Coercion-resistant Proxy Voting

Oksana Kulyk[a], Stephan Neumann[a], Karola Marky[a], Jurlind Budurushi[a], Melanie Volkamer[a,b]

[a] *Technische Universität Darmstadt, Darmstadt, Germany*
[b] *Karlstad University, Karlstad, Sweden*

1. Introduction

Democratic elections represent an important value in many modern states. This is not limited to governments, also companies and other organizations are constituted in democratic elections. In general, most democratic elections follow the principle of equal elections, meaning that one person's vote should be worth as much as another's, i.e. one man – one vote [16]. However, this principle often represents a challenge to voters, who are not sufficiently informed regarding the particular matter that is voted on. In order to address this issue, a new form of voting has been proposed, the so called *proxy voting*. In proxy voting, each eligible voter has the possibility to delegate her voting right to another eligible voter, so called *proxy*, that she considers a trusted expert on the matter. In this way, the proxy can use the received delegations to vote for her preferred voting option, so that the proxy's vote would count several times (i. e. as many times as the number of delegations she received, plus her own vote if the proxy votes as an eligible voter in the election herself) in the tally.

A few proxy voting implementations, provided by different organizations, already exist. Two widely known implementations are LiquidFeedback[1] and Adhocracy[2]. Further proxy voting proposals are the approaches proposed in [21] and [24].

However, all existing proxy voting proposals fail to address the issue of voter coercion: namely, the case when the adversary threatens the voter to vote in a particular way, or to abstain from voting. This issue has been commonly considered for non-proxy Internet voting, and a number of Internet voting schemes have been proposed, that address the problem of coercion, e.g. by providing coercion resistance [13] or coercion evidence [9]. In this paper, we build upon [13, 6] and an extension proposed by Spycher *et al.* [20] to propose a coercion resistant end-to-end verifiable Internet proxy voting scheme.

This paper is structured as follows: In section 2 we identify and derive security requirements that are relevant for proxy voting. Section 3 introduces the

[1] http://liquidfeedback.org/, last accessed January, 7, 2016.
[2] https://adhocracy.de/, last accessed January, 7, 2016.

fundamentals used for our proposal, which we present in section 4. In section 5 we evaluate the security of our proposal with respect to the requirements. Section 8 summarizes our contributions and provides directions for future research.

2. Requirements for Proxy Voting

We consider the following use case for proxy voting. The voter is not sure, which candidate or which voting option she wants to support in the election. However, she knows an expert, a public person or even a friend or a relative, whom she considers to be more informed on the election issue and whom she trusts to make the right decision. Hence, in this scenario the proxy can use the delegations she received from the voters to vote on her own discretion. On the other hand, the voters that know how they want to vote are supposed to vote directly without involving proxies. While conditional delegations, e. g. as the voter requesting from a proxy to choose between candidates A and B, or to vote for anyone but candidate C, are an interesting direction of future research on proxy voting, we consider such requests to be out of scope of our work.

Note that non-Internet proxy voting such as in Netherlands considers a different use case, whereby the voter might know how to vote, but delegate to a proxy due to being unable to physically get to the voting booth. Such a use case, however, would not be relevant for Internet voting, where physical presence at the polling station is not required for casting a vote.

The following **functional requirements** should be provided by a proxy voting system:

Delegation cancellation. If the voter for any reasons decides to vote herself, she should be able to cancel the delegation at any point of time before the tallying.

Delegation back-up. The voter can assign up to T priorities to her proxies. Only the vote from the proxy having highest priority will be included in the vote count. This functionality is useful if the voter wants to have a "back-up" delegation, in case her first choice of a proxy abstains from the election.

The **security requirements** in Internet voting have been thoroughly investigated in the literature, and both formal [8] and informal [18, 15] definitions have been proposed. In this work, we aim to address the following security requirements for the election.

Vote integrity. All votes cast by eligible voters should be included in the result.

Availability. It should be possible to compute election result even if some of the involved entities are faulty.

Vote secrecy for voters. The voting system should not reveal a link between the voter and the cast vote to anyone but the voter herself.
We aim at achieving the following security requirements for the delegation process:

Delegation integrity. Only the proxy having a valid permit from the voter should be able to cast a vote on this voter's behalf. The proxy should not be able to alter the priority given to them by the voter.

Delegation availability. A proxy should not be selectively prevented from having the votes delegated to her.

Vote secrecy for proxies. The voting system should not reveal a link between the proxy and her cast vote to anyone but the proxy herself.

Delegation privacy. The voting system should not reveal the link between the proxy and the identity of the voter who delegated to this proxy to anyone but the voter herself. Furthermore, it should not reveal whether a voter has delegated a vote or voted directly to anyone but the voter herself.

Delegation power privacy. The voting system should not reveal how many voters have delegated their vote to a specific proxy to anyone, including the proxy herself. Note that while the proxy can know her number of upcoming delegations, delegation power privacy is still ensured if the proxy cannot tell, how many of these delegations are valid, i. e. how many of her cast delegated votes will be included in the tally.
Note, that as we want to ensure coercion resistance, we require that vote secrecy for both voters and proxies, as well as delegation privacy and delegation power privacy, should be ensured also for the case when the adversary is capable of communicating with the voter or proxy. As such, we aim to prevent the following coercion scenarios:

Direct voter coercion. Similar to the definition in [13], we aim to ensure receipt-freeness (i. e. the voter should not be able to create a receipt that proves how she voted) and protect against forced abstention (i. e. coercing the voter to abstain from the election), simulation (i. e. the adversary voting instead of the voter) and randomization (i. e. coercing the voter to invalidate her vote by casting a randomly composed ballot). Preventing coercion for direct voters means, that vote secrecy is preserved also given the assumption, that the coercer can communicate with the voter.

Delegation coercion. We aim to prevent attacks, whereby the voter is coerced to delegate to a specific proxy. Preventing delegation coercion means, that delegation privacy and delegation power privacy are preserved also given the assumption that the coercer can communicate with the voter.

Proxy coercion. We aim to prevent attacks, whereby the proxy is coerced to forward her received delegations to an adversary or to use them to vote in a specific way. In particular, for the votes cast by proxies, the same coercion resistance as for the voters should be achieved (i. e. ensuring receipt-freeness, as well as protecting against forced abstention, simulation and randomization attacks). Preventing proxy coercion means, that vote secrecy for proxies and delegation

power privacy are preserved also given the assumption that the coercer can communicate with the proxy.

3. Background

In this section we introduce the fundamentals used to design our coercion-resistant verifiable proxy voting scheme.

3.1. Cryptographic Primitives

In the following we describe the cryptographic primitives our scheme relies on. Hereby, $\mathbb{G}_q$ denotes a cyclic multiplicative group with order q and $\mathbb{Z}_q$ denotes the cyclic additive group of integers modulo q.

