Proceedings of the 52nd Hawaii International Conference on System Sciences | 2019

A Proxy Voting Scheme Ensuring Participation Privacy and Receipt-Freeness

Oksana Kulyk
Karlsruhe Institute of Technology
oksana.kulyk@kit.edu

Melanie Volkamer
Karlsruhe Institute of Technology
melanie.volkamer@kit.edu

Abstract

Proxy voting is a form of voting meant to support the voters who want to delegate their voting right to a trusted entity, the so-called proxy. *Depending on the form of proxy voting, the proxy is either authorized to cast a ballot for the voting option that the voter chooses, or to vote according to her own wishes, if the voter is not sure how to vote and wants to delegate the decision making in the election. While the first form of proxy voting has been applied to traditional elections in order to support the voters who are unable to physically get to a polling station, the second form has been a topic of research in Internet voting. Recently, an Internet voting scheme has been proposed, that extends the well-known Helios scheme towards the functionality of proxy voting. This scheme, however, also has the drawbacks of Helios regarding participation privacy and receipt-freeness. As such, the information whether any voter participated in the election either by casting a direct vote or delegating their vote can be deduced from the published information. The scheme furthermore allows both the voters and the proxies to create receipts that prove casting a ballot for a specific candidate, as well as allows the voters to create receipts that prove delegating to a specific proxy. In this work we use the idea of dummy ballots, proposed in another extension of Helios to extend the proxy voting scheme towards participation privacy and receipt-freeness.*

1. Introduction

Well-established forms of voting, such as direct democracy, rely upon the principle of "one voter – one vote". The voter is entrusted to make an informed decision by choosing a candidate or a voting option that she wants to support in the election, and to vote via casting her ballot directly during the election. As an alternative to such direct voting, the concept of *proxy voting* has been introduced, where the voter can appoint a trusted entity, the so-called *proxy*, to vote on this voter's behalf. Proxy voting can serve several purposes. As such, in traditional elections in some of the countries such as Belgium, the voters can vote via proxy if they are physically unable to get to a polling station. The proxy is then assumed to cast a ballot according to the voter's wishes. A further application of proxy voting has been designed to support the voters who might feel overwhelmed with making a decision on how to vote in the election. Hence, by appointing a proxy (who might be a friend, a relative or a trusted expert), the voter delegates her voting right in the election to this proxy. The proxy can then use this delegation to cast a ballot according to her own wishes on behalf of the voter.

As Internet voting has been gaining attention in both research and practical applications in real-world elections [1], a number of Internet voting schemes have also been designed to support the latter case for proxy voting [2, 3, 4, 5]. In particular, one of these schemes [5] extends the well-established Internet voting scheme, Helios [6], towards proxy voting functionality. The scheme in [5] preserves the security properties and user experience of Helios. At the same time, however, it also inherits the privacy weaknesses of Helios. As such, it does not ensure participation privacy, as the identities of the voters who participated in the election, as well as of the voters who delegated their vote[1] are published. Furthermore, the voters in [5], as well as in Helios, are capable of creating receipts that prove to a third party how they voted. This enables vote selling due to the lack of receipt-freeness. The extension in [5] enables the voters to create similar receipts that prove that they delegated to a specific proxy.

Our contribution is addressing these weaknesses and proposing an Internet voting scheme for proxy voting that ensures participation privacy and receipt-freeness. In particular, we use the idea of dummy ballots introduced in the Helios extension that has been proposed in [7] and its security proven in [8, 9]. Our extension ensures

[1] Note that some versions of Helios ensure participation privacy by using pseudonyms instead of voter identities. However, such an approach comes with a cost of eligibility verifiability, since the entity assigning the pseudonyms should be trusted to assign them only to eligible voters.

URI: https://hdl.handle.net/10125/59721
ISBN: 978-0-9981331-2-6

participation privacy by hiding the information, whether a given eligible voter participated in the election, cast her ballot directly or delegated to some proxy. It furthermore ensures receipt-freeness by preventing the voters and the proxies from creating receipts that prove that they voted for a specific voting option, and by preventing voters from creating receipts that prove that they delegated to a specific proxy.

2. Security Requirements

In this section we outline the security requirements we aim to ensure in our work. We base this list on the previous work in [5] with the addition of the requirements of participation privacy and receipt-freeness.

We first provide the list of requirements that are not specific to the delegation.

Vote privacy The voting system should not provide any information to establish a link between the voter and her vote, aside from what is available from the election result. Furthermore, the voter should be unable to create a receipt that proves that she voted for a specific voting option (*receipt-freeness*). Note that we do not consider cases of forced abstention, whereby the voter is requested to provide proof to the adversary that she did not participate in the election or cast an invalid ballot. We only consider cases where voters are requested to provide a proof for voting for a particular voting option.

Eligibility Only the ballots cast by eligible voters should be included in the tally.

Vote integrity All the cast ballots should be correctly processed and included in the tally result.

Availability It should be possible to compute the election result, even if some of the components of the voting system are faulty.

Participation privacy The voting system should not reveal whether a given voter has abstained, cast a direct ballot or delegated in the election.

Note that 'vote privacy' and 'vote integrity' should also hold for proxies casting delegated ballots.

The next set of requirements concerns the security of the delegation process.

Delegation eligibility The proxy should only be able to cast delegated ballots on behalf of eligible voters.

Delegation integrity A proxy should only be able to cast a delegated ballot on behalf of the voter, if this voter has authorized the proxy to do so.

Delegation integrity for proxies The valid votes cast by proxies are correctly included in the final tally.

Delegation privacy The voting system should not provide any information to establish a link between the delegating voter and the voter's chosen proxy. Furthermore, the proxy herself should not be able to tell which voter has delegated to her. Furthermore, the voter should not be able to create a receipt that proves that she delegated to a specific proxy (receipt-freeness).[2]

Delegation power privacy The voting system should not provide any information about the delegating power of a proxy, i.e. the number of eligible voters who delegated to this proxy.

3. Background

In this section we describe the background information required for the understanding of our scheme.

