Ann. Telecommun. (2016) 71:323–336
DOI 10.1007/s12243-016-0519-6

Receipt-free remote electronic elections with everlasting privacy

Philipp Locher[1,2] · **Rolf Haenni**[1]

Received: 29 June 2015 / Accepted: 28 April 2016 / Published online: 12 May 2016

Abstract We present a new cryptographic voting protocol for remote electronic voting that offers three of the most challenging features of such protocols: verifiability, everlasting privacy, and receipt-freeness. Trusted authorities and computational assumptions are only needed during vote casting and tallying to prevent the creation of invalid ballots and to achieve receipt-freeness and fairness, but not to guarantee vote privacy. The implementation of everlasting privacy is based on perfectly hiding commitments and non-interactive zero-knowledge proofs, whereas receipt-freeness is realized with mix networks and homomorphic tallying.

Keywords Verifiable elections · Everlasting privacy · Receipt-freeness · Zero-knowledge proofs

1 Introduction

Voter coercion and vote buying are two serious threats in electronic elections. They have not newly emerged with the introduction of Internet elections, but they have reached a new dimension regarding their scalability. While both selling the right to vote and voting in someone else's name is usually regarded as a serious disruption of the normal voting process and may therefore imply severe legal consequences to both the seller and the buyer, the voting system should at least not encourage the practice of vote buyers paying rewards to voters for providing sufficient evidence that they voted in a particular way.

Benaloh and Tuinstra first remarked the important difference between concealing a vote and not revealing a vote to a third party [7]. A polling both in the traditional setting achieves both by physical measures. But this is not automatically the case in remote electronic voting. To achieve verifiability, many existing systems provide a *receipt* to voters, which allows them to verify the inclusion of their vote in the final result, but also to reveal their vote to someone else. Consequently, such systems are prone to vote buying and coercion. Receipt-freeness is therefore an important aspect of vote secrecy when the voter is dishonest.

In a system offering vote privacy, neither the system nor a third party can link a plaintext vote to a particular voter. If this property is not based on computational intractability assumptions, like the impossibility of computing discrete logarithms or factoring large numbers, nor on the availability of trusted authorities, then the privacy is called *everlasting* in a information-theoretical sense. Everlasting privacy is a desirable property to avoid vote privacy violations by much more powerful adversaries far in the future.

1.1 Contribution

The contribution of this paper is a new cryptographic voting protocol for verifiable electronic elections offering receipt-freeness and everlasting privacy. The protocol is a continuation of the one described in [28], which offers everlasting privacy without the need of trusted authorities but not receipt-freeness. In the new protocol, trusted authorities are needed to guarantee receipt-freeness and fairness, but

✉ Philipp Locher
philipp.locher@bfh.ch; philipp.locher@unifr.ch

[1] Bern University of Applied Sciences, 2501 Biel, Switzerland

[2] University of Fribourg, 1700 Fribourg, Switzerland

not for privacy. Consequently, if all trusted authorities collude and publish their private values, then voters are able to obtain a receipt, but the privacy of the vote is still given. The same applies to computational intractability assumptions. They are only needed to prevent the creation of invalid ballots during vote casting and to prevent voters from creating a receipt, but not to protect privacy in the long run.

The core of the protocol is a combination of a set membership proof [6] and a proof of known representation of a committed value [4]. When casting a vote, the voter provides a non-interactive zero-knowledge proof of knowledge of the representation of one of the registered public voter credentials. The same voter may cast multiple votes, but what counts at the end is the sum of all valid votes cast. In this way, precedent votes can be canceled out or overridden. The votes of a given voter are linked over an encrypted election credential, but the links remain hidden to an observer using a mix-net. The entire voting procedure consists of four consecutive steps:

- Registration: Each voter creates a pair of private and public voter credentials and sends the public credential over an authentic channel to the election administration.
- Election preparation: The election administration publishes the list of public voter credentials—one for every registered voter—on the public bulletin board.
- Vote casting: The voter creates an electronic ballot and sends it over an anonymous channel to the public bulletin board. The ballot consists of the encrypted vote, a commitment to the public credential, an encryption of the election credential, and the above-mentioned composition of zero-knowledge proofs.
- Tallying: The trusted authorities verify the proofs included in the ballots, shuffle the list of valid ballots in a mix-net, decrypt the election credentials, and accumulate under encryption the votes for each voter. The accumulated votes are shuffled in another mix-net and decrypted into plaintext votes.

