Does Helios ensure ballot secrecy?*

Ben Smyth and Véronique Cortier

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 voters' privacy. This vulnerability has been successfully exploited to break privacy in a small 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.

1 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. By comparison, computer systems are auditable by nature, and hence, the provision of electronic voting systems which ensure voters' privacy is an active research topic [1–3].

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

- *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:

- *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

* This work has been supported by the ANR-07-SeSur-002 AVOTÉ project.

affect a voter's decision. One aspect of fairness is *ballot independence*, which, based upon [7, §1.1], can be informally stated as follows: observing another voter's interaction with the election system does not allow a voter to cast a *related* vote, and a voter cannot affect the election outcome by *pulling out*. (We remark that the Fujioka, Okamoto & Ohta protocol [1] does not satisfy the latter condition, this may explain why their definition of fairness was restated in a variant [8] of their scheme.) The individual and universal verifiability properties (also called *end-to-end verifiability* [3, 9–12]) 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, 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 attempts to deliver end-to-end verifiability (cf. [14, 15] and [16, Chapter 3] for an analysis of end-to-end verifiability in Helios). Helios is particularly significant due to its real-world deployment: 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 [17]. In addition, the International Association of Cryptologic Research trialled the scheme in a non-binding poll [18].

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 (without knowing the vote contained within that ballot). This 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. A variant of this vulnerability abuses the malleability of ballots to ensure replayed ballots are distinct, thereby ensuring a covert attack. 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. Finally, we discuss modifications that should ensure ballot independence and prevent ballot replaying.

Related work. The concept of independence was introduced by Chor *et al.* [19] and the possibility of compromising security properties due to lack of independence has been considered, for example, by [20–23]. In the context of electronic voting, Gennaro [7] demonstrates that the application of the Fiat-Shamir heuristic in the Sako-Kilian electronic voting protocol [24] violates ballot independence, and Wikström [25, 26] 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.

Estehghari & Desmedt [27] 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, [27] represents an exploration of known vulnerabilities rather than an attack.

Structure of this paper. Section 2 recalls the cryptographic primitives used by Helios, Section 3 presents the Helios electronic voting scheme, and Section 4 analyses ballot secrecy. Finally, Section 5 emphasises the significance of this attack and presents a brief conclusion.

2 Background: Cryptography

Helios exploits the additive homomorphic [28–30] and distributed decryption [31, 32] properties of ElGamal [33]. This section summarises the necessary cryptographic primitives.

2.1 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 is $(a \cdot a' \bmod p,\ b \cdot b' \bmod p)$.

Distributed decryption. Given a ciphertext (a, b), each trustee $i \in n$ computes their share of the decryption key $k_i = a^{x_i}$. 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 [34].

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 $h_i = h$) and, for integrity of decryption, each trustee $i \in n$ must demonstrate equality between discrete logarithms $\log_g h_i$ and $\log_a k_i$. In addition, the voter must demonstrate that a valid vote has been encrypted. These proofs can be achieved using signatures of knowledge.

2.2 Signatures of knowledge

Let $\mathcal{H}$ denote a hash function. In Helios, $\mathcal{H}$ is defined to be SHA-256.

Knowledge of discrete logs. Given the aforementioned cryptographic parameters (p, q, g), a signature of knowledge demonstrating knowledge of a discrete logarithm $\log_g g^x$ can be derived, and verified, as defined by [35–37].

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, $h \equiv g^x \bmod p$.

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 [31, 32].

Sign. Given f, g, x, select a random nonce $w \in_R \mathbb{Z}_q^*$. Compute witnesses $f' = f^w \bmod p$, $g' = g^w \bmod p$, 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 share of the decryption key; that is, the proof asserts $\log_g h_i = \log_a k_i$, as required for integrity of decryption.

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 [28, 30]. 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 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, 28, 30].

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 $a_i = g^{s_i}/a^{c_i} \bmod p$, $b_i = h^{s_i}/(b/g^i)^{c_i} \bmod p$. Select a random nonce $w \in_R \mathbb{Z}_q^*$. Compute witnesses $a_m = g^w \bmod p$, $b_m = h^w \bmod p$, 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 $m \in \{\mathsf{min}, \ldots, \mathsf{max}\}$.