3.1. Cryptographic primitives

We first describe the cryptographic primitives used in our scheme. For encrypting the data in our scheme, the **ElGamal cryptosystem**[10] is used, with $\mathsf{Enc}_{\mathsf{pk}}(r, m)$ denoting an encryption of the plaintext m given public key pk and randomness value r. The ciphertexts are decrypted via **threshold verifiable distributed decryption**, as described by Pedersen [11].

We furthermore use a variety of **zero-knowledge proofs**, such as the proof of discrete logarithm knowledge [12]. We furthermore use the techniques as described in [13] for constructing disjunctive proofs that prove the validity of one out of several statements without revealing the information on which statement is true, and the generalized techniques in [14] for constructing proofs of general statements about discrete logarithms. In the paper, we use the following notation: given an example for proving the knowledge of discrete logarithm $x = \log_g h$ for public parameters g, h and secret x, the proof of knowledge π is denoted as $\pi = PoK\{x : g^x = h\}$. The non-interactive proofs are constructed with strong Fiat-Shamir heuristic as described in [15].

For checking whether two ciphertexts encrypt the same plaintext without revealing any further information about them, **plaintext equivalence tests** (PETs) [16] are used. The test $\mathsf{PET}(c_1, c_2)$ performed by several entities in a distributed way and outputs 1 if c_1 and c_2 encrypt the same plaintext, and a random value otherwise.

[2] Note that the relevance of this requirement might differ depending on the election setting. We include it in our work, considering the settings where it might be considered a breach of privacy if it becomes known to a third party, which proxy the voter trusts. Furthermore, the voter's inability to create receipts that prove voting for a specific proxy would prevent the forms of voter coercion, where the voter is forced or persuaded to delegate to a malicious proxy, e.g. as a threat or in exchange for a monetary reward.

In order to anonymize the list of ciphertexts before decrypting, a **mix net shuffle** is used, ensuring that the link between the ciphertexts in the input list $c_1, ..., c_N$ and the shuffled output list $\mathsf{Mix}(c_1, ..., c_N) = c'_1, ..., c'_N$ remains secret. The same approach can be used for mixing the tuples of ciphertexts $\bar{c_1}, ..., \bar{c_N}$ with $\bar{c_i} = (c_{i,1}, ..., c_{i,k})$. In order to prove that the ciphertexts have not been manipulated during mixing, the proof of shuffle validity (e.g. as proposed in [17]) is used.

3.2. Proxy Voting Scheme in [5]

We now describe the scheme that extends Helios [6] towards proxy voting functionality as proposed in [5].

Setup The voters submit their public signing keys pk_i that are published on the bulletin board. Futhermore, each voter id_i prepares a set of T *delegation credentials* g^{x_i}, with the secret delegation keys x_i known only to the voter herself[3]. These delegation credentials are also published on the bulletin board next to the voter identities. The tabulation tellers generate an ElGamal election key, the public part of which $\mathsf{pk} = (g, h)$ is also published on the bulletin board. Futhermore, the list of valid voting options $\{v_1, ..., v_L\}$ is published.

Delegating In order to delegate her vote, the voter id_i computes a so-called *delegation token* by encrypting her delegation credential g^{x_i} as $c_d = (a_d, b_d) = \mathsf{Enc}_{\mathsf{pk}}(r_d, g^{x_i})$. She furthermore computes $\sigma = g^m$ of a randomly chosen value $m \in \mathbb{Z}_q$ and a non-interactive signature of secret key knowledge on σ, $\pi_d = PoK\{(r_d, x_i) : a_d = g^{r_d} \wedge b_d = g^{x_i} h^{r_d}\}(\sigma)$[4]. The purpose of π_d is to prevent a violation of delegation integrity whereby a third-party delegates on behalf of the voter by taking the published delegation credential g^{x_i} without knowing x_i. Note that while a third-party can construct a delegation token for an arbitrary value $g^{x'}$ with randomly chosen x', such a token will be discarded during tally unless $g^{x'}$ is in the list of published delegation credentials g^{x_i} from eligible voters. Incorporating σ into π_d furthermore prevents a non-authorized proxy from using the resulting delegation token to cast a delegated vote for id_i. The values (σ, m, c_d, π_d) then serve as a delegation token, with (σ, c_d, π_d) as public and m private. The delegation token (σ, m, c_d, π_d) are sent to the voter's chosen proxy over a private anonymous channel.

[3]The scheme in [5] also allows the voter to delegate to multiple proxies with different priorities by using several delegation credentials $g^{x_{i,j}}$ for each voter; however, for the sake of simplicity, we describe the scheme where each voter has only one priority.

[4]The construction of π_d is described in [5].

Voting For casting a **direct ballot**, the voter id_i encrypts her chosen voting option as $(a, b) = \mathsf{Enc}(\mathsf{pk}, v_i)$[5]. For casting a **delegated ballot**, the proxy encrypts her chosen voting option v_i as $c_v = (a_v, b_v) = \mathsf{Enc}_{\mathsf{pk}}(r, v_i)$ for some randomness r, and computes the zero-knowledge proof $\pi_v = PoK\{m \in \mathbb{Z}_q : \sigma = g^m\}$. The proof π_v is meant to ensure that the proxy knows the private part of the corresponding delegation token m, hence, was authorized to delegate by the voter who submitted the delegation token (σ, c_d, π_d). The tuple $(\sigma, c_v, \pi_v, c_d, \pi_d)$, which includes the public part of the delegation token and the delegated ballot, is then submitted by the proxy to the bulletin board over an anonymous channel. If the voter changes her mind after delegating, she **cancels her delegation** by casting a direct ballot.

Tallying During the tallying, the delegated ballots cast by the proxies are first processed by the tabulation tellers. The ballots with non-valid proofs are removed, and the ballots cast using the same delegation token are processed according to vote updating policy[6] and the remaining tuples are processed through a mix net with each tabulation teller acting as a mix node. The delegation credentials c_d attached to the anonymized delegation ballots are jointly decrypted by the tabulation tellers, and the ballots with either 1) invalid delegation credentials (i.e. the ones not among the published credentials g^{x_i} for an eligible voter id_i), or 2) delegation credentials from the voters who cast a direct vote, are removed from further tally. The remaining delegated ballots are being finally processed together with direct ballots by anonymizing them, e.g. via applying a mix net[7], and the anonymized result is being jointly decrypted by the tabulation tellers.