This protocol provides everlasting privacy for the same reasons as its predecessor protocol presented in [28], i.e., all the identifying information contained in a ballot is either a perfectly hiding commitment or a zero-knowledge proof. Receipt-freeness is achieved by not revealing the number of votes cast by a single voter. The voter obtains a receipt for every single vote and can hand them over to the vote buyer, but the vote buyer will not know if the obtained amount of receipts is complete. The voter can therefore cheat the vote buyer by submitting an additional vote that cancels out or overrides all previous ones. For this to work, vote buyers must remain passive during the voting period, i.e., they do not cast votes in behalf of the voters. Generally, our protocol is not safe against active adversaries running impersonation or forced-abstention attacks.

1.2 Related work

A considerable amount of research has been conducted on receipt-free and coercion-resistant voting protocols and on systems offering everlasting privacy. But to the best of our knowledge, none of the existing approaches provides both properties simultaneously.

Benaloh and Tuinstra first mentioned the important difference between traditional paper-based voting in private voting booths and remote electronic voting based on a cryptographic voting protocol [7]. They were the first to define receipt-freeness and provide an approach for a possible solution based on an abstract model of a *voting booth* and the assumption that such voting booths exist. Based on a slightly weaker assumption of an *untappable channel*, Sako and Kilian described a receipt-free voting protocol for yes/no votes based on mix-nets [34]. Various other receipt-free protocols based on untappable channels have been proposed by different authors [24, 32, 39]. At the time of writing this paper, Kulyk et al. presented a method for achieving receipt-freeness in Helios by allowing voters to submit multiple votes and by considering the sum of all submitted votes in the final tally [27]. Their basic idea for achieving receipt-freeness is identical to the one presented in this paper, but their protocol does not offer everlasting privacy. The idea that the final counted vote is a composition of all submitted votes goes back to the non-cryptographic voting protocol ThreeBallot [33].

Juels et al. introduced an extended definition of coercion-resistance, which considers general impersonation, randomization, and forced-abstention attacks in addition to vote buying based on receipts [26]. Their solution requires an untappable channel only for registration. Additionally, the protocol assumes that voters have access to an anonymous channel at some point during the vote casting process. Unlike untappable channels, which require strong physical assumptions, anonymous channels can be implemented in practice, for example using mix-nets. Several follow-up papers have been published to speed up the tallying [2, 3, 15, 35, 36] or to prevent board flooding attacks [23].

With respect to vote privacy, Chaum argued that votes must be *unconditionally secure*, meaning that the partial tally of a group of voters can only be determined by a coalition of all other voters [14]. Moran and Noar introduced the term *everlasting privacy* in the context of a traditional setting, in which ballots are cast in a private polling booth [29–31]. In their definition, everlasting privacy is a weaker form of unconditional privacy, which only excludes that an attacker with unlimited computational power can break vote privacy. All three protocol are based on trusted authorities. Demirel et al. proposed several ways of achieving everlasting privacy in the context of remote electronic elections [10, 18, 19]. While the information published on the public

bulletin board does not reveal anything about somebody's vote, the trusted server could potentially break the ciphertext votes transmitted over the private channel between voter and server. The same restriction applies to the method presented in [17], which uses commitment consistent encryption to generate a perfectly private audit trail. The first protocol that offers everlasting privacy without trusted authorities is the one on which this paper is based [28].

Another important line of related work are the protocols based on blind signatures [16, 21, 25, 39]. They are also based on submitting votes over an anonymous channel, but they achieve everlasting privacy under much stronger trust assumptions. Their main problem is ballot-stuffing by malicious signing authorities, which cannot be detected. More generally speaking, protocols based on blind signatures do not support the verification of the electorate. Other disadvantages are the facts that voters need to interact with the authorities during vote casting and that the authorities learn who actually voted. To overcome some of the drawbacks of blind signatures, Canard and Traor introduced a system based on list signatures [13].