Software implementation. At the time of writing, the Helios cryptographic algorithms can be found in `http://github.com/benadida/helios-server/blob/master/helios/crypto/algs.py`. In particular, the function to compute signatures of knowledge demonstrating equality between discrete logs is defined by `EGZKProof.generate`, and the function for deriving signatures of knowledge demonstrating a disjunct proof of equality between discrete logarithms is defined by `EGCiphertext.generate_disjunctive_encryption_proof`.

3 Helios 2.0

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 $\tilde{t} = (t_1, \ldots, t_l)$, 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 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.)
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, 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, but the ballot is now invalid. (Invalidating audited ballots provides some practical resistance against vote selling.) See [38] for further details on ballot auditing.
4. When the voter has decided to cast her ballot, she 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 their share of the decryption key, for the encrypted tally, together with a signature of knowledge proving that their key share is well-formed. Anyone can verify these proofs.
8. The election officer decrypts the tally and publishes the result. Anyone can check this decryption.

Formally, Step 1 is defined in Figure 1 (signatures of knowledge demonstrating a disjunct proof of equality between discrete logarithms implicitly assume parameters $\mathsf{min} = 0$ and $\mathsf{max} = 1$). Checking voter eligibility (Step 4) is beyond the scope of Helios and [13] proposes the use of existing infrastructure. The remaining steps follow immediately from the application of cryptographic primitives (see Section 2 for details).

Generalisation to approval voting. For simplicity, the ballot construction algorithm (Figure 1) considers a vote $v \in \tilde{t}$. This can be generalised to a vote $\tilde{v} \subseteq \tilde{t}$,

Fig. 1 Ballot construction by the browser script

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

with minor modifications to the ballot construction algorithm; in particular, Step 5 of the algorithm would use a signature of knowledge $(\bar{a}_1, \bar{b}_1, \bar{c}_1, \bar{s}_1, \ldots, \bar{a}_{\mathsf{cmax}}, \bar{b}_{\mathsf{cmax}}, \bar{c}_{\mathsf{cmax}}, \bar{s}_{\mathsf{cmax}})$, rather than $(\bar{a}, \bar{b}, \bar{c}, \bar{s}, \bar{a}', \bar{b}', \bar{c}', \bar{s}')$, where cmax is a system parameter determining the maximum number of candidates a voter is allowed to select.

4 Analysis: 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, formal definitions of ballot secrecy [4–6] are also violated.

Intuitively, an adversary may identify a voter's ballot on the bulletin board 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 exploits this knowledge to violate the voter's privacy. A formal description of the attack will now be presented in the case of three eligible voters and Section 4.2 considers a more realistic setting.

4.1 Attacking ballot secrecy

Let us consider an election with three eligible voters: id_1, id_2 and id_3, where id_1, id_2 are honest voters and id_3 is a dishonest voter controlled by the adversary.

Further assume that the honest voters have cast their ballots and the bulletin board entries are as follows:

$$id_1, ciph_1, spk_1, spk'_1$$
$$id_2, ciph_2, spk_2, spk'_2$$

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

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

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

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

$$id_1, ciph_1, spk_1, spk'_1$$
$$id_2, ciph_2, spk_2, spk'_2$$
$$id_3, ciph_k, spk_k, spk'_k$$

It is trivial to see that each bulletin board represents a permitted vote; that is, $spk_1, spk'_1, spk_2, spk'_2, spk_k, spk'_k$ are all valid signatures of knowledge. We have shown that Helios does not satisfy ballot independence, 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 necessary decryption keys, 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.

Malleable ballots. In the attack description, the ballots cast by two voters are identical. For a covert attack, the adversary may prefer to cast a distinct ballot. This can be achieved by exploiting the malleability of signatures of knowledge; for example, given a valid signature $(a_{\mathsf{min}}, b_{\mathsf{min}}, c_{\mathsf{min}}, s_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}, c_{\mathsf{max}}, s_{\mathsf{max}})$, the signature $(a_{\mathsf{min}}, b_{\mathsf{min}}, c_{\mathsf{min}}, s_{\mathsf{min}} + q, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}, c_{\mathsf{max}}, s_{\mathsf{max}} + q)$ is also valid, where min, max are system parameters. This technique is particularly useful when the bulletin board includes the hash of the ballot, rather than the complete ballot, because the hashes will be distinct.

