Extending Helios Towards Private Eligibility Verifiability

Oksana Kulyk[1], Vanessa Teague[2], and Melanie Volkamer[1,3]

[1] Technische Universität Darmstadt/CASED, Darmstadt, Germany
name.surname@secuso.org
[2] University of Melbourne, Melbourne, Australia
vjteague@unimelb.edu.au
[3] Karlstad University, Karlstad, Sweden

Abstract. We show how to extend the Helios voting system to provide eligibility verifiability without revealing who voted which we call private eligibility verifiability. The main idea is that real votes are hidden in a crowd of null votes that are cast by others but are indistinguishable from those of the eligible voter. This extended Helios scheme also improves Helios towards receipt-freeness.

1 Introduction

Electronic voting protocols must allow the public to verify the accuracy of the election outcome. Much of the research focus has been devoted to *universal end-to-end verifiability*, which includes *cast-as-intended*, *stored-as-cast* and *tallied-as-stored* verifiability. In particular, the Helios voting system [1] was designed to ensure these requirements. Another crucial property is *eligibility verifiability*, meaning that anyone can verify that only the votes of eligible voters have been accepted and included into the tally, thus preventing ballot stuffing by the voting system. The original version of Helios requires trust in the voting system for accepting votes from eligible voters only, thus lacking the option to verify eligibility by the general public.

In Helios and related systems, votes are cast encrypted and are anonymized before decryption and tallying. One simple way to ensure eligibility verifiability would be to make use of an existing public-key infrastructure (PKI), letting voters sign their encrypted vote and publishing all signed encrypted votes on the Bulletin Board. In that case, everyone could verify that each vote had been signed by an eligible voter. However, this approach inevitably reveals who voted in the election and who abstained.

It is worth noting that different democracies take slightly different attitudes to vote privacy in practice. In Australia and many other countries, voting is compulsory and hence a matter of (somewhat) public record. However, in other countries including Germany and Switzerland, the fact of whether or not a person has voted is regarded as private; in the United States political organisations quite openly target likely voters on an individual basis. Hence it is sometimes important to hide who voted—we call this *participation privacy*.

This work shows how to extend Helios to ensure both universal eligibility verifiability and participation privacy for voters, *i.e.* private eligibility verifiability. The proposal could also be used to improve other schemes like the Estonian voting scheme [23], or could be implemented as an independent system.

The main idea is to pad out the Helios votes with null votes cast by others. Anyone may add null votes to any voter's row; and voters can update their votes. We have decided to identify specific participants called "posting proxies" who do most of the padding with null votes (according to some, presumably randomised, algorithm).

These null votes need to have some important properties: They should not have any effect on the election outcome. The null votes must also be indistinguishable from the voter's contributions, and numerous and unpredictable enough to provide proper cover for voters. We achieve it by introducing witness-indistinguishable disjunctive proofs that ensure that each cast vote is either a null vote (represented by an encryption of 1), or a vote cast by an eligible voter. We rely on an anonymous channel that is used by posting proxies as well as by voters, in order to hide the origin of the cast votes. Our scheme uses an existing PKI rather than relying on a dedicated credential-based mechanism like JCJ/Civitas or caveat coercitor [18]. All voters must use their signing key to vote. The list of assumptions for the overall scheme as well as for the individual security requirements are listed and discussed in Section 5. Our construction also prevent voters from proving which valid vote they have cast. Thus, it improves Helios even fruther as it offers some level of receipt-freeness as protection against vote selling. The remaining security properties, such as preserving the integrity and secrecy of the vote, should hold for honest voters under the same assumptions as in the original Helios system. Note, we use the same cast-as-intended verification methods as Helios.

We would not claim that our protocol satisfies all the requirements necessary for government elections. It remains susceptible to coercion for abstaining and randomizing the vote. Also the strong assumptions about the public key infrastructure, and the smart card's good behaviour mean that it is not truly end-to-end verifiable. The necessity of participation of other contributors in each person's row may also be too strong for sufficient privacy for some elections.

The paper is structured as follows. In Section 2 we describe the related work relevant for our proposal. In Section 3 we give the background information with the building blocks used in our scheme. In Section 4 we describe the scheme itself, followed by its security analysis, including the ensured security requirements and the assumptions needed for this, in Section 5. Section 6 gives an efficiency analysis of the scheme, and Section 7 provides the conclusion.

2 Related work

Haenni and Spycher [22] provide a scheme with eligibility verifiability and participation privacy (which they call anonymity). The scheme, however, does not provide receipt-freeness. Eligibility verifiability for Helios was considered in [40].

This work takes a complementary approach to ours, because it assumes that there is no public key infrastructure and relies instead on tokens generated specifically for the election. The protocol provides eligibility verifiability and prevention of ballot stuffing under reasonable assumptions, but does not attempt either to hide who voted or to provide receipt freeness.