Zero-knowledge proofs. In order to prove the correctness of statements within the voting scheme without revealing anything beyond the correctness *zero-know-ledge proofs* are employed. For this sake, techniques such as proving the knowledge of discrete logarithm [19] or discrete logarithm equality [5] are being used. These proofs can be made non-interactive by employing the strong version of the Fiat-Shamir heuristic suggested in [3]. An important extension of such proofs are *designated-verifier proofs* described in [12]. Given the verifier's public key, these proofs convince only the designated verifier about the correctness, rather than the general public.

Linear Encryption. In some parts of our scheme, we use a modified encryption scheme suggested in [4] (further denoted as *LE-ElGamal*). This scheme is semantically secure under the DLIN (decisional linear) assumption which is implied in groups where the DDH assumption holds. Namely, let $pk = (g_1, g_2, h) \in \mathbb{G}_q^3$ be the public keys of the encryption and $(x_1, x_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$ the private keys with $g_1^{x_1} = g_2^{x_2} = h$. If the keys are jointly generated by multiple parties with x_1, x_2 as threshold-shared secrets, then, according to [2] at least 2/3 of the parties have to be honest.

The message $m \in \mathbb{G}_q$ is encrypted as follows: two random values $(r_1, r_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$ are chosen and then the encryption - denoted as $\{\{m\}\}_{pk} \in \mathbb{G}_q^3$ - is calculated as $(c_1, c_2, c_3) = (g_1^{r_1}, g_2^{r_2}, m \cdot h^{r_1+r_2})$. The decryption then proceeds as $m = c_3 \cdot c_1^{-x_1} \cdot c_2^{-x_2}$. Ciphertexts can then be reencrypted by multiplying a ciphertext by an encryption of 1 using a random value $(r_1', r_2') \in \mathbb{Z}_q \times \mathbb{Z}_q$. Further operations used together with the ElGamal encryption, such as mix net shuffle, well-formedness proofs and plaintext equivalence tests can be adjusted for LE-ElGamal as well.

Plaintext Equivalence Tests. In order to check whether a given ciphertext c encrypts the same plaintext as another ciphertext c' without revealing any additional information about plaintexts, *plaintext-equivalence tests* [11] can be employed. This can be performed by the holders of an encryption secret key and consists of jointly decrypting the value $(c/c')^r$ given a secretly shared random value $r \in \mathbb{Z}_q$ which results in 1 in case the plaintexts are equal or in random value otherwise.

Proxy Reencryption. Let $\{m\}_{pk_1}$ be a ciphertext encrypting message m with ElGamal public key $pk_1 = (g_1, h_1)$. Given the knowledge of corresponding secret key $x_1 = \log_{g_1} h_1$ which can also be a shared secret between several participants the method described in [10] allows for computing a new ciphertext $\{m\}_{pk_2}$, that encrypts the same message using a different ElGamal public key pk_2.

Mix nets. Important components in electronic voting systems are *mix net schemes* which are used for anonymizing lists of ciphertexts $e_1, ..., e_N$. In addition to ensure the integrity of the shuffle a number of zero-knowledge proofs have been proposed in the literature. The most efficient schemes - up to now - are presented in [22] and [1]. A modification of such proofs can be used to mix tuples of ciphertexts $(\bar{e_1}, ..., \bar{e_N})$ with $\bar{e_i} = (e_{i,1}, ..., e_{i,k})$ while preserving the ordering within the tuple.

3.2. JCJ/Civitas Scheme

For our goal to provide a scheme for coercion resistant proxy voting we chose the JCJ/Civitas voting scheme proposed in [13] and then extended and implemented in [6] as basis. The coercion resistance of the scheme is based on the application of voting credentials. These credentials are used to authorize votes from eligible voters. In case a coercer demands the credential from a voter she can simply provide a fake credential which could not be distinguished from the real one by the coercer. We briefly outline the scheme below.

Setup and Registration. Prior to the election the election supervisor announces the different election authorities, namely the *registrar*, *registration tellers* and *tabulation tellers* and publishes their respective public keys on the bulletin board. The registrar publishes the electoral register which contains the identity, the public registration key, and the public designation key of each voter. Building upon a homomorphic cryptosystem the tabulation tellers generate an election key pair in a distributed manner and publish the respective public key pk on the bulletin board.

For each voter V_i each registration teller $j = 1, ..., N$ generates a credential share $c_{i,j}$ and publishes its encryption next to the respective voter's identity on the bulletin board from which the encryption of the voting credential $E_i = \{c_i\}_{pk} = \prod_{j=1}^{N} \{c_{i,j}\}_{pk}$ can be computed by multiplying all the individual credential shares. The shares $c_{i,j}$ in plaintext together with corresponding designated-verifier correctness proofs are then being forwarded to the voter. Now the voter can use them to compute their secret voting credential $c_i = \prod_{i=1}^{N} c_{i,j}$. Finally, the encrypted credentials E_i are being shuffled via mix net.

Voting. The voters use anonymous channels for vote casting. As her vote, the voter casts a tuple

$$\langle A_i = \{o\}_{pk}, C_i = \{c_i\}_{pk}, \sigma \rangle$$

with o as a chosen voting option and σ as well-formedness proof for A_i and proof of plaintext knowledge for C_i. The tuple is sent to one of the available *ballot*

boxes which stores the votes. In case the voter is forced to reveal her credential to a coercer she can give a fake credential c' instead while the coercer is not able to distinguish it from a real one.

Tallying. The tallying is jointly performed by the tabulation tellers. The votes with invalid proofs are excluded and the plaintext-equivalence tests are used for identifying the votes with duplicated credentials which are handled according to the rules concerning vote updating. The remaining tuples $\langle A_i, C_i \rangle$ are being shuffled and of votes are being anonymized with mix net shuffling. Afterwards, plaintext-equivalence tests are applied for checking the validity of the voting credential by each C_i with each authorized credential from the shuffled list E_i. For the votes with valid credentials the voting options A_i are being decrypted.

Security assumptions that JCJ/Civitas relies on:

1. The adversary is not capable of simulating the voters during the registration process. The adversary is also not present at the registration phase.
2. At least one registration teller is trusted and the communication channel between this teller and the voter is untappable[3].
3. The voting client is trusted.
4. At least k out of n tabulation tellers do not fail during decryption.
5. At least $n-k+1$ out of n tabulation tellers are trusted not to reveal their key shares.
6. The channels between voters and voting system used for vote casting are anonymous.
7. The DDH assumption and the RSA assumption hold and a random oracle is implemented via cryptographic hash function.
8. At least one ballot box to which the cast votes are submitted is correct.

In addition to the security requirements, it is assumed that the voters are capable of handling the credentials, e.g. by using some kind of secure credential management.

4. Proposed Proxy Voting Scheme

To tailor our JCJ/Civitas extension towards proxy voting we introduce a new kind of credentials, so called *delegation credentials*. In addition to a unique voter credential in JCJ/Civitas, each voter i obtains a list of prioritized *delegation credentials*. To delegate a vote with a certain priority j the voter selects the j-th credential from her list and forwards it to the intended proxy. Voters are allowed to forward different credentials with different priorities to different proxies. Throughout the tallying phase for each voter only the vote cast with the highest priority is counted. Due to the fact that delegation credentials are

[3]That is, the adversary is incapable of reading the messages sent over the channel.

generated on the same principles as the voting credentials in the original scheme the security of our extension also relies on the fact that the delegating credentials can be faked by the voter in case of coercion.