1.3 Paper overview

In the next section, we introduce the cryptographic building blocks of our protocol. In Section 3, we explain first the adversary model before we provide a detailed description of our protocol. We conclude the section with a discussion of the resulting security properties and possible protocol extensions. In Section 4, we analyze the running times and memory consumptions of the different protocol procedures and present the results from corresponding performance tests. Finally, we summarize the findings of this paper in Section 5.

2 Cryptographic preliminaries

Let $\mathcal{G}_p$ be a multiplicative cyclic group of prime order p, for which the discrete logarithm assumption is believed to hold. Furthermore, let $\mathbb{G}_q \subset \mathbb{Z}_p^*$, be a large prime-order subgroup of the group of integers modulo p, where $k = (p-1)/q$ denotes the corresponding co-factor. Finally, suppose that independent generators $g_0, g_1 \in \mathcal{G}_p$ and $h, h_0, h_1, \ldots, h_n \in \mathbb{G}_q$ are publicly known. Independence with respect to generators of a cyclic group means that their relative discrete logarithms are unknown.[1]

[1]To ensure that generators are independent, they need to be generated in some publicly reproducible way, for example by deriving them from a common reference string.

2.1 Homomorphic commitments and encryptions

In our protocol, we use two instances of the perfectly hiding Pedersen commitment scheme, one over $\mathcal{G}_p$ and one over $\mathbb{G}_q$. We distinguish them by $\mathrm{com}_p(u,r) = g_0^r g_1^u$ for a commitment to $u \in \mathbb{Z}_p$ with randomization $r \in \mathbb{Z}_p$ and $\mathrm{com}_q(v,s) = h_0^s h_1^v$ for a commitment to $v \in \mathbb{Z}_q$ with randomization $s \in \mathbb{Z}_q$. In the case of $\mathbb{G}_q$, we write $\mathrm{com}_q(v_1,\ldots,v_n,s) = h_0^s h_1^{v_1} \cdots h_n^{v_n}$ for a commitment to n values $v_1,\ldots,v_n \in \mathbb{Z}_q$. Recall that Pedersen commitments are perfectly hiding, computationally binding, and additively homomorphic.

The protocol also requires two instances of a homomorphic encryption scheme such as ElGamal or Paillier. One of them needs to be additively homomorphic. In our presentation, we choose ElGamal for its simplicity and compatibility with the above setting. We use one instance of standard ElGamal and one instance of exponential ElGamal, both over $\mathbb{G}_q$ and with a common key pair $x \in \mathbb{Z}_q$ and $y = h^x \in \mathbb{G}_q$. In our protocol, we will have a shared private key x generated in a distributed manner. Corresponding key shares x_i can be used to perform the decryption in a distributed way. Recall that ElGamal is IND-CPA secure under the decisional Diffie-Hellman (DDH) assumption. Furthermore, standard ElGamal is multiplicatively and exponential ElGamal additively homomorphic.

In case of standard ElGamal, we write $E = \mathrm{enc}_y^{\times}(m,r) = (h^r, my^r) \in \mathbb{G}_q \times \mathbb{G}_q$ for encrypting a message $m \in \mathbb{G}_q$ with randomization $r \in \mathbb{Z}_q$ and $m = \mathrm{dec}_x^{\times}(E) = ba^{-x}$ for decrypting a given ciphertext $E = (a,b)$ with the private key x. To decrypt multiple ciphertexts $\mathbf{E} = \{E_1,\ldots,E_n\}$ using the same private key x, we write $\mathbf{M} = \mathrm{dec}_x^{\times}(\mathbf{E})$ for the resulting list $\mathbf{M} = (m_1,\ldots,m_n)$ of plaintext messages $m_i = \mathrm{dec}_x^{\times}(E_i)$.