The goal of ensuring both eligibility verifiability and the participation privacy of the voters is addressed, among other security requirements, by the schemes aiming to provide the property of coercion resistance. The issue of coercion resistance in remote voting was addressed in the work of Juels, Catalano and Jacobson (JCJ) in [25], presenting a scheme that provides coercion resistance – the definition of which includes receipt freeness as well as protection against forced abstention, randomization and simulation attacks – against strong attacker. This scheme, however, is unsuited for practical use, due to the fact, that its performance is $\mathcal{O}(N^2)$ with N as the number of eligible voters. Therefore, a number of works have presented the improvements to the JCJ system, that preserve the coercion-resistance properties while achieving linear complexity – among others, approaches based upon group signatures [2], panic passwords [10], concurrent ballot authorization [15], anonymity sets [34] or using the voter roll [38]. Furthermore, several improvements focused on improving other shortcomings in JCJ scheme, such as addressing the issue of board flooding [26], or improving usability with using tamper-resistant smartcards [29]. These improvements, however, still require complex forms of credential management, thus lacking in usability from the voter's perspective. A number of other schemes has been suggested that provide some level of coercion resistance [27, 32], which, however, also require complex actions from the voter. The Caveat Coercitor scheme [18] aims at detecting whether coercion took place during the election, but not at preventing it.

3 Background

In this section we provide the background information we base our scheme upon.

3.1 Helios

The Helios voting system incorporates a simple yet powerful collection of methods for end-to-end verifiable voting. Each person's encrypted vote is tabulated, along with some authentication information, on a public bulletin board. Well-behaved voting clients are supposed to delete the randomness they use when generating the ciphertext—if they do so, the person cannot subsequently prove how they voted. If they fail to do so, that randomness can be used to open the ciphertext and prove its contents to a coercer. It is obvious from the bulletin board which voters have participated and which have abstained.

The client is trusted for privacy. If more than a threshold of talliers collude they too can violate privacy. No entities are trusted for integrity though of course verification procedures for the voting process and the bulletin board must be followed. Helios uses the "Benaloh challenge" [5] to allow voters to verify that their vote is cast as they intended.

3.2 Cryptographic Building Blocks

We describe the cryptographic primitives and protocols that underlie our scheme.

ElGamal encryption: Let $(g, h) \in \mathbb{G}_q^2$ be a public ElGamal key, where $\mathbb{G}_q \subset \mathbb{Z}_p$ is a multiplicative group of order q, with both p, q large primes, $p = 2q + 1$. An ElGamal encryption of $v \in \mathbb{G}_q$ using the public key (g, h) is defined as a tuple $Enc_{(g,h)}(v) = (a, b) := (g^r, v \cdot h^r)$ for some randomness $r \in \mathbb{Z}_q$. In case it is necessary to ensure for a ciphertext (a, b) encrypts a vote in $\mathbb{G}_q$, one checks whether $a^q = b^q = 1 \bmod p$.

Proof of an encryption of 1: In order to prove that a given ciphertext (a, b) encrypts 1, one has to present a zero-knowledge proof:

$$ZKP\{\exists r : a = g^r \bmod p \wedge b = h^r \bmod p\}$$

The proof, presented in [9], is given in Appendix A.1.

Note, that this as well as further proofs can be made non-interactive according to Fiat-Shamir heuristic [16], by providing the challenge c as a hash function of the "commitment" values sent in the first step of the proof, as well as to other relevant parameters, as suggested in [7].

Proof of knowledge of DSA signing key: Let $(g_s, h_s = g^s) \in \mathbb{G}_q$ be the DSA public key. Proving the possession of valid secret key s could be done with Schnorr's proof of knowledge of discrete logarithm [35], restated in Appendix A.2. The proof is easily extended to a proof of knowledge of an ElGamal plaintext.

Proof of knowledge of RSA signatures: Let (N, e) with e prime be a public RSA signature key, d secret signature key. For a message m and some encoding function $h(m)$ that is used for signing[4], in order to prove the knowledge of a valid signature on m, one has to show:

$$ZKP\{\exists s : s^e \equiv h(m) \bmod N\}$$

The proof, described in [20, 21], is given in Appendix A.3.

[4] Usually a hash value of m and/or padding, according to common RSA signature standards.

Reencryption mix nets: Re-encryption mix nets shuffle a set of ciphertexts without needing to know the private key, and provide a proof of correct shuffling. Modern re-encryption mixnets run efficiently, and many are appropriate for ElGamal encryption, including [17, 19, 28, 41].

It is important to defend against Pfitzmann's attack on mixnet privacy [31], in which a malicious participant copies someone else's vote as a way of exposing it. Our scheme hence requires a proof of the validity and knowledge of the vote when it is posted, as in [6].

