer, 1990, pp. 239–252.