A practical attack against Helios 3.0. 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 for use. A video demonstrating the attack against this implementation has been produced [39].

Malleable data formats. In practice, electronic voting protocols are reliant on data structures, rather than bitstrings. For example, Helios 3.0 uses the JavaScript Object Notation format. Malleability in data formats may be similarly exploited to derive a distinct ballot when launching a covert attack. Indeed, in Helios 3.0, it is possible to add whitespace to a ballot. Once again, this is particularly useful when the bulletin board displays hashes of ballots, rather than the complete ballot, since the hashes are indistinguishable whereas the ballots may be linked.

Generalised attack against ballot secrecy. 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. However, some information is leaked, and colluding voters can replay sufficiently many ballots to leak the voter's vote. Intuitively, it should follow that Helios cannot satisfy ballot secrecy in formal settings [4–6]. 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 one in which $\mathcal{A}$ votes t' and $\mathcal{B}$ votes t. This can be expressed by the equivalence

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

Moreover, these definitions can be extended to consider privacy with arbitrary many voters. As conjectured, Helios can be shown not to satisfy these definitions. We deduce either: these definitions are too strong, or there is indeed a weakness in the Helios protocol. In the next section, we discuss the feasibility of this attack in a real-world election.

4.2 Case study: French legislative election

We now examine the possibility of compromising ballot secrecy in real-world elections. As an illustrative example, we will focus 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 [40]; 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 explained in the previous section. 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. This case study considers an arbitrary district in Aulnay-sous-Bois and more rural district in Toul.

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 1), 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.

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 1. Results of the 2010 legislative election in Aulnay-sous-Bois (Source [41])

Table 2 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. For example, for medium-size parties (in terms of votes received), including 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 is feasible to corrupt abstaining voters, and therefore an attack can be launched without any voter sacrificing their vote.

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 2. Number of duplicate ballots for a significant change in the tally

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.

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 V's chosen candidate and in cases where exactly one candidate received more than n votes, the voter's vote can be deduced. This is sufficient to violate privacy in real world elections.

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 [42]. 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.

4.3 Towards ballot independence

The 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 research directions to ensure ballot independence, and future work should formally consider the suitability of these approaches.

Weeding replayed ballots. The ballots replayed in this attack can be identified by checking for duplicates, and these ballots can be rejected. Formally, this can be achieved by invalidating ballots where: 1) there already exists an identical ballot on the bulletin board; or 2) a ballot contains values outside the expected range (for example, the response component of signatures of knowledge should be in the interval $[0, q-1]$).

While such mechanisms indeed prevent our attack, it might still possible to modify the ballot in a different way that would not be captured by this countermeasure. Indeed, it is important to notice that the scheme used for signatures of knowledge is not provably non-malleable.

Binding ballots to voters. The previous approach requires a special mechanism to handle duplicate ballots. We now introduce a technique that makes such actions futile; in essence, we ensure that proofs associated with replayed ballots are considered invalid.

Based upon inspiration from [7, §4.2], we bind the link between a voter and her ballot. This is achieved by refining the construction of commitments used by signatures of knowledge. More precisely, for voter id, the sign algorithm 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 $c_m = \mathcal{H}(a_{\mathsf{min}}, b_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}, id) - \sum_{i \in \{\mathsf{min},\ldots,m-1,m+1,\ldots,\mathsf{max}\}} c_i \pmod q$, where values $a_{\mathsf{min}}, b_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}$ 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 $\mathcal{H}(a_{\mathsf{min}}, b_{\mathsf{min}}, \ldots, a_{\mathsf{max}}, b_{\mathsf{max}}, id) \equiv \sum_{\mathsf{min} \le i \le \mathsf{max}} c_i \pmod q$.

Alternative non-interactive proofs. Helios uses a variant of the Fiat-Shamir heuristic [43] which allows malleability of the response component of signatures of knowledge. By comparison, constructing a non-interactive proof without using the Fiat-Shamir heuristic may be suitable to provide signatures of knowledge that ensure ballot independence [7, §4.2].