Plaintext equality tests: There are two approaches to prove the validity of encrypted vote (i.e. that the plaintext belongs to the set of allowed voting options) while preserving voter privacy. The first is to make the voter attach the proofs of validity during vote casting. This approach is inapplicable to our scheme because the product of several valid votes may be invalid. The second approach is to discard non-valid votes after vote casting and anonymization. A simple way would be to decrypt and publish all the votes. This approach destroys receipt freeness, however. For example, a coercer could demand a particular vote v and then, with 50% probability, either let a voter keep their vote or demand that they add some large number x, and then see whether the $x + v$ appeared in the list of decrypted votes.

To prevent this, instead of decrypting, we use plaintext equality tests ($\mathcal{PET}$) [24]. For a pair of ElGamal ciphertexts $e, e' \in \mathbb{G}_q^2$, $e = Enc_{(g,h)}(v) = (a, b), e' = Enc_{(g,h)}(v') = (a', b')$, these tests are performed in a distributed way by a group of trustees that own the shared corresponding decryption key. The trustees compute and jointly decrypt

$$((\frac{a}{a'})^z, (\frac{b}{b'})^z)$$

for a jointly generated random secret z.

The result is the value of $(\frac{v}{v'})^z$ which is 1 if $v = v'$, or a random value in $\mathbb{G}_q$ that reveals no information about v, v' or their relation to each other otherwise.

Other cryptographic tools: We use the disjunctive witness-hiding proofs of Cramer *et al.* [13]. The trustees share the decryption key using Shamir threshold secret sharing [36], jointly generated using Pedersen's scheme [30], as recommended for Helios by Cortier *et al.* [11].

4 Proposed scheme

In this section we describe the proposed voting scheme. The key generation, decryption trustees, mix nodes, bulletin board, and basic voter behaviour are the same or similar as for Helios. The main difference to Helios is the tallying stage, because we now allow multiple votes against one voter's name. Each row (corresponding to one voter) will be homomorphically totalled, then all the totals will be mixed and interpreted by $\mathcal{PET}$.

If board flooding is considered a possible problem, we could incorporate the token-based mechanisms of [26], which restricts the total number of postings by any individual[5]. Even more simply, we could impose a restriction on the number of ciphertexts that could be posted by each person against each voter's row. Even a small number (such as one or two) might be entirely sufficient, as long as Assumption 2 remained true.

4.1 Preparations

The list of all eligible voters $V_1, ..., V_N$ is posted on the bulletin board, as the list of their public keys and voter IDs. We think of the bulletin board as having a row for each of those eligible voters—in other words, each posted ciphertext is explicitly allocated to one of the voter IDs. The available voting options are represented as $C = \{c_1, ..., c_L\}$, with $1 \notin C$ representing the null/abstaining vote. Using distributed threshold secret sharing [30], the trustees generate a pair of ElGamal keys (g, h) for encrypting the votes. Furthermore, the trustees publish the list of ciphertexts resulting from the deterministic encryption (i.e. using fixed and public randomness value) of voting options: $\widehat{E} = \{\hat{e_1} = Enc_{(g,h)}(c_1), ..., \hat{e_L} = Enc_{(g,h)}(c_L)\}$.

4.2 Vote casting

In order to post a vote $v \in C$ for the voter V_i, one sends an ElGamal ciphertext $e = Enc_{(g,h)}(v)$, and a disjunctive witness-hiding proof given in Algorithm 1.

Algorithm 1 Witness hiding proof of valid vote posting

Public Input: The election ID, the ElGamal ciphertext e to be posted, the public key PK_V of the row to be posted on, the election ElGamal encryption parameters (p, q, g, h).

Poster's private input: Either the randomness used to produce e, or the private key corresponding to PK_V.

Proof: the poster proves
poster knows plaintext of e, using proof from Sec 3.2
AND
$\{$$e$ encrypts 1, that is, is a null vote, using proof from Sec 3.2
OR poster knows the private key corresponding to PK_V, using proof from Sec 3.2[6]$\}$

The proof is made noninteractive using the Fiat-Shamir heuristic applied to the entire public input.

[5] As the complexity of the computations in the tallying stage depends on the amount of eligible voters rather than total cast votes, and the complexity of the computations in the voting stage is linear in the number of cast votes, we presume board flooding is less likely to significantly hinder the election than it is in [25].

An alternative method, applicable in case the RSA-based PKI is used, would be that instead of proving knowledge of the signing key, the prover proves knowledge of a digital signature on $(\texttt{election ID}, e)$. Either way it is important to incorporate all of the public parameters of the proof into the Fiat-Shamir hash, to prevent reuse of the proof of knowledge.

As in the Helios system, the voter has the option to either audit the encrypted ballot via the Benaloh challenge, or to send the vote and the proof. Once the vote is sent, the voter can check the bulletin board in order to verify that it has been recorded. It follows that only the legitimate voter V_i can post non-null votes near her name, since she is the one possessing the secret signature key. If the voter wants to update her vote from v_A to v_B, she encrypts and casts the value of $v_A^{-1} v_B$.