In the case of exponential ElGamal, let $E = \mathrm{enc}_y^{+}(m,r) = (h^r, h^m y^r) \in \mathbb{G}_q \times \mathbb{G}_q$ denote the encryption of a message $m \in \mathcal{M} \subset \mathbb{Z}_q$ with randomization $r \in \mathbb{Z}_q$, where $\mathcal{M}$ is small enough to conduct an exhaustive search. The restriction on the message space is necessary to perform the decryption $m = \mathrm{dec}_x^{+}(E) = \log_h(ba^{-x})$ of a ciphertext $E = (a,b)$ efficiently. Again, to decrypt multiple ciphertexts using the same private key x, we write $\mathbf{M} = \mathrm{dec}_x^{+}(\mathbf{E})$. To re-encrypt a given ciphertext $E = \mathrm{enc}_y^{+}(m,r)$ with a new randomization r', we use the standard procedure $E' = \mathrm{reEnc}_y^{+}(E,r') = E \cdot \mathrm{enc}_y^{+}(0,r') = \mathrm{enc}_y^{+}(m, r+r')$ of multiplying the ciphertext with an encryption of 0.

2.2 Zero-knowledge proofs

The main cryptographic tools in our protocol are non-interactive zero-knowledge proofs of knowledge. The voter uses them to demonstrate knowledge of some secret values involved in a mathematical statement, but without revealing

any information about the secret values. One of the most fundamental type of zero-knowledge proofs of knowledge is a preimage proof for a one-way group homomorphism $\phi : X \to Y$, denoted by

$$NIZKP[(a) : b = \phi(a)],$$

where $a \in X$ is the secret preimage of a public value $b = \phi(a) \in Y$. Examples of such preimage proofs result from the above homomorphic Pedersen commitment and ElGamal encryption schemes, for example $NIZKP[(u, r) : C = \text{com}_p(u, r)]$ for proving knowledge of the opening of a commitment, $NIZKP[(m, r) : E = \text{enc}_y^+(m, r)]$ for proving knowledge of the plaintext and randomization of an exponential ElGamal ciphertext, or $NIZKP[(x) : \mathbf{M} = \text{dec}_x^\times(\mathbf{E}) \wedge y = h^x]$ for proving knowledge of the private key used in a batch decryption.

The most common construction of a non-interactive preimage proof is the Σ-protocol in combination with the Fiat-Shamir heuristic [20]. Proofs constructed in this way are perfect zero-knowledge in the random oracle model. Their transcript consists of one or multiple commitments and one or multiple responses to a challenge obtained from querying the random oracle with the public inputs and the commitments. In practice, the random oracle is implemented with a cryptographic hash function. In Section 3, we will write $\pi_i = NIZKP[\cdot]$ for the transcripts of the non-interactive proofs used in the voting protocol.

2.2.1 Set membership proof

Let $U = \{u_1 \ldots, u_M\}$ be a finite set of values $u_i \in \mathbb{Z}_p$ and $C = \text{com}_p(u, r)$ a commitment to an element $u \in U$. Both U and C are publicly known. With a *set membership proof*, denoted by

$$NIZKP[(u, r) : C = \text{com}_p(u, r) \wedge u \in U],$$

the prover demonstrates knowledge of corresponding values $u \in U$ and $r \in \mathbb{Z}_p$, but without revealing any information about them. Such a proof can be constructed by a standard OR combination of individual preimage proofs for each $u \in U$, but this proof has a size linear to M and is therefore not efficient. The first set membership proof with a sub-linear size has been given by Camenisch et al. [11].

As suggested by Brands et al. [9], a general way of constructing a set membership proof is to compute the polynomial $P(X) = \prod_{i=1}^{M}(X - u_i)$ and to demonstrate that $P(u) = 0$. This proof, denoted by

$$NIZKP[(u, r) : C = \text{com}_p(u, r) \wedge P(u) = 0],$$

is a particular case of a *polynomial evaluation proof*. In a recent publication [6], Bayer and Groth proposed a polynomial evaluation proof with a logarithmic size, which is the current state-of-the-art. A summary of the proof generation and verification using the same formal notation and cryptographic setting as introduced above is given in [28].