4.1. Necessary Modifications

We describe the modifications to the JCJ/Civitas scheme that are needed for implementing the delegation while ensuring the requirements listed in Section 2.

Ballot Clustering. Within the JCJ/Civitas scheme coercion-resistance is achieved by breaking the link between a voter's identity and votes cast in her name, both real and fake votes. The introduction of prioritized delegation credentials requires a relation between different credentials being maintained throughout the vote tallying phase. Retaining such a relation might however cause vulnerabilities with regard to coercion. To address these challenges we build upon proposals from scientific literature. Spycher *et al.* [20] present a JCJ extension towards linear tallying time. Therefore, the authors propose to assign identifiers to cast (real and fake) votes. During vote tallying after anonymization cast votes are only compared against the public credential assigned to their respective identifier. This reduces the tallying complexity from quadratic to linear regarding the number of cast votes. We build upon this approach: votes cast by the voter or delegated to different proxies share the same identifier such that within the set of votes sharing the same identifier the vote with the highest priority is tallied.

Delegation Server. Forwarding voting credentials to proxies results in a coercion vulnerability: The adversary might coerce a proxy to forward all received voting credentials. In order to test whether the proxy complies the adversary could anonymously delegate a credential to her and check whether this credential is being forwarded back to her. We address this problem by introducing a new entity - possibly implemented in a distributed way - that functions as *delegation server* (DS). The underlying idea is that proxies do not receive delegated credentials directly from the voter. Instead the voter blinds her credential and sends it to the DS (over an anonymous and untappable channel) which forwards an anonymization of the blinded credential to the proxy. The unblinding value is sent to the proxy over the private and anonymous channel. In this way, even if the coercer demands from a proxy to forward all credentials to her, the introduction of the delegation server, who is trusted for coercion resistance, allows the proxy to generate fake credentials instead. Since the blinded credential is anonymized, the fake credentials forwarded by the proxy to the adversary are indistinguishable from the credentials submitted by the voters.

Inclusion of Linear Encryption. To prevent unauthorized usage of voting credentials the JCJ/Civitas scheme forces the voter to include the plaintext knowledge proof for the ciphertext encrypting the credential. This solution, however, is inapplicable to the delegation process since the proxy does not get to know the value of the credential as outlined above. We address this challenge by publishing voting credentials (in the registration and voting phases)

encrypted with linear encryption rather than ElGamal encryption. On the other hand for delegating her vote the voter encrypts their delegating credential with standard ElGamal using (g_2, h) as an encryption key. The resulting tuple $\{c\}_{pk} = (a = g_2^r, b = ch^r)$ together with other necessary information (see Section 3.2) is being forwarded to the proxy. If the proxy wants to cast the vote she chooses a random value of s and computes (g_1^s, a, bh^s) which is an LE-ElGamal encryption of c with randomness values r, s for which the proxy also can prove the knowledge of s as $\log_{g_1} g_1^s$.

4.2. Scheme Description

In this section we provide a detailed description of our proposed scheme. The scheme involves following entities:

- *Registration tellers* are responsible for generating and distributing the credentials to the voters. At least one of the registration tellers is trusted not to reveal private information to the adversary and to establish an untappable channel to the voters.
- *Delegation server* is responsible for forwarding the delegation credentials from the voters to the proxies. The server is trusted not to reveal its private information to the adversary.
- *Bulletin board* is responsible for publishing and storing all the public data in an election. It is trusted to act as a reliable append-only broadcast channel, i.e. not to remove or modify its contents once they are published and show the same view to everyone.
- *Tabulation tellers* are responsible for performing the tallying after all the ballots have been cast. More than $\frac{2}{3}$ of them is trusted not to reveal their private information to each other or to the adversary, and at least $\frac{1}{3}$ of them is trusted to provide valid output during the election and

Preliminary Stage. During this stage the keys used for encrypting votes and/or credentials are generated. The DS generates ElGamal keys with $pk_D = (g_D, h_D)$ as public key. The tabulation tellers distributively generate LE-ElGamal keys $pk_T = (g_1, g_2, h_T)$. We further denote $\{m\}_{pk_T}$ as ElGamal encryption with (g_2, h_T) as corresponding key and $\{\{m\}\}_{pk_T}$ as LE-ElGamal encryption.

The list of proxies $D_1, ..., D_d$ is being made public together with their public keys used for signing and designated-verifier proofs. For the sake of simplicity we assume that each proxy is eligible to vote herself as well. Furthermore, anonymous channels that enable communication between the proxies and the delegation servers as well as between proxies and the rest of the voters are established.

ID	Priority	Credential shares
id_1	1	$\{\{c_1^{1,1}\}\}_{pk_T}, \ldots, \{\{c_1^{1,N}\}\}_{pk_T}$
$\vdots$	$\vdots$	$\vdots$
id_1	T	$\{\{c_1^{T,1}\}\}_{pk_T}, \ldots, \{\{c_1^{T,N}\}\}_{pk_T}$
$\vdots$	$\vdots$	$\vdots$
id_k	1	$\{\{c_k^{1,1}\}\}_{pk_T}, \ldots, \{\{c_k^{1,N}\}\}_{pk_T}$
$\vdots$	$\vdots$	$\vdots$
id_k	T	$\{\{c_k^{T,1}\}\}_{pk_T}, \ldots, \{\{c_k^{T,N}\}\}_{pk_T}$

Table 1: Content of the bulletin board after the setup phase of the extended scheme.

Setup Phase. Opposed to the standard JCJ/Civitas scheme, each registration teller generates T credential shares at random for each voter. Analogously to the JCJ/Civitas scheme, the encrypted credentials are publicly assigned to the respective voters whereby the order of the credential shares is of central importance to the delegation process. A public identifier, e.g. the position of the respective voter in the electoral roll is assigned to each voter. After the setup phase the bulletin board contains T credentials for every voter $V_1, ..., V_k$ (see Table 1) as well as individual credential shares from each of N registration tellers for each priority $1, ..., T$. We consider the lower number to denote the higher priority.

Registration. The registration phase remains identical to the standard JCJ/ Civitas scheme except the fact that each registration teller releases T ordered credential shares to the voter. The voter can then verify whether the received shares from the tellers for a credential $c_i^{(j)}$ correspond to the encrypted shares published on the bulletin board near id_i and priority j.