All the cast votes are validated at the moment of posting: exact duplicated postings are removed[7], also discarded are the ciphertexts with invalid zero-knowledge proofs, or those that encrypt a value $v \in \mathbb{Z}_p \backslash \mathbb{G}_q$[8].

4.3 Tallying

At the end of vote casting stage, the bulletin board looks as shown on Table 1. It is assumed that for all $i = 1, ..., N$, it holds that $m_i > 0$. Furthermore, only the voter V_i knows how many of the votes $v_{i,1}, \ldots v_{i,m_i}$ are null votes, and how many are real ones.[9]

Table 1. Bulletin board prior to tallying stage

Voter ID	Cast votes
V_1	$e_{1,1}, \ldots e_{1,m_1}$
$\vdots$	$\vdots$
V_N	$e_{N,1}, \ldots e_{N,m_N}$

The final ciphertext for each V_i is computed by elementwise multiplication

$$e_i = \prod_{j=1}^{m_i} e_{i,j}$$

Since the null votes are all encryptions of 1, only the non-null votes influence the final vote included in tallying. If for some i, the voter V_i has abstained, the resulting ciphertext e_i is an encryption of a null vote.

[6] This step assumes, that the DSA-based PKI is used.

[7] This prevents manipulating someone's vote by re-posting something they have genuinely contributed.

[8] This can be done by checking whether $b^q = 1$ for a ciphertext (a, b) with a valid proof of plaintext knowledge, and is needed to prevent information leakage about plaintext from $\mathcal{PET}s$ during tallying.

[9] This and other assumptions are further discussed in Section 5.

The resulting ciphertexts $e_1, ..., e_N$ are then processed through the mix net for the sake of removing the link between the voter and the decrypted vote, with the anonymized list $e'_1, ..., e'_N$ as output.

For each ciphertext $e'_i = Enc_{(g,h)}(v_i)$ that results from shuffling and each voting option c_j encrypted as $\hat{e_j}$, the trustees perform $\mathcal{PET}s$ for e'_i and $\hat{e_j}$. If the test is positive for some j, the trustees conclude that $v_i = c_j$; otherwise, they conclude that v_i is either a null vote, or an invalid vote, and thus should be discarded from tallying. The result can then be directly computed from the non-discarded votes.

5 Security analysis

In this section we conduct the security analysis of our scheme, by listing the security requirements we want to ensure, and identifying the security assumptions that are needed for them. Further we discuss the ways to ensure the assumptions that we make.

5.1 Security requirements

In general, we need to rely on the following assumptions regarding the cryprography used in the scheme:

- Cryptography Assumption: the cryptography used in witness-indistinguishable proofs and in the ElGamal encryption scheme is reliable:
  - the DDH assumption holds,
  - the random oracle model is instantiated by a hash function,
  - if RSA is used for PKI, the RSA assumption holds.

In the following we explain for each security requirement why it is ensured and under which further security assumptions:

Eligibility verifiability: This requirement suggests, that everyone should be able to verify that only the votes from the eligible voters have been included in the tallying result.

This requirement is ensured due to the application of the proof in Algorithm 1 in Section 4.2 and its soundness (i.e. it that non voters can only cast null votes, which have no effect on the final tallying), if the following assumptions hold:

- Secret Key Leakage Assumption: The voter's secret key is not leaked to the adversary without voter's knowledge.
- Secret Key Re-Usage Assumption: The voting device cannot use the secret key to cast any additional votes without voter's knowledge.
- Authentic List of Keys Assumption: Only eligible voters have their public keys published in the voting register.

Individual verifiability: Each voter should be able to verify, that her vote has been cast and stored by the voting system according to her intention.

This requirement is ensured due to the following mechanisms:

- the application of the Benaloh challenges for cast-as-intended verifiability;
- the possibility for voters to check that the encrypted votes that she submitted herself appear on the bulletin board for stored-as-cast verifiability;
- the functionality to re-send a vote if voters detect that a posting proxy withholds a vote (just like the single bulletin board in Helios); and
- the application of the proof in Algorithm 1 in Section 4.2 and its soundness

if the following assumptions hold:

- Secret Key Leakage Assumption: The voter's secret key is not leaked to the adversary without voter's knowledge.
- Secret Key Re-Usage Assumption: The voting device cannot use the secret key to cast any additional votes without voter's knowledge.

Universal verifiability: Everyone should be able to verify, that the tallying result is computed from the valid votes stored by the voting system only.

This requirement is ensured due to the following mechanisms:

- all the ciphertexts posted on the bulletin board include a valid proof, thus being either encryptions of null votes or of votes from eligible voters,
- these ciphertexts are included in the product defining the final votes,
- the final votes are correctly processed through mix net, and
- each one of the plaintext equality tests outputs the correct result, either assigning a correct valid voting option to each vote, or determining that the vote is invalid

if the following assumption holds:

- Bulletin Board Assumption: The bulletin board is a reliable broadcast channel with memory.