The scheme in [5] relies on the following assumptions to ensure the security requirements described in Section 2 with the exception of participation privacy and receipt-freeness:

(A-PrivChannels) The channels between the voters and the proxies are private and authenticated.

(A-AnonChannels) The channels between the voters and the proxies, as well as between the proxies and the bulletin board, are anonymous.

[5]As mentioned in [15], it is also advisable that the voter submits a proof of plaintext knowledge, i.e. the knowledge of v_i, in order to prevent ballot-copying attacks.

[6]For example, given the "last vote counts" policy, all the ballots but the latest one, cast using the same delegation token, are being removed from further processing.

[7]Another variant of anonymizing the ballots, proposed in [5] as well as in the later implementation of Helios [18] and in the original protocol it was based upon [19], is the use of homomorphic tallying approach, which, for the sake of simplicity, is not described in this paper.

(A-ProxySemiHonest) The proxies are semi-honest, meaning that they do not deviate from the protocol.

(A-TabTellerHonest) More than half of tabulation tellers are honest and capable of communicating with each other and the bulletin board.

(A-VotDeviceLeakage) The voting devices of the voters as well as those of the proxies do not leak information to an adversary.

(A-NoBBModification) The bulletin board does not remove or modify the data that is published on it.

(A-BBConsistency) The bulletin board shows the same contents to everyone.

(A-NoCoercion) Coercion or vote selling do not occur.

(A-CompRestricted) The adversary is computationally restricted.

(A-Verify) The voters and the proxies perform the verifications available to them within the system.

(A-VerDeviceTrusted) The verification devices of both voters and proxies are trustworthy.

(A-VotRegister) The voting register with the eligible voters public signing keys is trustworthy.

3.3. Privacy Improvements of Helios in [8, 7]

The idea of the extension in [8, 7] is to use the so-called *dummy ballots* cast by the *posting trustee* in order to obfuscate the presence of ballots cast by voters, ensuring participation privacy or receipt-freeness (by allowing voters to update their vote via casting an additional ballot without the adversary noticing). These dummy ballots are designed to be indistinguishable from non-dummy ballots while having no effect on the tally result. The scheme in [8, 7] runs as follows:

Setup The public signing keys of eligible voters are published on the bulletin board. The tabulation tellers jointly generate the election key, the public part $\mathsf{pk} = (g, h)$ of which is published as well, together with the list of valid voting options $\{v_1, ..., v_L\}$.

Voting In order to **cast her ballot**, the voter id_i encrypts her chosen voting option as $(a, b) = \mathsf{Enc}_{\mathsf{pk}}(r, v)$ for some randomness r. She further computes the so-called eligibility proof: given s as a valid signature on (a, b) with a corresponding public signing key pk_i, the voter computes the proof of knowledge π as $\pi = PoK\{s, r : \mathsf{VerifySign}(\mathsf{pk}_i, s, (a, b)) = 1 \vee g^r = a \wedge h^r = b\}$. Thereby, $\mathsf{VerifySign}\mathsf{pk}_i, s, (a, b)$ signifies the function of verification of a digital signature. The proof is computed using the proof of discrete logarithm equality [20] and proof of signature knowledge [21], combined using the technique of disjunctive proofs [13]. Hence, only given the possession of the valid signature s, the proof for a non-dummy vote (i.e. for a ciphertext encrypting $v \neq 1$) can be computed. The values $id_i, (a, b), \pi$ are sent by the voter to the bulletin board over an anonymous channel. The purpose of the anonymous channel is to disguise, whether the ballot comes from the voter id_i herself or from a posting trustee casting a dummy ballot on behalf of the voter.

For each eligible voter id_i, the posting trustee casts a random number of **dummy ballots**. Namely, the posting trustee computes $(a, b) = \mathsf{Enc}_{\mathsf{pk}}(r, 1)$ for some randomness r and the eligibility proof π as described above, using the public signing key pk_i. Note, as (a, b) encrypts a dummy vote, the posting trustee no longer requires the knowledge of s in order to compute π.

If the voter wants to **update her vote** from the voting option v_A to v_B, she computes and submits a new ballot encrypting v_B/v_A.

Tallying The ballots next to each voter id are being multiplied, resulting in the list of final ballots $c_1, ..., c_N$. Note that due to additively homomorphic property of ElGamal, the final ballot for each voter encrypts the product of all the votes submitted by the voter, and as the dummy ballots encrypt a dummy vote (i.e. 1), they do not affect the content of the final ballot. The final ballots are being shuffled via mix net by the tabulation tellers. Finally, for each anonymized ciphertext c, the plaintext equivalence tests are being applied to check whether c encrypts a valid voting option $v_i \in \{v_1, ..., v_L\}$. In case a match is found via $PETs$, the corresponding voting option is output, otherwise c is counted as a dummy vote and discarded from the tally.

The scheme in ensures the security requirements outlined in Section 2 under the following assumptions[8]:

(A-TabTellerHonest) More than half of the tabulation tellers are honest.

(A-PosTrusteeHonest) At least one posting trustee is honest.

(A-VotDeviceLeakage) The voting devices used by the voters do not leak information to an adversary.

(A-NoBBModification) The bulletin board does not remove or modify the data published on it.

[8] A more detailed specification of the assumptions together with the corresponding security proofs is provided in [8, 9].

(A-BBConsistency) The bulletin board shows the same contents to everyone.

(A-CompRestricted) The adversary is computationally restricted.

(A-Verify) The voters perform the verifications which are available to them within the system.

(A-VerDeviceTrusted) The verification devices of voters are trustworthy.

(A-VotRegister) The voting register with the list of eligible voters public signing keys is trustworthy. If the voters' public keys are only available on the bulletin board, the assumptions (A-NoBBModification, A-BBConsistency) are further required.

(A-AnonChannels) The channels between the honest voters and the bulletin board, as well as between the posting trustees and the bulletin board, are anonymous.