Before the voting, the voter merges her $N \cdot T$ credential shares as follows:

$$\begin{aligned} c_i^{(1)} &= c_i^{1,1} \cdot c_i^{1,2} \cdot \ldots \cdot c_i^{1,N} \\ &\vdots \quad\quad \vdots \\ c_i^{(T)} &= c_i^{T,1} \cdot c_i^{T,2} \cdot \ldots \cdot c_i^{T,N} \end{aligned}$$

Voting. To cast a vote (without considering delegation) for voting option o voter i prepares the following tuple:

$$\langle \{\{id_i\}\}_{pk_T}, \{\{c_i^{(j)}\}\}_{pk_T}, \{\{o\}\}_{pk_T}, \sigma \rangle$$

Here σ signifies both the well-formedness proofs for o as well as proof of randomness knowledge for $\{\{c_i^{(j)}\}\}_{pk_T}$: namely, given $\{\{c_i^{(j)}\}\}_{pk_T} = (c_1, c_2, c_3) = (g_1^{r_1}, g_2^{r_2}, c_i^{(j)} h^{r_1+r_2})$ the voter proves the knowledge of randomness r_1 as $\log_{g_1} c_1$.

Note that due to the zero-knowledge property of σ, it does not leak any information about the plaintext $c_i^{(j)}$. The value of j is chosen depending on the voter's delegations where we distinguish the following cases:

1. If the voter does not intend to delegate her vote at a later point in time she sets $j = 1$.
2. If the voter might intend to delegate her vote at a later point in time she sets j as the lowest available priority: that is $j = T$ in case she did not delegate any vote yet or $j = j_d - 1$ if j_d is the highest priority that was already delegated.

Additionally, the voter i casts her identifier id_i in an encrypted manner which later serves for clustering ballots from the same voter with different credentials.

Delegating. To delegate her vote with priority $j = 2, \ldots, T$ to the proxy D_k the following protocol (see Figure 1) is executed.

1. The voter i chooses a random value x and sends the following tuple to one or more of the DS:

$$\langle \{\{id_i\}\}_{pk_T}, \{(c_i^{(j)})^x\}_{pk_D}, \sigma, id_{D_k} \rangle$$

   Here $c_i^{(j)}$ is the j-th credential from her credential list $(c_i^{(1)}, \ldots, c_i^{(T)})$, id_i is the voter's index, σ is the proof of plaintext knowledge for $\{(c_i^{(j)})^x\}_{pk_D}$[4] and id_{D_k} is the identifier, e.g. the public key, of the chosen proxy. The voter also sends $x, \{\{id_i\}\}_{pk_T}$ to D_k over a private channel.
2. The DS computes $\{(c_i^{(j)})^x\}_{pk_T}$ from $\{(c_i^{(j)})^x\}_{pk_D}$ using proxy reencryption scheme and a designated-verifier proof using the public designated-verifier key of D_k that both $\{(c_i^{(j)})^x\}_{pk_T}$ and $\{(c_i^{(j)})^x\}_{pk_D}$ encrypt the same plaintext. The proof and the values of $\{(c_i^{(j)})^x\}_{pk_T}$, $\{(c_i^{(j)})^x\}_{pk_D}$ together with the voter's index $\{\{id_i\}\}_{pk_T}$ are being forwarded to D_k.
3. The proxy D_k verifies the proof and sends the signed value of $\{(c_i^{(j)})^x\}_{pk_D}$ back to the voter as confirmation.[5]
   She further computes $\{c_i^{(j)}\}_{pk_T}$ as $\{(c_i^{(j)})^x\}_{pk_T}^{1/x}$.

Casting a delegated vote. To cast a delegated vote for a (unknown) voter X with (unknown) priority Y the proxy first calculates $\{\{c_X^{(Y)}\}\}_{pk_T}$. For this - given an encryption $\{c_X^{(Y)}\}_{pk_T} = (c_1, c_2)$ - she chooses a random value s and computes $\{\{c_X^{(Y)}\}\}_{pk_T} = (g_1^s, c_1, c_2 h^s)$. Then she encrypts her voting option o and prepares her ballot according to the voting process outlined above.

[4] Same as for voting, due to its zero-knowledge property σ does not leak any information about the plaintext $(c_i^{(j)})^x$.

[5] This can be done via a two-way anonymous channel or by publishing the signature on $\{(c_i^{(j)})^x\}_{pk_D}$ on the bulletin board.

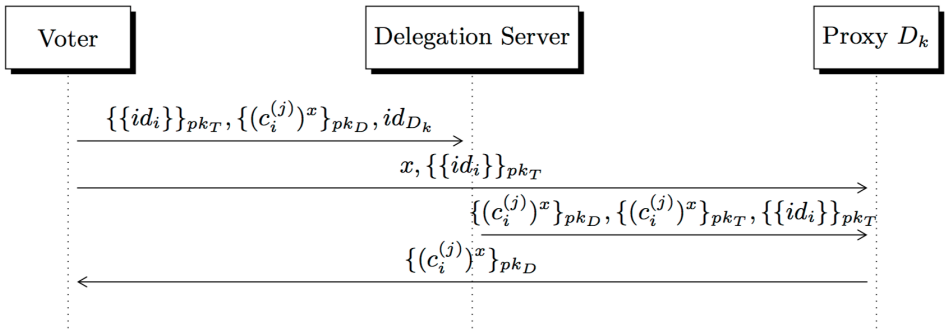

Figure 1: Delegation of the voter id_i to the proxy D_k, with zero-knowledge proofs omitted.

Cancelling a delegation. If the voter intends to withdraw one or several vote delegations (but not excluding delegation in general), she issues the highest prioritized unused credential, overriding all previously cast votes on her behalf.

Obfuscating the number of cast votes. The fake votes intended to hide the number of votes cast by a specific voter or her proxy are added accordingly to Spycher *et al.*. For each identifier id_i the tabulation tellers cast a random number of votes of the following form:

$$\langle \{\{id_i\}\}_{pk_T}, \{\{r\}\}_{pk_T}, \{\{o\}\}_{pk_T}, \sigma \rangle$$

Here r denotes a fake credential randomly drawn for each fake vote, a random valid voting option o, and the respective zero knowledge proofs σ. The number of fake votes for each voter is secret and is distributed as outlined in [20]. Note, as the result of the obfuscation, the tabulation tellers know which ballots originated from them and which ones were cast by the voters, but they do not know the identities of the voters who cast their ballots. At the same time, since each tabulation teller casts her obfuscating ballots independently, a tabulation teller knows which ballots were cast by other tabulation tellers, but she does not know, on behalf of which voter these ballots were cast (i.e. which *id* was used).

First anonymization. After all the votes have been cast, votes with invalid proofs are removed and excluded from further tallying. The remaining ballots are further processed by the tabulation tellers in order to remove duplicates (i.e. the ballots cast using the same credentials) as described in [23], according to the election policies on vote updating. [6]Thereafter, tuples of the form $\langle \{\{id\}\}_{pk_T}, \{\{c\}\}_{pk_T}, \{\{o\}\}_{pk_T} \rangle$ are anonymized by the application of a mix net, whereby each tabulation teller acts as a mix node. The purpose of the first

[6]Note that this step can be omitted if the election does not allow vote updating – in that case, from the ballots cast with the same valid credential, a random one will be included in the tally.

anonymization step is to remove the link between the voter *id* and the cast ballot, so that neither the tabulation tellers nor the general public knows which ballots were submitted by the voters themselves, and which were cast by the tabulation tellers in the obfuscation stage.