Participation privacy: The system should not disclose the fact, whether an individual voter has participated in the election, to the passive adversary, who only has access to the public output.

This requirement is ensured due to the the application of the proof in Algorithm 1 in Section 4.2 and its soundness if the following assumption hold:

- Hidden Origin Vote Assumption (Passive): There is at least one ciphertext which the adversary is unable to distinguish between a null vote and an effective vote from an eligible voter.

Note, the tallying process does not reveal information about individual votes, or even the presence of particular invalid votes.

Vote secrecy The adversary should not be able to learn for which candidate each individual voter has voted from the public output.

The secrecy of the vote relies on

- the vote anonymization performed in a proper way, and
- the individual ciphertexts not being decrypted prior to being anonymized.

if the following assumption holds:

- Trustee Assumption: Less than a threshold number of trustees disclose their private key shares to the adversary.

Receipt-freeness The voter should not be able to provide a receipt to the adversary, proving that she has voted for a particular candidate c.

The receipt-freeness relies on the fact, that even if the voter proves that she cast a ciphertext encrypting c[10], there are some additional ciphertexts in the voter's row, for which she would not be able to prove that it does not encrypt some value $c' \cdot c^{-1}$ thus replacing c with c'[11]. This is ensured as long as the following assumption holds:

- Hidden Origin Vote Assumption (Active): There is at least one ciphertext in the voter's row, that the voter cannot prove whether she cast it.

An active coercer can force abstention or randomization, and a voter can prove that they have abstained by casting an invalid vote.

5.2 Discussion on the assumptions

We summarize the assumptions identified in Section 5.1 in the list below:

1. Hidden Origin Vote Assumption (Passive): There is at least one ciphertext which the adversary is unable to distinguish between a null vote and an effective vote from an eligible voter.
2. Hidden Origin Vote Assumption (Active): There is at least one ciphertext in the voter's row, that the voter cannot prove whether she cast it.
3. Secret Key Re-Usage Assumption: The voting device cannot use the secret key to cast any additional votes without voter's knowledge.
4. Secret Key Leakage Assumption: The voter's secret key is not leaked to the adversary without voter's knowledge.
5. Trustee Assumption: Less than a threshold number of trustees disclose their private key shares to the adversary.
6. Bulletin Board Assumption: The bulletin board is a reliable broadcast channel with memory.
7. Cryptography Assumption: the cryptography used in witness-indistinguishable proofs and in the ElGamal encryption scheme is reliable:
   - the DDH assumption holds,
   - the random oracle model is instantiated by a hash function,
   - if RSA is used for PKI, the RSA assumption holds.

[10] She can do this for the ciphertext $(g^r, c \cdot h^r)$ by disclosing the randomness r to the adversary.

[11] Note, that v' can be the legitimate vote for another candidate (i.e. the one the voter actually intends to vote for), but also some random or even unknown to the voter value that results in an invalid vote.

8. Authentic List of Keys Assumption: Only eligible voters have their public keys published in the voting register.

We discuss possible ways the assumptions that our scheme requires could be implemented in the following paragraphs:

Hidden Origin Vote Assumption (Passive) and **Hidden Origin Vote Assumption (Active)** can be summarized in following: each voter, against every adversary, must have at least one opportunity to communicate with the bulletin board (perhaps via a third party) and which

– the adversary cannot detect whether the voter has communicated, and
– if the voter doesn't communicate to the BB, some other participant posts a null vote

In other words, it should not be possible for a voter to prove that they have cast all of the ciphertexts in their row that the adversary thinks they might possibly have cast.

It could be implemented by a few ways, for example:

– some posting proxy not colluding with the coercer, with whom the voter can communicate via an untappable channel,
– an anonymous channel to the bulletin board,
– a receipt-free attendance voting scheme (such as Prêt à Voter [33], Wombat [4], StarVote [3] or Scantegrity II [8]) that the voter can attend physically and not tell the coercer about.

This also implies, that at least one posting proxy faithfully adds null ciphertexts in a way that is sufficiently numerous and unpredictable for the adversary, in order to provide adequate cover for voters.

This is enough for participation privacy against a passive coercer. In order to achieve receipt freeness we need to assume that the cryptographic protocol does not allow an actively participating voter to cause Assumption 2 to fail by deviating from the protocol. This will be shown below.

The **Secret Key Re-Usage Assumption** is a particularly strong one for voting, though standard enough for electronic commerce and banking. The issue here is that in standard Helios each voter casts only one vote, and hence can see whether an extra one has been added on their behalf. In our scheme, the whole mechanism relies on the possibility of multiple additive vote casting. The assumption that the smartcard reader doesn't add extras implies a strong trust assumption on the hardware (with, for example, a display/PIN setup). This is also necessary for banking and other contracts, and is what a PIN-based smartcard reader is supposed to achieve. However, this is a stronger assumption than truly end-to-end verifiable protocols, such as basic Helios. We consider it an important aspect of future work to remove this assumption, which was identified as one source of vulnerability in the Estonian Internet voting protocol [37]. Similarly, **Secret Key Leakage Assumption** could also be facilitated by assuming the tamper-resistant and trusted smartcard that stores the key.