2.2.2 Proof of known representation

In a cyclic group such as $\mathbb{G}_q$ with generators $h_1, \ldots, h_n$, a tuple $(v_1, \ldots, v_n)$ of values $v_i \in \mathbb{Z}_q$ is called *DL-representation* (or simply *representation*) of $u \in \mathbb{G}_q$ with respect to values $(h_1, \ldots, h_n)$, if $u = h_1^{v_1} \cdots h_n^{v_n}$ [8]. Note that the general definition of DL-representation does not require the values $h_1, \ldots, h_n$ to be generators, nor do they need to be independent or distinct. On the other hand, every opening of a Pedersen commitment is clearly a DL-representation of the commitment with respect to the given independent generators.

Let $C = \text{com}_p(u, r)$ be a commitment to a single value $u \in \mathbb{G}_q \subset \mathbb{Z}_p$ and $D = \text{com}_q(v_1, \ldots, v_n, s)$ a commitment to multiple values $v_1, \ldots, v_n \in \mathbb{Z}_q$. Both C and D are publicly known. Following Au et al. [4], a proof of known *representation of a committed value* (or simply *representation proof*), denoted by

$$NIZKP[(u, r, v_1, \ldots, v_n, s) : C = \text{com}_p(u, r) \wedge D = \text{com}_q(v_1, \ldots, v_n, s) \wedge u = h_1^{v_1} \cdots h_n^{v_n}],$$

demonstrates that the tuple of committed values in D is a DL-representation of the committed value in C. Note that this is a generalization of proof of knowledge of double discrete logarithms, $NIZKP\{(v) : C = g^{(h^v)}\}$, by Camenisch and Stadler [12]. A summary of the representation proof generation and verification is given in [28].

2.3 Cryptographic shuffling

The input of a cryptographic shuffle is a list $\mathbf{Z} = (z_1, \ldots, z_n)$ of input values $z_i \in Z$. The mixer applies a keyed one-way function $f_{k_i} : Z \to Z$ to each input value z_i and permutes the results by picking a random permutation $\phi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ from the set Φ_n of permutations of length n. The output of a cryptographic shuffle is therefore a list $\mathbf{Z}' = (z'_1, \ldots, z'_n)$ of values $z'_j = f_{k_i}(z_i)$ for indices $j = \phi(i)$. We write

$$\mathbf{Z}' = \text{shuffle}_{f_K}^{\phi}(\mathbf{Z})$$

for the whole procedure, where $K = (k_1, \ldots, k_n)$ denotes the list of involved keys. Since the goal of a cryptographic shuffle is to unlink the output values from corresponding inputs, the shuffling is usually performed multiple times in a mix network by independent mixers. The unlinkability in such a network is guaranteed as long as at least one

permutation remains secret. Additionally, each mixer needs to provide a non-interactive zero-knowledge proof,

$$NIZKP[(K, \phi) : \mathbf{Z}' = \text{shuffle}^{\phi}_{f_K}(\mathbf{Z})],$$

to prove the correctness of the shuffle. There are several competing techniques for providing such proofs [5, 37].

In our protocol, we need two instances of a cryptographic shuffle. In the first case, the input values are pairs of ElGamal ciphertexts $(E_i, F_i) \in (\mathbb{G}_q \times \mathbb{G}_q) \times (\mathbb{G}_q \times \mathbb{G}_q)$. For random values $\delta \in_R \mathbb{Z}_q \setminus \{0\}$ and $\sigma_i \in_R \mathbb{Z}_q$, the function $f_{\delta,\sigma_i}(E_i, F_i) = (E_i^{\delta}, \text{reEnc}_y^+(F_i, \sigma_i))$ is applied to each input for keys $k_i = (\delta, \sigma_i)$. Note that raising E_i to the power of δ disconnects both the plaintext and ciphertext from their original values, whereas re-encrypting F_i only disconnects the ciphertext. To prove the correctness of the shuffle, the mixer computes

$$NIZKP[(\delta, \sigma_1, \ldots, \sigma_n, \phi) : \mathbf{Z}' = \text{shuffle}^{\phi}_{f_{(\delta,\sigma_1),\ldots,(\delta,\sigma_n)}}(\mathbf{Z})]$$

to prove knowledge of $K = ((\delta, \sigma_1), \ldots, (\delta, \sigma_n))$ and ϕ.