(A-NoForcedAbstention) Coercion in form of forced abstention does not occur.

(A-NoUnknownPlaintext) The adversary does not cast ballots on behalf of the voter which plaintexts the voter does not know.

(A-HiddenVote) The voters have the possibility to cast a ballot without being observed by the adversary.

(A-Unpredictable) The number of dummy ballots cast by the posting trustee on behalf of the posting trustee is unpredictable to the adversary.

The assumption (A-NoUnknownPlaintext), in particular, is crucial for the design of the scheme in [8, 7] in order to ensure that the voter can still vote for her preferred voting option in case of coercion or vote buying. If the assumption is not fulfilled, the voter can change the ballot cast by an adversary to an invalid vote without being noticed, but not to a vote for a specific voting option of voter's choice. Note that the assumption can be ensured as long as the adversary does not have access to the voter's credentials (e.g. the eID smartcard) used for casting the vote and as long as there is no two-way communication between the adversary and the voter during vote casting, which is unlikely for large-scale vote buying.

4. Our Scheme

In this section we describe our proposed extension of the scheme in Section 3.2 with the privacy improvements of participation privacy and receipt-freeness.

4.1. Modifications

We first provide an overview of the modifications introduced into our extension.

Posting Trustee and Dummy Ballots As in the scheme proposed in [8, 7], our extension relies on the so-called *dummy ballots* cast during the voting by a special kind of entity, the *posting trustee*. The dummy ballots in [8, 7] are designed to be indistinguishable from non-dummy ballots and have no effect on the tally result. They are meant to obfuscate the presence of ballots cast by voters, ensuring participation privacy or receipt-freeness (by allowing voters to update their vote via casting an additional ballot without the adversary noticing). In the extension proposed in this paper, we introduce dummy ballots both for direct ballots and for the delegated ballots.

The dummy ballots for direct ballots are constructed in the same way as described in Section 3.3. A similar concept is applied to obfuscate the ballots cast by the proxies. These dummy ballots are constructed in the following way. We modify the proofs π_d and π_v that are attached to the delegation token and the corresponding delegated ballot. Namely, the proof π_d should enable the posting trustee to construct dummy delegation tokens with the credentials g^{x_i} without knowing x_i; however, these dummy tokens should only enable casting dummy votes. For this purpose, we require an independent generator $\hat{g}$, so that $\log_g \hat{g}$ is unknown. The delegation token on behalf of the voter id_i (computed either by the voter herself or by the posting trustee) consists of the following values: (1) a ciphertext $c_d = (a_d, b_d) = \mathsf{Enc}_{\mathsf{pk}}(r_d, g^{x_i})$, (2) a value $\sigma = \hat{g}^m$ or $\sigma = g^m$ for a random m, (3) a proof of knowledge π_d as $\pi_d = PoK\{(r_d, x_i, m) : \sigma = \hat{g}^m \vee a_d = g^{r_d} \wedge b_d = g^{x_i} h^{r_d}\}$.

Given that only the voter knows the value of x_i, only she can cast delegation tokens using $\sigma = g^m$, while the posting trustee has to set $\sigma = \hat{g}^m$ in constructing her tokens. The proof is provided in Algorithm 1.

For casting a delegated ballot $c_v = (a_v, b_v)$ corresponding to the delegation token $(\sigma, (a_d, b_d), \pi_d)$, then, the proxy or the posting trustee computes a proof of knowledge $\pi_v = PoK\{(r_v, m) : a_v = g^{r_v} \wedge b_v = h^{r_v} \vee \sigma = g^m\}$. The construction of the proof is described in Algorithm 2. Since g and $\hat{g}$ are independent generators, and for a given σ one can only know the value of either $\log_g \sigma$ or $\log_{\hat{g}} \sigma$, it follows that the dummy delegation tokens can be only used to cast dummy votes, i.e. $(a_v, b_v) = \mathsf{Enc}_{\mathsf{pk}}(r_v, 1)$ for some randomness r_v.

Our scheme requires casting a random number of dummy ballots for both direct and delegated ballots. The choice of a random distribution for determining

Algorithm 1 Proof of knowledge for constructing the delegation token for the voter i

Private input: $m, r_d, x_i \in \mathbb{Z}_q$
Public Input: $(g,h),(a_d,b_d) \in \mathbb{G}_q^2, \sigma, \hat{g} \in \mathbb{G}_q$
Proof:
$w_1, w_2, w_3, e' \leftarrow \mathbb{Z}_q$
if $(a_d, b_d) = (g^{r_d}, g^{x_i} h^{r_d})$ **then**
  $t_1 \leftarrow g^{w_1}, t_2 \leftarrow g^{w_2} h^{w_1}, t_3 \leftarrow \sigma^{e'} \hat{g}^{w_3}$
else
  $t_1 \leftarrow a_d^{e'} g^{w_1}, t_2 \leftarrow a_d^{e'} g^{w_2} h^{w_1}, t_3 \leftarrow \hat{g}^{w_3}$
end if
$e \leftarrow H(\sigma||\hat{g}||g||h||a_d||b_d||t_1||t_2||t_3)$
if $(a_d, b_d) = (g^{r_d}, g^{x_i} h^{r_d})$ **then**
  $e_1 \leftarrow e - e', e_2 \leftarrow e'$
  $s_1 \leftarrow w_1 - r_d e_1, s_2 \leftarrow w_2 - x_i e_1, s_3 \leftarrow w_3$
else
  $e_1 = e', e_2 = e - e'$
  $s_1 \leftarrow w_1, s_2 \leftarrow w_2, s_3 \leftarrow w_3 - m e_2$
end if
$\pi_d \leftarrow (e_1, e_2, t_1, t_2, t_3, s_1, s_2, s_3)$
Verification:
$e \leftarrow H(\sigma||\hat{g}||g||h||a_d||b_d||t_1||t_2||t_3)$
if $a_d^{e_1} g^{s_1} = t_1 \wedge b_d^{e_1} g^{s_2} h^{s_1} = t_2 \wedge \sigma^{e_2} \hat{g}^{w_3} = t_3 \wedge e_1 + e_2 = e$ **then**
  $\mathsf{Verify}(\pi_d) = 1$
else
  $\mathsf{Verify}(\pi_d) = 0$
end if