An alternative way of realising Secret Key Re-Usage Assumption without trusted hardware is to insist that all votes but one be cast in a physical polling place at which eligibility was carefully established. It would be important to do this in a way that preserved the Hidden Origin Vote Assumption. Again anyone who was not concerned about coercion could simply cast their ordinary vote online, and check that exactly that vote appeared on the bulletin board. There would be no need to trust their hardware not to cast subsequent votes, because there would be no option to cast a second vote remotely.

The **Trustee Assumption** is common in voting protocols that employ distributed decryption, and might be facilitated by selecting e.g. representatives of groups with conflicting interests, such as of different parties as the trustees. The **Bulletin Board Assumption** can be facilitated either by establishing a central server, supervised by trusted third-party observers, or implemented in a distributed way (e.g. [14]). The **Cryptography Assumption** is common to the voting systems based on cryptographic mechanisms, and the **Authentic List of Keys Assumption** is based upon the public list of all eligible voters, the integrity of which is something that should be ensured in traditional elections as well.

6 Efficiency analysis

Assuming T as the number of tallying trustees, that are responsible for both the mixing of the votes and performing the $\mathcal{PET}s$ with t as threshold parameter (usually suggested as $t = \lfloor T/2 \rfloor + 1$), $N' = \sum_{i=1}^{N} m_i$ as all the votes posted during vote casting (including null votes posted by posting proxies), L as number of candidates (for example, $L = 2$ for referendum), following estimations can be made regarding the performance of the scheme. We count the required number of modular exponentiations during each phase, summarizing the findings in Table 2. We assume, that the verifiable mix net scheme proposed in [41] is used during the tallying stage, requiring $8N + 5$ modular exponentiations for the proof of validity, and $9N + 11$ modular exponentiations for its verification. Furthermore, we assume that the DSA-based PKI is used in the election.

Table 2. Efficiency of individual phases

Preparations	$3T + t - 2 + 2L$
Vote casting (cast)	5
Vote casting (verify)	$9N'$
Tallying	$(4T + 5t - 1)NL + (19N + 16)T$

7 Conclusion

We have presented a novel method of achieving private eligibility verifiability and participation privacy by padding the real votes with null votes that are indistinguishable from the non-null ones. With this, the presence of null votes obscures who has actually voted.

Retaining the individual verifiability assumption of ordinary Helios depends on the strong assumption that the voter's signing key cannot be used to post a ciphertext without the voter's knowledge. (This issue does not arise in ordinary Helios because each voter can cast at most one vote.)

The scheme further provides a level of receipt-freeness, preventing voters from proving that they have voted for a particular candidate. The protocol is still susceptible to forced abstention and randomization attacks.

It is important further work to quantify how many and how random padded votes we need. This depends of course on assumptions about collusion between the posting proxies. It is therefore important to consider developing an algorithm which an honest posting proxy should follow when deciding when to cast a null vote.

Usability and public understanding (both important to increase trust in an electronic voting system [39]) remain important open problems. In this system, a voter who doesn't want to participate in the receipt-freeness or participation privacy aspects may simply ignore them and cast a single vote (optionally using the Benaloh challenge for cast-as-intended verification). The possibility that they might have participated is still enough to offer them privacy.

However, possible issues of understandability or usability might arise: for example, the voters might get confused seeing several votes cast in their row, thus leading to distrust in the system; or the need to remember all the previously cast votes in order to be able to update them might become an issue. Many of the complexities of the protocol could be hidden behind a helpful user interface, for example one that remembered what votes had been cast before. Nevertheless the tradeoffs between security, verifiability, public understanding, and ease of use remain challenging, and require further exploration (for example, in forms of user studies).

Acknowledgment

This project (HA project no. 435/14-25) is funded in the framework of Hessen ModellProjekte, financed with funds of LOEWE – Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz, Förderlinie 3: KMU-Verbundvorhaben (State Offensive for the Development of Scientific and Economic Excellence).

References

1. Adida, B.: Helios: Web-based open-audit voting. In: USENIX Security Symposium. vol. 17, pp. 335–348 (2008)
2. Araújo, R., Traoré, J.: A practical coercion resistant voting scheme revisited. In: E-Voting and Identify, pp. 193–209. Springer (2013)
3. Bell, S., Benaloh, J., Byrne, M.D., DeBeauvoir, D., Eakin, B., Fisher, G., Kortum, P., McBurnett, N., Montoya, J., Parker, M., Pereira, O., Stark, P.B., Wallach,

D.S., Winn, M.: STAR-vote: A secure, transparent, auditable, and reliable voting star-vote: A secure, transparent, auditable, and reliable voting system. USENIX Journal of Election Technology and Systems (JETS) 1(1) (August 2013)