The second instance of a cryptographic shuffle is a special case of the first one. The inputs are single ElGamal ciphertexts $F_i \in \mathbb{G}_q \times \mathbb{G}_q$, which are re-encrypted using $f_{\sigma_i}(F_i) = \text{reEnc}_y^+(F_i, \sigma_i)$ for random values $\sigma_i \in_R \mathbb{Z}_q$. With the corresponding non-interactive proof,

$$NIZKP[(\sigma_1, \ldots, \sigma_n, \phi) : \mathbf{Z}' = \text{shuffle}^{\phi}_{f_{\sigma_1,\ldots,\sigma_n}}(\mathbf{Z})],$$

the mixer proves knowledge of $K = (\sigma_1, \ldots, \sigma_n)$ and ϕ.

3 Receipt-free elections with everlasting privacy

In this section, we present our new protocol for receipt-free electronic elections with everlasting privacy. We start with a discussion of the adversary model and the underlying trust assumptions. Then, we provide a detailed formal description of the protocol, analyze its security properties, and propose three protocol extensions.

3.1 Adversary model and trust assumptions

We consider three types of adversaries with different capabilities and goals. An adversary of the first type acts at the present time, i.e., before, during, or shortly after an election, whereas an adversary of the second type acts at any point in the future. We call them *present adversary* and *future adversary*, respectively. The *vote buyer* is a special case of a present adversary.

The general goal of present adversaries is to break the integrity or secrecy of the votes during an election, for example by submitting votes in the name of someone else or by linking votes to voters. We assume present adversaries to be polynomially bounded and thus incapable of solving the DL or DDH problems in large prime order groups or breaking cryptographic primitives such as contemporary hash functions. This implies that present adversaries cannot efficiently find valid openings of Pedersen commitments or valid proof transcripts for zero-knowledge proofs of knowledge without knowing the secret inputs. We also assume that present adversaries cannot control the machines used for vote casting[2] and that voters do not reveal their private credentials.

The goal of a vote buyer is to manipulate the outcome of an election by paying rewards to voters if they can prove that they voted in a particular way. The information required to convince the vote buyer is called *receipt*. We assume that vote buyers pay rewards for such receipts, but not for obtaining the voters' private credentials. In other words, we exclude active impersonation attacks by buying someone's right to vote. Vote buyers remain entirely passive during an election and therefore do not interfere with the vote casting process.

For a future adversary, the only goal is breaking the secrecy of the votes of an election that took place at the present time. To avoid the problem of estimating the available computational resources far in the future, we simply assume the strongest possible adversary, one with unlimited resources in terms of computational power and time. Although contemporary cryptography will be completely useless in the presence of such an adversary, the secrets hidden in perfectly hiding commitments or in zero-knowledge proofs of knowledge will never be revealed, even if they were generated today.

From the point of view of the necessary communication infrastructure, the protocol requires an authentic channel between voter and election administration during the registration process. In the basic protocol version of Section 3.2, voters need to re-register in every new election, but we will show later how to circumvent this limitation. Furthermore, the protocol requires a broadcast channel with memory, for example in the form of a robust append-only public bulletin board collecting the entire election data. We assume that the election administration and the trusted authorities have their own designated areas on the bulletin board for posting their messages. Finally, for sending their votes to the bulletin board, voters need access to an anonymous channel. We assume that no adversary is capable of intercepting and recording the whole traffic over this channel during an election and storing the data for future vote privacy attacks [1].

[2] We are aware that requiring a secure platform is a strong assumption. We do not explicitly address this problem in this paper, but our protocol allows voters at least to detect a compromised platform as long as they can read the bulletin board in a secure way.