Ballot clustering. After anonymizing the tuples the values of *id* are jointly decrypted by the tabulation tellers. For any index id_i appearing within at least one anonymized tuple the respective ordered list of credentials $(\{\{c_1^{(1)}\}\}_{pk_T}, \ldots, \{\{c_1^{(T)}\}\}_{pk_T})$ is obtained from the bulletin board. All tuples sharing the same id_i are clustered. The ordered list of credentials is attached to each cluster, and the value id_i is removed from all tuples.

Second anonymization. All lists of resulting tuples together with the respective list of attached credentials are anonymized again. The purpose of the second anonymization is to remove the link between the voter *id* and the used credentials, so that neither the tabulation tellers nor the general public can tell, whether a ballot on behalf of a given voter (cast either by the voter herself or by the tabulation tellers in the obfuscation stage) was cast with a valid or invalid credential, or whether and with which priority a given voter delegated to some proxy. The anonymization proceeds as follows. First, a second set of obfuscating ballots is cast by the tabulation tellers for the purpose of preventing information leakage from the cluster size. Namely, given X as the size of the largest cluster, a corresponding amount of the tuples $\langle\{\{r\}\}_{pk_T}, \{\{o\}\}_{pk_T}, \sigma\rangle$ are added to each cluster by the tabulation tellers as in the previous obfuscation, so that each resulting cluster has exactly X ballots. Afterwards, a mix net is being applied with each tabulation teller acting as a mix node, so that the lists are being reencrypted and permuted. The shuffle results and the corresponding proofs of shuffle validity (provided by the tabulation tellers) are published on the bulletin board. Note that the order of ciphertexts within the tuples is preserved so that the priority order of delegation credentials remains intact.

Extracting votes to be counted. In addition to weeding out the votes with invalid credentials our goal is to tally only the vote with the highest priority in case delegation took place. In order to do this for any element of the cluster - namely a list of tuples and public credentials - the list of tuples is matched on the basis of plaintext equivalence tests (PET), in decreasing order against the list of public credentials. Starting with the highest priority credential c_1 PETs are jointly executed by the tabulation tellers between all credentials of submitted tuples until a match is found. The results of PETs and corresponding validity proofs are published on the bulletin board. If a match is found the value $\{\{o\}\}_{pk_T}$ is extracted from the matching tuple. Once the process has been terminated for all tuples a list of encrypted voting options $(\{\{o_1\}\}_{pk_T}, \ldots, \{\{o_n\}\}_{pk_T})$ has been extracted.

Third anonymization and result calculation. The list of encrypted voting options is anonymized by the application of a mix net with each tabulation teller

acting as a mix node. The purpose of this anonymization is to remove a link between the vote and the delegation priority, so that neither the general public nor the tabulation tellers know, whether a given vote was cast directly by some voter or by some proxy, and with which delegation priority it was cast. Eventually, the anonymized encrypted voting options are jointly decrypted by the tabulation tellers, and the resulting anonymized list together with the proofs of shuffle validity is published on the bulletin board.

5. Security of the Proposed Scheme

In this section we consider the security evaluation of our scheme. We first summarize the security assumptions that need to be made, and then provide an informal security argument regarding the requirements given in Section 2.

5.1. Security Assumptions

We summarize the security assumptions specific to our scheme in the list below:

1. More than 2/3 of all tabulation tellers are trusted not to disclose private key shares to the adversary.
2. At least 1/3 of all tabulation tellers does not fail during decryption.
3. At least one of DS does not fail to forward the delegated votes to the proxy.
4. The DS does not disclose private information to the adversary.
5. The public key infrastructure for the proxies is trustworthy.
6. The private signing and designated-verifier keys of the proxy is not leaked.
7. Communication channels between the voters and the proxies are private.

5.2. Security Evaluation

We hold on to the assumptions of the original scheme which we provide in Section 3.2. The security properties of our scheme and the additional assumptions that are needed for them can then be evaluated as follows:

Vote integrity. The assumptions on integrity for voters remain the same as in the original scheme.

Availability. The election results can be computed under the assumption that at least $k = \frac{1}{3}n$ tabulation tellers participate in the decryption.

Vote secrecy for voters. The system provides probabilistic coercion resistance, as there are two scenarios where the coercer can tell whether the voter has obeyed. The first scenario occurs if the number of fake votes for some voter equals the known minimal number. The probability of this can be controlled with the choice of an appropriate distribution function for fake votes. In the second attack, the coercer requests all the delegation credentials from the voter, and casts a vote with priority j that is unknown to the voter. In that case, unless there is a vote cast with the same priority from another voter, the coercer knows whether the credential given to her was real. The success probability of such an attack can be reduced with a smaller value of T. For example, with $T = 3$ the voter can either vote herself, delegate once or choose one back-up proxy, which we assume to be sufficient functionality for most of the elections.

Outside of these scenarios, the system provides vote secrecy, and with it coercion resistance for the voters under the same assumptions as the original scheme with $k \leq \frac{1}{3}n$. Since the delegating credentials are generated and distributed in the same way as the voting credentials the voter can cheat the coercer who acts as a proxy by providing a fake credential instead. Even if the coercer is watching the voter directly during the delegation, the voter can input a random value as her delegating credential so that the coercer cannot distinguish it from the real one.

Delegation integrity. Casting a delegated vote without voter's permission or a delegated vote with a higher than given priority would require the knowledge of the corresponding credential $c_i^{(j)}$ for the voter i and priority j. Given that each one of those credentials is generated in the same way as the voter credentials in the original scheme it holds that they are not leaked under the same assumptions. Another way to break the delegation integrity would be to intercept the value x used for blinding the credential forwarded to the proxy which requires control over the communication channel between voter and proxy. Furthermore, it must be assumed that the public key infrastructure for the proxies is trustworthy so that impersonation of a proxy to a voter is infeasible.

Delegation availability. The proxy receives the credential from the DS accompanied by the designated verifier proof. In case no credential is sent the voter gets no confirmation and thus is able to repeat the delegation by choosing another DS. For this it must be assumed that the proxy's private key is not leaked and the confirmation cannot be sent by someone else. However, the DS is capable of submitting an invalid credential if it can fake the designated verifier proof. Therefore, it must be assumed that the private designated-verifier key of the proxy is not being leaked.

Vote secrecy for proxies. Given an anonymous channel between the proxies and the bulletin board the secrecy of votes cast by proxies corresponds to the vote secrecy for the voters. An additional assumption is required that the DS is not collaborating with the adversary. In this case using a designated-verifier proof the proxy can fake the credentials resulting from the delegation process and present them to the coercer if forced to do so.