4. Ben-Nun, J., Fahri, N., Llewellyn, M., Riva, B., Rosen, A., Ta-Shma, A., Wikström, D.: A new implementation of a dual (paper and cryptographic) voting system. In: 5th International Conference on Electronic Voting (EVOTE) (2012), http://www.wombat-voting.com
5. Benaloh, J.: Simple verifiable elections. In: Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop. pp. 5–5. USENIX Association (2006)
6. Bernhard, D., Cortier, V., Pereira, O., Smyth, B., Warinschi, B.: Adapting helios for provable ballot privacy. In: Computer Security–ESORICS 2011, pp. 335–354. Springer (2011)
7. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In: Advances in Cryptology–ASIACRYPT 2012, pp. 626–643. Springer (2012)
8. Carback, R., Chaum, D., Clark, J., Conway, J., Essex, A., Herrnson, P.S., Mayberry, T., Popoveniuc, S., Rivest, R.L., Shen, E., Sherman, A.T., Vora, P.L.: Scantegrity II municipal election at Takoma Park: The first E2E binding governmental election with ballot privacy. In: Proc. USENIX Security (2010)
9. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Advances in Cryptology—CRYPTO'92. pp. 89–105. Springer (1993)
10. Clark, J., Hengartner, U.: Selections: Internet voting with over-the-shoulder coercion-resistance. In: Financial Cryptography and Data Security, pp. 47–61. Springer (2012)
11. Cortier, V., Galindo, D., Glondu, S., Izabachène, M.: Distributed elgamal à la pedersen: Application to helios. In: Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society. pp. 131–142. ACM (2013)
12. Cramer, R., Damgård, I., MacKenzie, P.: Efficient zero-knowledge proofs of knowledge without intractability assumptions. In: Public Key Cryptography. pp. 354–372. Springer (2000)
13. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Advances in Cryptology—CRYPTO'94. pp. 174–187. Springer (1994)
14. Culnane, C., Schneider, S.: A peered bulletin board for robust use in verifiable voting systems. In: Computer Security Foundations Symposium (CSF), 2014 IEEE 27th. pp. 169–183. IEEE (2014)
15. Essex, A., Clark, J., Hengartner, U.: Cobra: toward concurrent ballot authorization for internet voting. In: Proceedings of the 2012 International Conference on Electronic Voting Technology/Workshop on Trustworthy Elections, EVT/WOTE. p. 3 (2012)
16. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Advances in Cryptology—CRYPTO'86. pp. 186–194. Springer (1987)
17. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: Advances in Cryptology—CRYPTO 2001. pp. 368–387. Springer (2001)
18. Grewal, G.S., Ryan, M.D., Bursuc, S., Ryan, P.Y.: Caveat coercitor: Coercion-evidence in electronic voting. In: Security and Privacy (SP), 2013 IEEE Symposium on. pp. 367–381. IEEE (2013)
19. Groth, J.: A verifiable secret shuffe of homomorphic encryptions. In: Public Key Cryptography—PKC 2003, pp. 145–160. Springer (2002)

20. Guillou, L.C., Quisquater, J.J.: A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In: Advances in Cryptology—Eurocrypt'88. pp. 123–128. Springer (1988)
21. Guillou, L.C., Quisquater, J.J.: A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In: Proceedings on Advances in cryptology. pp. 216–231. Springer-Verlag New York, Inc. (1990)
22. Haenni, R., Spycher, O.: Secure internet voting on limited devices with anonymized dsa public keys. In: Proceedings of the 2011 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections. pp. 8–8. EVT/WOTE'11, USENIX Association (2011)
23. Heiberg, S., Laud, P., Willemson, J.: The application of i-voting for estonian parliamentary elections of 2011. In: E-Voting and Identity, pp. 208–223. Springer (2012)
24. Jakobsson, M., Juels, A.: Mix and match: Secure function evaluation via ciphertexts. In: Advances in Cryptology—ASIACRYPT 2000, pp. 162–177. Springer (2000)
25. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Proceedings of the 2005 ACM workshop on Privacy in the electronic society. pp. 61–70. ACM (2005)
26. Koenig, R., Haenni, R., Fischli, S.: Preventing board flooding attacks in coercion-resistant electronic voting schemes. In: Future Challenges in Security and Privacy for Academia and Industry, pp. 116–127. Springer (2011)
27. Kutyłowski, M., Zagórski, F.: Verifiable internet voting solving secure platform problem. In: Advances in Information and Computer Security, pp. 199–213. Springer (2007)
28. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: Proceedings of the 8th ACM conference on Computer and Communications Security. pp. 116–125. ACM (2001)
29. Neumann, S., Feier, C., Volkamer, M., Koenig, R.: Towards a practical jcj / civitas implementation. In: INF13 - Workshop: Elektronische Wahlen: Ich sehe was, das Du nicht siehst - öffentliche und geheime Wahl. pp. 804–818 (2013)
30. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Advances in Cryptology—CRYPTO'91. pp. 129–140. Springer (1992)
31. Pfitzmann, B.: Breaking an efficient anonymous channel. In: Advances in Cryptology—EUROCRYPT'94. pp. 332–340. Springer (1995)
32. Raykova, M., Wagner, D.: Verifable remote voting with large scale coercion resistance. Tech. rep., Tech. Rep. CUCS-041-11, Columbia (2011)
33. Ryan, P.Y., Bismark, D., Heather, J., Schneider, S., Xia, Z.: Prêt à voter: a voter-verifiable voting system. Information Forensics and Security, IEEE Transactions on 4(4), 662–673 (2009)
34. Schläpfer, M., Haenni, R., Koenig, R., Spycher, O.: Efficient vote authorization in coercion-resistant internet voting. In: E-Voting and Identity, Lecture Notes in Computer Science, vol. 7187, pp. 71–88. Springer Berlin Heidelberg (2012)
35. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of cryptology 4(3), 161–174 (1991)
36. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)
37. Springall, D., Finkenauer, T., Durumeric, Z., Kitcat, J., Hursti, H., MacAlpine, M., Halderman, J.A.: Security analysis of the estonian internet voting system. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. pp. 703–715. ACM (2014)