Registration (Voter):
1. Pick private credential $(\alpha, \beta, \gamma) \in_R \mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q$.
2. Compute public credential $u = h_1^\alpha h_2^\beta h_3^\gamma \in \mathbb{G}_q$.
3. Send u over an authentic channel to the election administration.

Fig. 1 Summary of the registration phase

3.2 Protocol description

As outlined in Section 1.1, the protocol consists of four consecutive phases. We will now present the details of each phase using the cryptographic primitives and formal notation introduced in the previous section. Summaries of all phases are included in corresponding figures at the end of each subsection. Note that the registration and election preparation phase are identical to the predecessor protocol [28], and vote casting is very similar. To achieve receipt-freeness, complexity has been added mainly to the tallying phase. At the end of this subsection, we will also give an overview of the verification process.

3.2.1 Registration

The first step of the protocol is the registration of voters before an election. To register, voter V picks a *private credential* $(\alpha, \beta, \gamma) \in_R \mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q$ at random and computes the *public credential* $u = h_1^\alpha h_2^\beta h_3^\gamma \in \mathbb{G}_q$. Note that the private credential is a representation of the public credential with respect to (h_1, h_2, h_3). Finally, the voter sends u over an authentic channel to the election administration (Fig. 1).[3]

3.2.2 Election preparation

After the registration phase, the election administration defines the list $\mathbf{U} = ((V_1, u_1), \ldots, (V_M, u_M))$ based on the electoral roll. Each pair $(V_i, u_i) \in \mathbf{U}$ links a public credential to the corresponding voter. Next, the list $\mathbf{A} = (a_0, \ldots, a_M)$ of coefficients $a_i \in \mathbb{Z}_p$ of the polynomial $P(X) = \prod_{i=1}^{M}(X - u_i) \in \mathbb{Z}_p[X]$ is computed to allow voters creating the set membership proof during the vote casting phase. As the computation of those coefficients is quite expensive ($\frac{1}{2}M^2$ multiplications in $\mathbb{Z}_p$), it is performed by the election administration, possibly already during the registration phase in an incremental way. Note that the coefficients can be re-computed and verified by anyone, and voters can efficiently verify the inclusion of their public credential u by checking $P(u) = 0$. Finally, two independent *election generators* $\hat{h}_1, \hat{h}_2 \in \mathbb{G}_q$ are defined in some publicly reproducible way and $(\mathbf{U}, \mathbf{A}, \hat{h}_1, \hat{h}_2)$ is posted into the administration's designated area of the public bulletin board (Fig. 2).

3.2.3 Vote casting

During the election, voters select their vote by choosing their preferred election options and encoding them by an element of the set $\mathbb{V} \subseteq \mathbb{Z}_q$ of valid votes. We assume that the election options and their encoding in $\mathbb{V}$ are publicly known. Note that nothing prevents the voter from selecting and submitting an invalid vote $v \notin \mathbb{V}$. In fact, we explicitly allow the submission of arbitrary values $v \in \mathbb{Z}_q$ as a mechanism for canceling out or overriding votes submitted previously by the same voter. In our protocol, the sum of all submitted votes is what counts at the end for a single voter, and this value must be in $\mathbb{V}$ to be included in the final tally. Therefore, submitting $-v$ cancels out a previously submitted value v, whereas $v' - v$ overrides v with v'.

To cast the vote, the voter computes two commitments $C = \mathrm{com}_p(u, r)$ and $D = \mathrm{com}_q(\alpha, \beta, \gamma, s)$ to the public and private credentials. Next, the voter computes a standard ElGamal encryption $E = \mathrm{enc}_y^\times(\hat{u}, \rho)$ of the *election credential* $\hat{u} = \hat{h}_1^\alpha \hat{h}_2^\beta \in \mathbb{G}_q$ and an exponential ElGamal encryption $F = \mathrm{enc}_y^+(v, \sigma)$ of the vote v. For this, we assume that the public key y of the trusted authorities is known to everyone. Finally, the voter generates three non-interactive zero-knowledge proofs. The first proof,

$$\pi_1 = NIZKP[(u, r) : C = \mathrm{com}_p(u, r) \wedge P(u) = 0],$$

is a set membership proof proving that C is indeed a commitment to the public credential of one of the eligible voters listed in U. The second proof,

$$\pi_2 = NIZKP[(r, \alpha, \beta, \gamma, s) : C = \mathrm{com}_p(u, r) \wedge D = \mathrm{com}_q(\alpha, \beta, \gamma, s) \wedge u = h_1^\alpha h_2^\beta h_3^\gamma],$$