Delegation privacy. The proxy could be capable of learning the identity of the delegating voter in following ways: 1) by identifying the message's origin via network, 2) by learning the voter's ID, 3) by being able to assign the given delegating credential to the voter's identity. The communication channels between voters and proxies are anonymous, the voter ID is sent encrypted and only decrypted after anonymization. The delegating credentials are not otherwise leaked which is similar to the voter's credentials in the original scheme. This implies that the proxy is incapable of determining the identity of voters delegating to her. In case of a coerced proxy the coercer would not have a possibility of knowing whether the credential passed to them is valid and whether the voter has cast another vote herself. This corresponds with the coercion-resistance properties of the original scheme.

Delegation power privacy. Given the anonymous communication channels between the DS and the voters a proxy cannot prove for given credentials that they come from actual voters and are not simulated by the proxy herself. Moreover, due to the coercion resistance properties of the original scheme and its credential generation process the proxy herself cannot verify whether the credential she received is valid. This implies that the proxy is incapable of constructing a convincing proof of possessing any amount of delegated votes.

6. Discussion

One bottleneck in the efficiency of our scheme is the addition of fake ballots during the second anonymization: by casting X ballots on behalf of some voter with X high enough, the adversary can slow down the tallying stage by the factor of $\mathcal{O}(XN)$ with N as the number of eligible voters. Such a loss of efficiency can be a serious drawback to our scheme, which we plan to address in future work. The step of adding fake ballots, however, is crucial to prevent the information leakage from the size of ballot clusters: otherwise, for large enough X, it is likely that the cluster with at least X ballots will be the only one in the election, so that the adversary would be able to identify the cluster of the coerced voter even after the second anonymization, which could violate the coercion resistance of the scheme. Namely, an adversary can conduct an attack similar to a 1009-attack on the original JCJ described in [23] and cast X ballots using the same credential provided by the voter. Later, given $S > X$ as the size of the coerced voter's cluster, if no match is found after $S - X + 1$ comparisons during the extraction of the votes to be counted, the adversary knows that she was provided with an invalid credential[7]. In this section we discuss the ways to minimize the impact of this vulnerability.

Note that the voter can still provide a fake credential for a requested priority to the adversary and cast her own vote with a real credential for the same priority herself. In order to make sure, that the adversary is not able to tell, which

[7]We thank the anonymous reviewer for pointing out this attack to us.

credential has been used, following changes have to be made for the protocol. Before comparing the credentials used for casting the ballots $\{\{c\}\}_{pk_T}$ with the public credentials, the duplicate credentials (i.e. $\{\{c\}\}_{pk_T}$ that encrypt the same plaintext) should be removed. For this purpose, the method described in [23] can be used. As a result, only one ballot from the adversary using a fake credential would be kept in the cluster after second anonymization. Hence, if a voter has cast her ballot using the same credential, the adversary would not able to tell which ballot has been counted.

Note, however, that being able to identify the cluster of the coerced voter implies that the adversary would know, with which priority the voter has delegated her vote, or whether the voter abstained from the election. Hence, forced abstention attacks would be possible. Furthermore, the following attack would violate coercion resistance, enabling the adversary to tell with non-negligible probability whether the voter providing her the credentials has cheated. Let T be the total number of delegation priorities. The coercer demands that the voter sends her all T of her delegation credentials $c_i^{(j)}$. She then chooses a random number $P \in \{1, ..., T\}$ and votes using the credential $c_i^{(P)}$. As the coercer is able to identify the cluster of a coerced voter at the time when the votes with invalid credentials are being removed, she would be able to tell whether a valid vote with priority P was cast by the voter. While the voter can attempt to cheat by sending fake credentials to the adversary and voting with a valid credential $c_i^{(P')}$ with $P' \in \{1, ..., T\}$ she attempts to guess. If the voter guesses the priority correctly (i.e. $P = P'$), then the coercer would not be able to tell whether the voter has cheated; otherwise, however, the coercer will notice the absence of a vote with priority P for a coerced voter. Hence, the probability for the voter to cheat the coercer without being detected is $1/T$, which, especially for large values of P, can be unacceptably small. As this attack would not be possible if the adversary cannot distinguish the cluster of a coerced voter (which should be achieved with the help of fake ballots), a trade-off between efficiency and security has to be taken when considering whether the step of adding these fake ballots should be included in the scheme.

7. Comparison With Other Proxy Voting Schemes

In this section we compare our proposal with other schemes that ensure proxy voting with cryptographic methods, namely [14, 24, 21]. We first provide a high-level description of these proposals, followed by a brief discussion of their security and functionality assurances, as compared to the scheme provided in this chapter.

The table that summarizes the differences of each system is provided in Table 2. The differences conclude such characteristics as: 1) whether a scheme provides coercion resistance, 2) whether it preserves the vote secrecy for proxies, 3) whether it hides the fact, whether a given voter has delegated her vote, cast a direct vote or abstained, 4) whether it preserves delegation power privacy either before or after the tally (i.e. whether a proxy is capable of proving to a third

party, how many delegations she has received), 5) whether the schemes allows to verify, that only the votes from eligible voters are included in the tally, 6) whether the scheme implements trust distribution (i.e. so that no single entity is capable to violate a security requirement such as vote secrecy or integrity), 7) whether the voters are allowed to choose a back-up proxy for their delegation, 8) whether the proxies are allowed to update their delegated votes, and 9) whether the proxies are allowed to re-delegate their delegated votes to other proxies, while retaining the right to cancel the re-delegation.

Table 2: Comparison of proxy voting schemes

	This paper	[14]	[24] (client-side)	[24] (server-side)	[21]
Coercion resistance	Yes	No	No	No	No
Vote secrecy for proxies	Yes	Yes	No	No	Yes
Delegation privacy (whether the voter delegated)	Yes	No	Yes	Yes	Yes
Delegation power privacy (before the tally)	Yes	Yes	Yes	Yes	No
Delegation power privacy (after the tally)	Yes	Yes	Yes	No	No
Eligibility verifiability	Yes	Yes	No	No	No
Trust distribution	Yes	Yes	No	No	Yes
Delegation back-up	Yes	Yes	No	No	Yes
Transitivity	No	No	No	No	Yes
Proxy re-voting	Yes	Yes	No	Yes	Yes

7.1. Introducing Proxy Voting to Helios by Kulyk et al. [14]

Another scheme for proxy voting functionality in Internet voting has been suggested by Kulyk et al. in [14]. The proposal extends a well-known and widely used Helios Internet voting system (namely, building upon its variant that uses digital signatures for voter authentication, as proposed by Cortier et al. [7]) by enabling the voters to delegate their votes and cast delegated votes as proxies. For this purpose, similar to the scheme described in this paper, the voters get the so-called *delegation credentials* consisting of a public and a secret key, one for each delegation priority, and use them to construct *delegation tokens* that are used for the delegation.

Thereby, the delegation token consists of two parts. The private part is a random value m that is chosen by the voter and privately sent to the proxy. The public part, anonymously published by the voter on the bulletin board, consists of an encrypted value of the public delegation credential c, the zero-knowledge proof of knowledge of the secret key for this delegation credential π_d, the value $\sigma = g^m$ for a public generator g and a proof/signature of knowledge π_d which

proves in zero-knowledge the knowledge of the secret key for the used delegation credential, while binding the proof to the value of σ. For casting the delegated vote, the proxy then sends her encrypted vote on behalf of the voter together with the proof of knowledge of m given σ.