5 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 this attack is not realistic 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 to the voter's candidate). Moreover, large scale privacy invasions would be expensive in terms of the required number of dishonest voters. If the views of these critics are to be entertained, then we must revise our definition of ballot secrecy; in particular, the definitions of [4–6]. However, our study of the French legislative elections shows that a coalition of voters can gain some information about a voter's vote in an arbitrary polling station. In addition, 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. Furthermore, we consider all voters to be equal and therefore believe that the preservation of voter privacy should be universal; that is, all voters have the right to ballot secrecy. Since our case study demonstrates the contrary – namely, privacy of individual voters can be compromised by a few dishonest voters – we believe our attack is significant. We also believe the vulnerability highlights a new class of attacks against electronic voting protocols (indeed, our preliminary results support this hypothesis) and our future work will examine ballot secrecy in other schemes.

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.

References

1. Fujioka, A., Okamoto, T., Ohta, K.: A Practical Secret Voting Scheme for Large Scale Elections. In: AUSCRYPT'92: Workshop on the Theory and Application of Cryptographic Techniques. Volume 718 of LNCS., Springer (1992) 244–251
2. Okamoto, T.: Receipt-Free Electronic Voting Schemes for Large Scale Elections. In: SP'97: 5th International Workshop on Security Protocols. Volume 1361 of LNCS., Springer (1998) 25–35

3. Juels, A., Catalano, D., Jakobsson, M.: Coercion-Resistant Electronic Elections. Cryptology ePrint Archive, Report 2002/165 (2002)
4. Kremer, S., Ryan, M.D.: Analysis of an Electronic Voting Protocol in the Applied Pi Calculus. In: ESOP'05: 14th European Symposium on Programming. Volume 3444 of LNCS., Springer (2005) 186–200
5. Delaune, S., Kremer, S., Ryan, M.D.: Verifying privacy-type properties of electronic voting protocols. Journal of Computer Security **17**(4) (July 2009) 435–487
6. Backes, M., Hriţcu, C., Maffei, M.: Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-calculus. In: CSF'08: 21st Computer Security Foundations Symposium, IEEE Computer Society (2008) 195–209
7. Gennaro, R.: Achieving independence efficiently and securely. In: PODC'95: 14th Principles of Distributed Computing Symposium, ACM Press (1995) 130–136
8. Ohkubo, M., Miura, F., Abe, M., Fujioka, A., Okamoto, T.: An Improvement on a Practical Secret Voting Scheme. In: ISW'99: 2nd Information Security Workshop. Volume 1729 of LNCS., Springer (1999) 225–234
9. Chaum, D., Ryan, P.Y., Schneider, S.: A Practical Voter-Verifiable Election Scheme. In: ESORICS'05: 10th European Symposium On Research In Computer Security. Volume 3679 of LNCS., Springer (2005) 118–139
10. Adida, B.: Advances in Cryptographic Voting Systems. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (2006)
11. Participants of the Dagstuhl Conference on Frontiers of E-Voting: Dagstuhl Accord. `http://www.dagstuhlaccord.org/` (2007)
12. Adida, B.: Helios: Web-based Open-Audit Voting. In: USENIX Security'08: 17th USENIX Security Symposium, USENIX Association (2008) 335–348
13. Adida, B., Marneffe, O., Pereira, O., Quisquater, J.: 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. Kremer, S., Ryan, M.D., Smyth, B.: Election verifiability in electronic voting protocols. In: ESORICS'10: 15th European Symposium on Research in Computer Security. Volume 6345 of LNCS., Springer (2010) 389–404
15. Smyth, B., Ryan, M.D., Kremer, S., Kourjieh, M.: 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. Volume 6186 of LNCS., Springer (2010) 165–182 To appear.
16. Smyth, B.: Formal verification of cryptographic protocols with automated reasoning. PhD thesis, School of Computer Science, University of Birmingham (2010)
17. University, P.: Princeton election server. `https://princeton-helios.appspot.com/` (2010)
18. Haber, S., Benaloh, J., Halevi, S.: The Helios e-Voting Demo for the IACR. International Association for Cryptologic Research. `http://www.iacr.org/elections/eVoting/heliosDemo.pdf` (May 2010)
19. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. In: FOCS'85: 26th Foundations of Computer Science Symposium, IEEE Computer Society (1985) 383–395
20. Chor, B., Rabin, M.O.: Achieving Independence in Logarithmic Number of Rounds. In: PODC'87: 6th Principles of Distributed Computing Symposium, ACM Press (1987) 260–268
21. Dolev, D., Dwork, C., Naor, M.: Non-Malleable Cryptography. In: STOC'91: 23rd Theory of computing Symposium, ACM Press (1991) 542–552