Algorithm 2 Proof of knowledge for constructing the delegation token for the voter i

Private input: $m, r_v \in \mathbb{Z}_q$
Public input: $(g,h),(a_v,b_v) \in \mathbb{G}_q^2, \sigma \in \mathbb{G}_q$
Proof:
$w_1, w_2, e' \leftarrow \mathbb{Z}_q$
if $(a_v, b_v) = (g^{r_v}, h^{r_v})$ **then**
  $t_1 \leftarrow g^{w_1}, t_2 \leftarrow h^{w_1}, t_3 \leftarrow \sigma^{e'} g^{w_2}$
else
  $t_1 \leftarrow a_v^{e'} g^{w_1}, t_2 \leftarrow b_v^{e'} g^{w_2} h^{w_1}, t_3 \leftarrow g^{w_2}$
end if
$e \leftarrow H(\sigma||g||h||a_v||b_v||t_1||t_2||t_3)$
if $(a_v, b_v) = (g^{r_v}, h^{r_v})$ **then**
  $e_1 \leftarrow e - e', e_2 \leftarrow e'$
  $s_1 \leftarrow w_1 - r_v e_1, s_2 \leftarrow w_2$
else
  $e_1 = e', e_2 = e - e'$
  $s_1 \leftarrow w_1, s_2 \leftarrow w_2 - m e_2$
end if
$\pi_d \leftarrow (e_1, e_2, t_1, t_2, t_3, s_1, s_2)$
Verification:
$e \leftarrow H(\sigma||g||h||a_v||b_v||t_1||t_2||t_3)$
if $a_v^{e_1} g^{s_1} = t_1 \wedge b_v^{e_1} h^{s_1} = t_2 \wedge \sigma^{e_2} g^{w_2} = t_3 \wedge e_1 + e_2 = e$ **then**
  $\mathsf{Verify}(\pi_v) = 1$
else
  $\mathsf{Verify}(\pi_v) = 0$
end if

this number influences the level of security the scheme provides, as well as the efficiency of the scheme. We refer to [8] for the analysis of how exactly the choice of random distribution for the number of dummy ballots influences the security of the scheme, as well as for some of the numerical examples for particular distributions.

Note that the described constructions of the zero-knowledge proofs do not require any credentials from the posting trustee. Similar to the proposal in [8, 7], this approach allows everyone, including the voters themselves, to take over the role of the posting trustees, thus extending the trust distribution of the scheme (recall that the scheme in [8, 7] relies on the assumption that at least one of the posting trustees is trustworthy). It, however, has a downside, as allowing everyone to cast dummy ballots could potentially lead to so-called board flooding, thus hindering the efficiency of the election. A possible alternative approach would be amending the zero-knowledge proofs to include the proof of knowledge for a secret credential of the posting trustee, if a dummy ballot is cast. Choosing between the approaches would require a careful consideration of an optimal trade-off between efficiency and privacy, which is the topic of future work. Hence, in this paper we consider the proofs as described above, without requiring secret credentials from the posting trustee.

Filtering Between Direct and Delegated Ballot We consider a list of valid voting options $\mathbb{V} \subset \mathbb{G}_q$ and a predetermined value $d \in \mathbb{G}_q \setminus \mathbb{V} \cup \{1\}$ that indicates that the voter chose to delegate her vote. We furthermore require a function $\mathsf{Filter} : \mathbb{G}_q^4 \to \mathbb{G}_q$ defined as follows:

$$\mathsf{Filter}(c, c') = \begin{cases} \mathsf{Dec}(c) & \text{if } \mathsf{Dec}(c) \in \mathbb{V} \\ \mathsf{Dec}(c') & \text{if } \mathsf{Dec}(c) = d \\ & \quad \wedge\, \mathsf{Dec}(c') \in \mathbb{V} \\ 1 & \text{otherwise} \end{cases}$$

This function can be implemented via PETs and mix net shuffle and is performed by the tabulation tellers as described in Algorithm 3. Its purpose is to filter the ballots in the following way: given c as the encryption of a voting option in a direct ballot c' as the encryption of a voting option in a delegated ballot from the same voter, Filter outputs the plaintext of c if it is a valid vote, the plaintext of c' if the voter indicated that she delegates by casting d and the corresponding proxy casts a ballot for a valid voting option, and a dummy vote in all the other cases. In case either c or c' does not encrypt a valid voting option, no further information is revealed about the corresponding plaintext.

4.2. Description

The modified scheme can be described as follows.

Algorithm 3 Function $\mathsf{Filter}(c, c')$

Input: $\{v_1, ..., v_L\}$, d, c, c'
$\{c_1^{(v)}, ..., c_L^{(v)}\} \leftarrow \mathsf{Mix}(\mathsf{Enc_{pk}}(1, \mathsf{v_1}), ..., \mathsf{Enc_{pk}}(1, \mathsf{v_L}))$
for $i = 1, ..., L$ **do**
  if $\mathsf{PET}(c, c_i^{(v)}) = 1$ **then**
    return $\mathsf{Dec(c)}$
  end if
end for
if $\mathsf{PET}(c, \mathsf{Enc_{pk}}(1, d)) = 1$ **then**
  for $i = 1, ..., L$ **do**
    if $\mathsf{PET}(c', c_i^{(v)}) = 1$ **then**
      return $\mathsf{Dec(c')}$
    end if
  end for
end if
return 0

Setup The setup runs as described in Section 3.2. The voters submit their public signing keys pk_i and their delegation credentials g^{x_i} that are published on the bulletin board. The tabulation tellers generate an ElGamal election key, the public part of which $\mathsf{pk} = (g, h)$ is also published on the bulletin board. Futhermore, the list of valid voting options $\{v_1, ..., v_L\}$, the value $d \notin \{v_1, ..., v_L\}$ and a generator $\hat{g}$ so that $g, \hat{g}$ are independent, are published.