Casting a direct vote occurs in the same way as in the original Helios system. After the period for casting of both direct and delegated votes has ended, the delegated votes go through the weeding process as follows: first, they are processed with a mix net for the sake of anonymisation. Afterwards, the encrypted delegation credentials belonging to each token are being decrypted in order to discard the delegated votes that either were cast using invalid delegation credentials (i.e. not the ones belonging to eligible voters), using the delegation credential from the voter who also cast a direct vote thus cancelling the delegation, or using the delegation credential ovewritten with another delegation from the same voter with a higher priority. The remaining votes are being combined with the direct votes, and further anonymised and decrypted as in the original Helios system.

Security and Functionality. The scheme aims to preserve the security properties of the Helios system for the direct voters. Hence, while it ensures vote secrecy and integrity under the certain trust assumptions (also present in Helios), it does not ensure coercion resistance, as both the voters and the proxies can prove casting a vote for a specific candidate (e.g. by storing the randomness used for encrypting the vote), and the voters can prove delegating a vote to a particular proxy (controlled by the adversary). Hence, similar to Helios, the extension towards proxy voting is suitable to be used in low-coercion environment only.

The same mechanisms that are used in Helios to ensure vote secrecy and integrity for the direct votes, also ensure vote secrecy and integrity for proxies. The encrypted delegation credential used in delegation tokens ensure, that the link between the voter and the proxy remains secret, thus preserving delegation privacy. It furthermore aims to preserve delegation power privacy, in that a proxy is incapable to find out, whether a given delegation token contains an encryption of a valid delegation credential, and thus can be used for casting a valid vote. However, the scheme reveals, whether a given voter has cast a direct vote, delegated her vote (and with which priority) or abstained from the election.

As Helios employs mechanisms for ensuring end-to-end verifiability, so does the extended scheme, when it comes to direct voting. In a similar vein, regarding the delegation process, the voters have the possibility to verify, that their public delegation tokens are properly stored on the bulletin board and the votes cast with these delegation tokens are properly included in the tally. The proxies, similar to the voters casting direct votes, are capable of verifying that their votes have been cast as intended, stored as cast and tallied as stored.

The functionality provided in the scheme is similar to the functionality of the scheme described in the present paper. The voters are able to cancel their delegation by casting a direct vote at any time prior to the tallying, and assign different priorities for delegating their vote to different proxies for delegation

back-up.

7.2. Secure and Privacy-Preserving Proxy Voting System by Zwattendorfer. et al. [24]

The authors of the scheme proposed in [24] take two approaches to delegating. In the first variant, which they denote as *client-side*, the voters delegate by encrypting via a non-determenistic encryption scheme and casting the name of their chosen proxy as their vote. Hence, after the tallying, the vote of each proxy counts as many times as there are votes encrypting her name. For cancelling the delegation, the voter casts another vote that either encrypts the name of another proxy (thus re-delegating the vote) or the voting option (thus casting a direct vote).

In the second approach, the so-called *server-side* delegation, the proxies cast and publish their votes prior to the election. Then, in order to delegate her vote, the voter simply encrypts the vote, cast by her chosen proxy.

The anonymity for the voters in both approaches is ensured via blind signatures. In order to cast her ballot, representing either a direct vote, a name of the chosen proxy (in server-side approach) or a copied vote of a chosen proxy (in client-side approach), the voter encrypts her choice, authenticates herself to the *ballot signer*, who checks whether a given voter has already voted in the election, and if not, returns a signed ballot to the voter. Together with the signed ballot, the voter also gets a so-called rejection key, which is also signed by the ballot signer.

The ballot, signed by the ballot signer, is then being anonymously sent by the voter to the voting server. For revoking the vote, the voter computes another ballot which is sent to the ballot signer with the rejection code. At the end of the elections, the private key is published, so that everyone can decrypt the cast ballots and tally the election result.

Security and Functionality. The system provides verifiability for the voters, in that they are able to record their cast ballots, and then check whether theses ballots are stored on the voting server and decrypted correctly. The eligibility of the election, however, could be violated by the ballot signer alone: given that it is the only entity responsible for authenticating the voters, the ballot signer is capable of adding the ballots that do not belong to eligible voters to the tally without any means to detect it. Furthermore, the ballot signer is capable of violating vote secrecy, as it knows the link between each cast ballot and the voter's identity. The system is also vulnerable to coercion and vote or delegation selling, as the voter can prove casting a vote for a particular candidate or delegating it to a particular proxy. Namely, let $\mathsf{Enc}(v, r)$ denote an encryption of a voting option v (which is either a direct vote or a name of a chosen proxy) with the randomness value r. As all the cast encrypted votes are published, if the voter sends v and r to the adversary, then the adversary can verify that $\mathsf{Enc}(v, r)$ is published and included in the tally, hence, the voter has cast a ballot for the voting option v. Moreover, as all the ballots are simply decrypted afterwards without any anonymization, the voter can send $\mathsf{Enc}(v, r)$

to the adversary before casting it as her ballot, and the adversary just has to verify that the ballot appears on the bulletin board later on, and then verify that it decrypts to v after the tally.

Regarding the security of the delegation process, the authors do not aim to protect the vote secrecy for proxies, claiming that it is crucial for the proxies' votes to be public for ensuring better transparency. In the server-side approach, as the tally result reveals, how many delegations each proxy has got, it does not ensure delegation power privacy. The client-side approach, while hiding the number of delegations to each proxy, as well as the total number of delegated votes, also provides some limitations on the functionality, as the proxy is not able to update her vote after the delegation.

The scheme also does not support delegation back-up by not allowing the voter to chose several proxies for delegating a vote. This, however, is justified in a setting, where all the proxies are required to cast their ballot prior to the election.

7.3. Liquid Democracy by Tchorbadjiiski [21]

A proposal for proxy voting by Tchorbadjiiski [21] takes yet another approach on how to implement the delegation functionality by using hash chains. The credentials that are issued to the voters consist of public and private parts and are constructed in the following way. The private part h_S is chosen by a voter at random and kept secret, while the public part h_A is calculated by computing a hash chain using the hash function $\mathcal{H}$ on h_S with depth T for T delegation priorities (i. e. for $i = 1, ..., T+1$, it holds that $h_0 = h_S$, $h_i = \mathcal{H}(h_{i-1})$, $h_A = h_{T+1}$). In this way, given a credential h_i, it is possible to verify that it belongs to a hash chain defined by the public value of h_A by computing $\mathcal{H}(...(\mathcal{H}(h_i)...)$ up to T times. Furthermore, for two values h_i, h_j from the same hash chain, it is possible in a similar way to determine, which value has higher priority (i.e. whether $i > j$).