22. Dolev, D., Dwork, C., Naor, M.: Nonmalleable Cryptography. Journal on Computing **30**(2) (2000) 391–437
23. Gennaro, R.: A Protocol to Achieve Independence in Constant Rounds. IEEE Transactions on Parallel and Distributed Systems **11**(7) (2000) 636–647
24. Sako, K., Kilian, J.: Secure Voting Using Partially Compatible Homomorphisms. In: CRYPTO'94: 14th International Cryptology Conference. Volume 839 of LNCS., Springer (1994) 411–424
25. Wikström, D.: Simplified Submission of Inputs to Protocols. Cryptology ePrint Archive, Report 2006/259 (2006)
26. Wikström, D.: Simplified Submission of Inputs to Protocols. In: SCN'08: 6th International Conference on Security and Cryptography for Networks. Volume 5229 of LNCS., Springer (2008) 293–308
27. Estehghari, S., Desmedt, Y.: 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)
28. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In: CRYPTO'94: 14th International Cryptology Conference. Volume 839 of LNCS., Springer (1994) 174–187
29. Cramer, R., Gennaro, R., Schoenmakers, B.: A Secure and Optimally Efficient Multi-Authority Election Scheme. In: EUROCRYPT'97: 16th International Conference on the Theory and Applications of Cryptographic Techniques. Volume 1233 of LNCS., Springer (1997) 103–118
30. Schoenmakers, B.: Voting Schemes. In Atallah, M.J., Blanton, M., eds.: Algorithms and Theory of Computation Handbook, Second Edition, Volume 2: Special Topics and Techniques. CRC Press (2009)
31. Pedersen, T.P.: A Threshold Cryptosystem without a Trusted Party. In: EUROCRYPT'91: 10th International Conference on the Theory and Applications of Cryptographic Techniques. Number 547 in LNCS, Springer (1991) 522–526
32. Chaum, D., Pedersen, T.P.: Wallet Databases with Observers. In: CRYPTO'92: 12th International Cryptology Conference. Volume 740 of LNCS., Springer (1993) 89–105
33. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**(4) (1985) 469–472
34. Shanks, D.: Class number, a theory of factorization and genera. In: Number Theory Institute. Volume 20 of Symposia in Pure Mathematics., American Mathematical Society (1971) 415–440
35. Chaum, D., Evertse, J., Graaf, J., Peralta, R.: Demonstrating Possession of a Discrete Logarithm Without Revealing It. In: CRYPTO'86: 6th International Cryptology Conference. Volume 263 of LNCS., Springer (1987) 200–212
36. Chaum, D., Evertse, J., Graaf, J.: 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. Volume 304 of LNCS., Springer (1988) 127–141
37. Schnorr, C.P.: Efficient Identification and Signatures for Smart Cards. In: CRYPTO'89: 9th International Cryptology Conference. Volume 435 of LNCS., Springer (1990) 239–252
38. Benaloh, J.: Ballot Casting Assurance via Voter-Initiated Poll Station Auditing. In: EVT'07: Electronic Voting Technology Workshop, USENIX Association (2007)
39. Smyth, B., Cortier, V.: Attacking ballot secrecy in Helios. YouTube video, linked from `http://www.bensmyth.com/publications/10-attacking-helios/` (2010)

40. : Article L65 of the French electoral code. http://www.legifrance.gouv.fr/
41. : Résultat par bureau du premier tour des élections régionales. http://www.monaulnay.com/wp-content/uploads/2010/03/resultat_regionale_par_bureau.pdf (2010)
42. : Est républicain. Daily French Newspaper (June, 18th 2007) Meurthe-et-Moselle edition.
43. Fiat, A., Shamir, A.: How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In: CRYPTO'86: 6th International Cryptology Conference. Volume 263 of LNCS., Springer (1987) 186–194