Delegating The delegation occurs the same way as in Section 3.2. The voter id_i encrypts her delegation credential g^{x_i} and computes $\sigma = g^m$ of a randomly chosen value $m \in \mathbb{Z}_q$ with the proof π_d as described in Section 4.1. She then sends the resulting *delegation token*, i.e. the values $(\sigma, m, (a_d, b_d), \pi_d)$ to the voter's chosen proxy over a private anonymous channel. For the sake of ensuring receipt-freeness for the proxies, the voter is furthermore encouraged to submit a random number of delegation tokens using the same g^{x_i} to her chosen proxy[9]. In order to finalize her delegation, the voter casts a direct ballot for d.

Voting Casting a **direct ballot** occurs as described in [8, 7]. The voter id_i encrypts her chosen voting option as $(a, b) = \mathsf{Enc_{pk}}(r, v_i)$ and a non-interactive zero-knowledge proof $\pi = PoK\{s, r : a = g^r \wedge b = h^r \vee \mathsf{VerifySign}(\mathsf{pk}_i, s, (a, b))\}$. The proof, also described in [8, 7], is meant to show that either the cast ballot encrypts a dummy vote, or the person who casts it knows the valid signature of the voter id_i. The voter furthermore computes a proof of plaintext knowledge $\pi_p = \{r, v : a = g^r \wedge b = vh^r\}$ as in [15].

Casting a **delegated ballot** occurs the same way as in Section 3.2. The proxy encrypts her chosen voting option as $(a_v, b_v) = \mathsf{Enc}(\mathsf{pk}, v_i)$ and computes the zero-knowledge proof $\pi_v = PoK\{m \in \mathbb{Z}_q : \sigma = g^m\}$ which serves as a proof of knowledge for m and the proof of plaintext knowledge π_p. The tuple $(\sigma, c_v, \pi_v, c_d, \pi_d, \pi_p)$ is then submitted by the proxy to the bulletin board over an anonymous channel.

As in [5], the delegation can be **cancelled** by the voter via casting a direct vote for any voting option $v \neq d$.

In addition to the ballots cast by the voters and the proxies, each posting trustee casts a random number of **dummy ballots**[10], both direct and delegated, on behalf of each voter. The dummy ballot for a direct vote is constructed the same way as in [8, 7]: the posting trustee chooses an eligible voter id_i, encrypts a dummy vote as $(a, b) = \mathsf{Enc_{pk}}(r, 1)$ and computes the proof $\pi = PoK\{s, r : a = g^r \wedge b = h^r \vee \mathsf{Verify}((a, b), \mathsf{sk}_i)\}$ as well as the proof of plaintext knowledge π_p. Note that as (a, b) encrypts a dummy vote, the proof does not require the knowledge of a valid signature by id_i, and due to its zero-knowledge property it is indistinguishable from the proofs π submitted with the non-dummy ballots. For delegated vote, the posting trustee computes the proof as described in Section 4.1 by submitting a tuple $(\sigma, c_v, \pi_v, c_d, \pi_d)$ as a delegated ballot. Note, unless the posting trustee knows the value of x_i, she can only cast delegated ballots with a dummy vote.

The voter can **update** her direct vote v_A to v_B by casting an additional ballot for v_B/v_A, as in [8, 7]. The proxy has two possibilities to update her delegated vote. One possibility is to reuse one of her delegation tokens and cast an additional ballot with the same token, so that all but the last ballots with the same token will be discarded. Alternatively, if the proxy wants to deny updating her vote, she can use another delegation token received from the voter during delegation. In the second case, same as for updating a direct vote, the proxy submits a delegated ballot for v_B/v_A if she wants to change her vote from v_A to v_B.

Tallying During the tallying, the ciphertexts next to each voter that represent her direct ballot are multiplied, the same way as in [8, 7], forming a list of ciphertexts $c_1, ..., c_N$ that represent final direct ballots. The delegated ballots are processed in the following way. First, the ballots with non-valid proofs and all ballots except latest one that was cast latest with the same delegation tokens are removed. The remaining tuples (c_v, c_d) are processed though a mix net. Afterwards, the delegation credentials c_d attached to the ballots are decrypted. Then the delegated ballots cast using the same delegation credential (which include dummy delegated ballots) are multiplied, forming a list of tuples $c'_1, ..., c'_N$,

[9] For the sake of better usability, this process can be automated by the voting software.

[10] As mentioned in Section 4.1, we refer to [8, 9] for the ways to decide how this random number is chosen.

representing the list of final delegated ballots. Thereby, the ballots that are attached to an invalid delegation credential, i.e. the ones that do not belong to any eligible voter, are discarded similar to [5]. As a result, for the voters $id_1, ..., id_N$ a list of ciphertexts $c'_1, ..., c'_N$ is formed, whereby c'_i denotes an encrypted voting option cast in a delegated ballot for the voter id_i, and $c'_i = \mathsf{Enc}_{\mathsf{pk}}(r, 1)$ for some randomness r for the voters, on which behalf no delegated ballots have been cast. The tuples (c_i, c'_i) are further anonymized via mix net, and afterwards the function f described in Section 4.1 is applied in order to assign each tuple to either a valid voting option or a dummy vote.

5. Security Evaluation

In this section we provide an informal security evaluation of our extension[11]. We aim to show, that our scheme relies on the same assumptions as in Section 3.2 and Section 3.3, with the exception of providing protection against vote or delegation selling (i.e. receipt-freeness). We furthermore accept two additional assumptions:

(A-DelBallotsUnpredicted) The number of dummy ballots and of dummy delegated ballots on behalf of each voter is unpredictable to the adversary. Note that this assumption mirrors the assumption (A-Unpredicted) in Section 4.1.

(A-DelTokensUnpredicted) The number of delegation tokens sent by the voter with each delegation is unpredictable to the adversary.

Vote privacy We first consider vote privacy for the voters and the proxies who do not attempt to create receipts to prove how they voted. The process of casting a direct vote is the same as in the scheme in [8, 7], hence, vote privacy is ensured under the same assumptions. Similarly, the process of casting a delegated ballot is the same as in [5], aside from the changes to the proof π_d. As the proof itself does not reveal any information on the content of the ballot, vote privacy for delegated ballots is preserved under the same assumptions as in Sections 3.2 and 3.3.