38. Spycher, O., Koenig, R., Haenni, R., Schläpfer, M.: A new approach towards coercion-resistant remote e-voting in linear time. Financial Cryptography and Data Security pp. 182–189 (2012)
39. Spycher, O., Volkamer, M., Koenig, R.: Transparency and technical measures to establish trust in norwegian internet voting. In: E-Voting and Identity, pp. 19–35. Springer (2012)
40. Srinivasan, S., Culnane, C., Heather, J., Schneider, S., Xia, Z.: Countering ballot stuffing and incorporating eligibility verifiability in helios. In: Network and System Security, pp. 335–348. Springer (2014)
41. Terelius, B., Wikström, D.: Proofs of restricted shuffles. In: Progress in Cryptology–AFRICACRYPT 2010, pp. 100–113. Springer (2010)

A Cryptographic Building Blocks

A.1 Proof of an encryption of 1

In order to prove that a given ciphertext (a, b) encrypts 1, one has to present a zero-knowledge proof:

$$ZKP\{\exists r : a = g^r \bmod p \wedge b = h^r \bmod p\}$$

The proof, presented in [9], is as follows:

1. Prover chooses a random $w \in_R \mathbb{Z}_q$, computes $\alpha = g^w \bmod p$, $\beta = h^w \bmod p$ and sends α, β to the Verifier.
2. Verifier sends the challenge $c \in_R \mathbb{Z}_q$ to the prover
3. Prover computes $u = w + cr \bmod q$ and sends u to Verifier
4. Verifier checks, that $g^u \equiv \alpha a^c \bmod p$ and $h^u \equiv \beta b^c \bmod p$ hold.

The proof has the soundness error of $1/q$.

A.2 Proof of knowledge of discrete log

The following proof can be used to prove knowledge of a DSA or ElGamal signing key, or knowledge of an ElGamal ciphertext.

$$\texttt{Proof of knowledge}\{s : h = g^s\}$$

Public Parameters: ElGamal/DSA parameters (g, h, p, q)
Prover knows: $s : h = g^s \bmod p$.

1. Prover selects a random value $w \in_R \mathbb{Z}_q$ and publishes $a = g^w$.
2. Verifier sends the challenge $c \in_R \mathbb{Z}_q$
3. Prover calculates and publishes $u = w + cs$
4. Verifier checks $g^u = ah^c$

The soundness error of the proof is $1/q$.

A.3 Proof of knowledge of RSA signature

$$\texttt{Proof of knowledge}\{s : s^e \equiv h(m) \bmod N\}$$

Public Parameters: Message m, encoding function $h(m)$, RSA public key (N, e) with e prime

Prover knows: $s : s^e \equiv h(m) \bmod N$, $d : d = e^{-1} \bmod \phi(N)$.

1. Prover selects a random value $r \in_R \mathbb{Z}_N^*$ and calculates $x = r^e \bmod N$
2. Verifier sends the challenge $c \in_R \mathbb{Z}_e$
3. Prover calculates $z = rs^c \bmod N$ and sends z to Verifier
4. Verifier checks $z^e \equiv x \cdot h(m)^c \bmod N$.

The soundness error of the proof is $1/e$. Note, that often the small prime values of e are used as public key in RSA system: commonly, $e = 3$ or $e = 2^{16}+1$. This leads to the proof being insufficiently sound. For this cases, a modification has been proposed in [12], where in order to prove the knowledge of e-th root s of $h(m)$, one proves the knowledge of e^t-th root s' of $h(m) \bmod N$, which can be calculated as $s' = h(m)^{d^t} \bmod N$. The modified proof has the soundness error of $1/e^t$.