In order to register their credentials, the voters authenticate themselves to the *voting register* (VR) and obtain a blind signature on h_A. For delegating their vote with priority $i = 1, ..., T$, the voters send the value h_i from the hash chain, calculated as described above, together with the value h_A signed by the VR, to the proxy of their choice. Note, that this scheme also allows for transitive delegation: given the value h_i, the proxy is able to compute the rest of the values $h_{i+1}, ..., h_T$ from the hash chain and use them to re-delegate the vote to other proxies.

For casting either a direct or delegated vote, the voter or the proxy anonymously sends the h_A signed by the VR, the blinded value of the suitable voting credential (i.e. h_S for direct votes, h_i for delegated votes) and the blinded value of her chosen voting option to the VC. The VC, after verifying the signature on h_A, signs the blinded values and returns them to the voter. The voter then unblinds the vote and the credential and sends them back to the VC, thus finalising the casting of her vote.

After the vote casting is finished, each public credential h_S can be assigned a vote that was cast with the highest priority in the hash chain characterised

by h_S. As the votes are published in plaintext, after discarding the votes overwritten by votes cast with highest priorities, the rest of the votes can be tallied.

Security and Functionality. The scheme ensures vote secrecy for both voters and delegates with the combination of blind signatures and anonymous channel used for vote casting. The votes are published in plaintext, with each vote assigned to a unique nonce that can be used by the voter to verify, that her vote has been stored correctly. However, unless the voter verifies her vote, it can be changed to an arbitrary value by either the VC (via replacing the vote) or by VR (by using the voter's credential to cast a vote herself). In the same vein, if the voter registers herself for the election but chooses to abstain from casting her vote, unless she verifies, a malicious VR alone can cast a vote on her behalf.

Similar to other schemes described in this chapter, the scheme in [21] does not provide any coercion resistance. As such, the scheme assigns unique nonces to each vote that the voter has to store until the result is published. In case a voter provides such a nonce to the adversary, the adversary can check herself how the voter has voted. Furthermore, in case the voter is coerced to delegate to a specific proxy, she can send all the values $h_0, ..., h_T$ to the proxy. As the value of h_A is published, the proxy can easily verify that $h_A = \mathcal{H}(h_T)$ and $h_i = \mathcal{H}(h_{i-1})$ hold, hence, the delegation is valid. As casting another ballot with the same value of h_i, or with a value h_j with $i < j$ (i.e. with a lower delegation priority) is not possible in the scheme, the proxy can ensure that her vote will count by using h_0 to cast it.

As the other proxy voting schemes, this scheme provides the functionality of cancelling the delegation by casting a direct vote, and of delegation back-up by using different priorities to delegate to different proxies.

8. Conclusion

Proxy voting constitutes a new voting mode in which voters are able to delegate their right to vote on issues beyond their expertise. At the same time, it also opens new attack vectors as there is a legitimate possibility to transfer a vote to another person who could be a coercer. To address these issues we created an Internet proxy voting scheme which focuses on coercion resistance and is based on the well-known JCJ/Civitas scheme. It provides functionality that enables vote delegation while at the same time ensuring the security of the delegation.

As our extension introduces additional credentials for delegation, which might overwhelm voters, an important part of future work would be to improve usability of the scheme. In the future, we will consider the proposal by Neumann and Volkamer [17] to address the credential handling in coercion-resistant proxy voting.

Acknowledgements

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).

This paper has been developed within the project 'VALID' - Verifiable Liquid Democracy - which is funded by the Polyas GmbH.

[1] Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Advances in Cryptology–EUROCRYPT 2012, pp. 263–280. Springer (2012)

[2] Bernhard, D., Neumann, S., Volkamer, M.: Towards a practical cryptographic voting scheme based on malleable proofs. In: E-Voting and Identify, pp. 176–192. Springer (2013)

[3] 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)

[4] Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Advances in Cryptology–CRYPTO 2004. pp. 41–55. Springer (2004)

[5] Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Advances in Cryptology—CRYPTO'92. pp. 89–105. Springer (1993)

[6] Clarkson, M.R., Chong, S., Myers, A.C.: Civitas: Toward a secure voting system. Tech. rep. (2008)

[7] Cortier, V., Galindo, D., Glondu, S., Izabachene, M.: Election verifiability for helios under weaker trust assumptions. In: Computer Security-ESORICS 2014, pp. 327–344. Springer (2014)

[8] Delaune, S., Kremer, S., Ryan, M.: Verifying privacy-type properties of electronic voting protocols: A taster. In: Towards Trustworthy Elections, pp. 289–309. Springer (2010)

[9] 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)

[10] Jakobsson, M.: On quorum controlled asymmetric proxy re-encryption. In: Public Key Cryptography. pp. 112–121. Springer (1999)

[11] Jakobsson, M., Juels, A.: Mix and match: Secure function evaluation via ciphertexts. In: Advances in Cryptology—ASIACRYPT 2000, pp. 162–177. Springer (2000)

[12] Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: Advances in Cryptology—EUROCRYPT'96. pp. 143–154. Springer (1996)

[13] 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)

[14] Kulyk, O., Marky, K., Neumann, S., Volkamer, M.: Introducing proxy voting to helios. In: Availability, Reliability and Security (ARES), 2016 11th International Conference on (2016)

[15] Langer, L., Schmidt, A., Buchmann, J., Volkamer, M.: A taxonomy refining the security requirements for electronic voting: analyzing helios as a proof of concept. In: Availability, Reliability, and Security, 2010. ARES'10 International Conference on. pp. 475–480. IEEE (2010)

[16] Neumann, S., Kahlert, A., Henning, M., Richter, P., Jonker, H., Volkamer, M.: Modeling the german legal latitude principles. In: 5th International Conference on eParticipation (ePart 2013). Lecture Notes in Computer Science, vol. 8075, pp. 49–56. Springer (Sep 2013)

[17] Neumann, S., Volkamer, M.: 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)

[18] Neumann, S., Volkamer, M.: A holistic framework for the evaluation of internet voting systems. Design, Development, and Use of Secure Electronic Voting Systems pp. 76–91 (2014)

[19] Schnorr, C.P.: Efficient signature generation by smart cards. Journal of cryptology 4(3), 161–174 (1991)

[20] Spycher, O., Koenig, R., Haenni, R., Schläpfer, M.: A new approach towards coercion-resistant remote e-voting in linear time. In: Proceedings of the 15th international conference on Financial Cryptography and Data Security. pp. 182–189. Springer-Verlag (2011)

[21] Tchorbadjiiski, A.: Liquid democracy diploma thesis. RWTH AACHEN University, Germany (2012)

[22] Terelius, B., Wikström, D.: Proofs of restricted shuffles. In: Progress in Cryptology–AFRICACRYPT 2010, pp. 100–113. Springer (2010)

[23] Weber, S.G., Araujo, R., Buchmann, J.: On coercion-resistant electronic elections with linear work. In: Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on. pp. 908–916. IEEE (2007)

[24] Zwattendorfer, B., Hillebold, C., Teufl, P.: Secure and privacy-preserving proxy voting system. In: e-Business Engineering (ICEBE), 2013 IEEE 10th International Conference on. pp. 472–477. IEEE (2013)