We now consider the receipt-freeness of our scheme. As mentioned in Section 2, similar to [8], we consider receipt-freeness as an inability for the voter to create a receipt for voting for a specific valid voting option, hence, excluding forced abstention. Both for voters and for proxies, the receipt-freeness, similar to the approach in [8, 7], relies on the existence of a so-called *counter-strategy*, that the voter (or, correspondingly, the proxy) should apply in order to fake a receipt. Receipt-freeness implies, that an adversary cannot tell whether a counter-strategy has been applied, hence, distinguish between a real and a fake receipt.

We first consider the receipt-freeness for direct ballots. As in [8, 7], it relies on deniable vote updating: when ordered to cast a ballot for the voting option v, the voter casts an additional ballot for $v' - v$, so that her final ballot is included in the tally as a vote for v'. As the process of casting a direct ballot is unchanged from [8, 7], the inability of the adversary to guess, whether there is an additional ballot updating her vote relies on the same assumptions as in Section 4.1.

If the voter is requested to provide a receipt for delegating to a specific proxy, the appropriate counter-strategy would be to cancel her delegation by casting a direct ballot. Given that the adversary is unable to detect such a ballot (which is ensured under the same assumptions as in Section 3.3 and the assumption (A-DelBallotsUnpredicted)) and the proper anonymization during tallying (ensured under the assumption, that the majority of the tabulation tellers is honest (A-TabTellersHonest) and the adversary is computationally restricted (A-CompRestricted)), receipt-freeness is ensured[12].

We finally consider the receipt-freeness for the proxies. Recall, that the proxy gets a random number of delegation tokens from the voter. Hence, as long as the adversary does not know the exact number of delegation tokens (A-DelTokensUnpredicted), the proxy can use one of them to deniably update her vote, which would be undetected by the adversary due to the presence of dummy delegation tokens cast by the posting trustee during the voting (A-DelBallotsUnpredicted). The proper anonymization during tallying is furthermore ensured under the assumptions (A-TabTellersHonest,A-CompRestricted). Hence, receipt-freeness for proxies is ensured under the same assumptions as in Sections 3.2 and 3.3 and the assumptions (A-DelTokensUnpredicted, A-DelBallotsUnpredicted).

Eligibility The proof submitted with direct ballots requires the knowledge of the voter's private signature key for submitting a non-dummy vote. Hence, given public signature keys of eligible voters the soundness of the proof ensures that the ballots that can influence the tally result can only be send by the voters possessing the

[11] Note that as we recognize the importance of formal security evaluation of e-voting protocols, we consider providing such an evaluation an important part of future work.

[12] Note, while the voter would be able to cast her own direct vote without the adversary knowing, she would be unable to delegate to a different proxy. We consider removing this restriction a part of future work.

corresponding private signature keys, which is ensured under the same assumptions as in Section 3.3.

Vote integrity For direct ballots, the proofs ensure, that the dummy ballots (i.e. the ballots not cast by the voter herself) only contain a dummy vote that does not influence the tally result. Following a similar argument, the dummy delegated ballots only contain dummy votes as well. The manipulation during tallying is furthermore prevented by the validity proofs attached to the mix net shuffle and to the PETs.

Availability As long as the contents of the bulletin board are available after the voting has finished, a threshold of tabulation tellers is required to conduct the tally. Hence, availability is ensured under the same assumptions as in Sections 3.2 and 3.3.

Participation privacy We first consider distinguishing whether a given voter has participated in the election by casting a direct ballot or abstained. For this, the adversary needs to be able to tell whether there are non-dummy ballots published next to the voter's id, which is prevented under assumptions in Section 3.3. Following a similar argument, the presence of a delegated ballot from the voter will be obscured by the dummy delegated ballots, cast by posting trustees, so that the adversary would be unable to tell whether a given voter has abstained or participated in the election by delegating (assumptions in Section 3.3 and (A-DelBallotsUnpredicted). Finally, anonymizing the final ballots before selecting either a direct or a delegated ballot for further tallying ensures, that the adversary would be unable to tell whether a voter cast a direct ballot or delegated to some proxy (A-TabTellersHonest, A-CompRestricted).

Delegation eligibility As long as the generators g, $\hat{g}$ are independent, the proofs π_d ensure that the knowledge of x_i is required to be able to cast a non-null delegated ballot with a delegation credential g^{x_i}. The proofs of validity used during tallying furthermore ensure, that the decryption reveals the credentials g^{x_i} used for casting the delegated ballot, either dummy or non-dummy. Hence, delegation eligibility is preserved under the same assumptions as in Section 3.2.

Delegation integrity In order to cast a delegated ballot on behalf of an non-authorized voter, a proxy would need to either find out the value of x_i, to fake the proof π_d attached to the delegation ballot or to manipulate the tally. As the delegation integrity of Section 3.2 relies on the same principles, delegation integrity in the scheme presented in this paper is ensured under the same assumptions.

Delegation privacy The delegation does not differ from [5] aside from a modified proof π_d, which, however, does not leak additional information about the identity of the voter. The tally procedure ensures proper anonymization by applying a mix net and distributed application of PETs, as long as the majority of the tabulation tellers is trustworthy. Hence, delegation privacy is ensured under the same assumptions as in Section 3.2.

Delegation power privacy As with delegation privacy, the modifications in the delegation and tally procedure does not reveal additional information as compared to the scheme in Section 3.2, hence, this requirement is preserved under the same assumptions.

6. Related Work

A number of cryptographic voting schemes with proxy voting functionality have been proposed [4, 2, 3, 22], which do not ensure any form of receipt-freeness for either voters or proxies, or otherwise protection against coercion. The scheme in [23] offers partial receipt-freeness, preventing the voters from creating receipts that show that they delegated to a specific proxy. The scheme, however, still allows both the voters and the proxies to prove how they voted. The proposal in [24] ensures coercion resistance (which includes receipt-freeness) in proxy voting by extending the well-known JCJ scheme [25]. It, however, inherits the complexity of JCJ, and therefore, its problems with practical applications such as lack of efficiency and usability for the voters [26].

7. Conclusion

We have presented an Internet voting scheme with proxy voting functionality by enabling the voters to delegate their vote to a trusted third entity, a proxy. Our proposal extends on the proxy voting scheme in [5] that modifies the well-established Helios scheme [6] towards proxy voting functionality. We further improve the privacy of [5] by introducing participation privacy and receipt-freeness. In our privacy improvements, we rely on the concept of dummy ballots introduced in an extension of Helios described in [7, 8]. Our proposal furthermore preserves the security requirements on both the general election and the delegation process specifically, as ensured in [5]. As we provided an informal security evaluation in this paper, future work

will consider formally proving its security. For this purpose, in particular, formal definitions of the security requirements have to be developed. We will also consider further efficiency improvements of our scheme, as well as evaluating and improving its usability.

Acknowledgements

This work has been supported by the German Federal Ministry of Education and Research within the Competence Center for Applied Security Technology (KASTEL).

References

[1] J. P. Gibson, R. Krimmer, V. Teague, and J. Pomares, "A review of e-voting: the past, present and future," *Annals of Telecommunications*, vol. 71, no. 7-8, pp. 279–286, 2016.

[2] B. Zwattendorfer, C. Hillebold, and P. Teufl, "Secure and privacy-preserving proxy voting system," in *ICEBE 2013: IEEE 10th International Conference on e-Business Engineering*, pp. 472–477, IEEE, Sept. 2013.

[3] A. Tchorbadjiiski, "Liquid democracy diploma thesis," *RWTH AACHEN University, Germany*, 2012.

[4] Y. Desmedt and P. Chaidos, "Applying divertibility to blind ballot copying in the helios internet voting system," in *ESORICS 2012: 17th European Symposium on Research in Computer Security*, pp. 433–450, Springer, Sept. 2012.

[5] O. Kulyk, K. Marky, S. Neumann, and M. Volkamer, "Introducing proxy voting to Helios," in *ARES 2016: 11th International Conference on Availability, Reliability and Security*, pp. 98–106, IEEE, Sept. 2016.

[6] B. Adida, "Helios: Web-based open-audit voting," in *SS 2008: 17th Conference on Security Symposium*, pp. 335–348, USENIX, July 2008.

[7] O. Kulyk, V. Teague, and M. Volkamer, "Extending Helios towards private eligibility verifiability," in *VoteID 2015: 5th International Conference on E-Voting and Identity*, pp. 57–73, Springer, Sept. 2015.

[8] D. Bernhard, O. Kulyk, and M. Volkamer, "Security proofs for participation privacy, receipt-freeness and ballot privacy for the helios voting scheme," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, p. 1, ACM, 2017.

[9] D. Bernhard, O. Kulyk, and M. Volkamer, "Security proofs for participation privacy, receipt-freeness, ballot privacy, and verifiability against malicious bulletin board for the helios voting scheme." Cryptology ePrint Archive, Report 2016/431, May 2016. `http://eprint.iacr.org/2016/431`.

[10] 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.

[11] T. P. Pedersen, "A threshold cryptosystem without a trusted party," in *EUROCRYPT 1991: 10th Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 522–526, Springer, Apr. 1991.

[12] C.-P. Schnorr, "Efficient signature generation by smart cards," *Journal of cryptology*, vol. 4, pp. 161–174, Aug. 1991.

[13] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *CRYPTO 1994: 14th Annual International Cryptology Conference on Advances in Cryptology*, pp. 174–187, Springer, Aug. 1994.

[14] J. Camenisch and M. Stadler, "Proof systems for general statements about discrete logarithms," tech. rep., Citeseer, 1997.

[15] D. Bernhard, O. Pereira, and B. Warinschi, "How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to Helios," in *ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 626–643, Springer, Dec. 2012.

[16] M. Jakobsson and A. Juels, "Mix and match: Secure function evaluation via ciphertexts," in *ASIACRYPT 2000: 6th International Conference on the Theory and Application of Cryptology and Information Security* (T. Okamoto, ed.), pp. 162–177, Springer, Dec. 2000.

[17] B. Terelius and D. Wikström, "Proofs of restricted shuffles," in *AFRICACRYPT 2010: 3rd International Conference on Cryptology in Africa*, pp. 100–113, Springer, May 2010.

[18] B. Adida, O. De Marneffe, O. Pereira, J.-J. Quisquater, and others, "Electing a university president using open-audit voting: Analysis of real-world use of Helios," *EVT/VOTE 2009: Electronic Voting Technology Workshop/Workshop on Trustworthy Elections*, vol. 9, pp. 10–10, 2009.

[19] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," *European transactions on Telecommunications*, vol. 8, no. 5, pp. 481–490, 1997.

[20] D. Chaum and T. P. Pedersen, "Wallet databases with observers," in *CRYPTO 1992: 11th Annual International Cryptology Conference on Advances in Cryptology*, pp. 89–105, Springer, Aug. 1992.

[21] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," in *EUROCRYPT 1998: 17th International Conference on the Theory and Application of Cryptographic Techniques*, pp. 591–606, Springer, June 1998.

[22] B. Zhang and H.-S. Zhou, "Digital liquid democracy: How to vote your delegation statement," 2017.

[23] O. Kulyk, S. Neumann, K. Marky, and M. Volkamer, "Enabling vote delegation in boardroom voting," in *Workshop on Advances in Secure Electronic Voting Associated with Financial Crypto 2017*, pp. 419–433, Springer, Apr. 2017. In press.

[24] O. Kulyk, S. Neumann, K. Marky, J. Budurushi, and M. Volkamer, "Coercion-resistant proxy voting," in *IFIP SEC 2016: 31st International Conference on ICT Systems Security and Privacy Protection*, pp. 3–16, Springer, June 2016.

[25] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *WPES 2005: 4th ACM workshop on Privacy in the electronic society*, pp. 61–70, ACM, Nov. 2005.

[26] S. Neumann and M. Volkamer, "Civitas and the real world: problems and solutions from a practical point of view," in *Availability, Reliability and Security (ARES), 2012 Seventh International Conference on*, pp. 180–185, IEEE, 2012.