Election Preparation (Election Administration):
1. Define $\mathbf{U} = ((V_1, u_1), \ldots, (V_M, u_M))$ based on the electoral roll.
2. Compute coefficients $\mathbf{A} = (a_0, \ldots, a_M)$ of $P(X) = \prod_{i=1}^{M}(X - u_i) \in \mathbb{Z}_p[X]$.
3. Define election generators $\hat{h}_1, \hat{h}_2 \in \mathbb{G}_q$.
4. Post $(\mathbf{U}, \mathbf{A}, \hat{h}_1, \hat{h}_2)$ into the designated area of the bulletin board.

Fig. 2 Summary of the election preparation phase

[3] To ensure that u has been computed from fresh values (α, β, γ), the voter could be asked to prove knowledge of (α, β, γ) by computing $NIZKP[(\alpha, \beta, \gamma) : u = h_1^\alpha h_2^\beta h_3^\gamma]$. As this is not an essential step for our protocol, we omit it in our presentation.

is a proof of known representation of the committed value in C. Its purpose is to prevent voters from taking just any credential from U. The third proof,

$$\pi_3 = NIZKP[(\alpha, \beta, \gamma, s, \rho, v, \sigma) : D = \mathrm{com}_q(\alpha, \beta, \gamma, s) \wedge E = \mathrm{enc}_y^{\times}(\hat{u}, \rho) \wedge F = \mathrm{enc}_y^{+}(v, \sigma) \wedge \hat{u} = \hat{h}_1^{\alpha}\hat{h}_2^{\beta}],$$

is a standard preimage proof showing that D and E have been generated using the same values α and β and that the vote contained in F is known to the voter.[4]

The ballot $B = (C, D, E, F, \pi_1, \pi_2, \pi_3)$ consisting of the two commitments, the two ciphertexts, and the three proofs is posted over an anonymous channel to the bulletin board. The voter may submit multiple such ballots during the election period. If multiple identical copies of the same ballot are posted to the bulletin board, we assume that only one of them is stored (Fig. 3).[5]

3.2.4 Tallying

At the end of the election period, the ballots submitted to the bulletin board need to be processed by the trusted authorities. We present this process by looking at the group of trusted authorities as a single entity performing the necessary shuffling and decryption tasks jointly. In reality, different trusted authorities will perform respective tasks using their own secret inputs and random values. The cryptographic shuffling is a serial and the distributed decryption (usually) a parallel process.[6]

To initiate the tallying process, the trusted authority retrieves the list $\mathbf{B}$ of all ballots from the bulletin board and verifies the non-interactive proofs π_1, π_2, π_3 for each ballot $B = (C, D, E, F, \pi_1, \pi_2, \pi_3) \in \mathbf{B}$. Ballots with invalid proofs are dropped. Then the two ElGamal ciphertexts (E, F) are selected from all ballots with valid proofs. We denote the resulting list of such pairs by $\mathbf{EF} = ((E_1, F_1), \ldots, (E_N, F_N))$. The validity of the proofs guarantees that each $(E_i, F_i) \in \mathbf{EF}$ originates from an eligible voter with valid private credentials. Clearly, two distinct pairs $(E_i, F_i), (E_j, F_j) \in \mathbf{EF}$ originate from the same eligible voter, whenever E_i and E_j contain the same plaintext.

[4] At first sight, it may appear that π_2 and π_3 are very similar proofs, but a subtle difference disallows π_2 to be implemented as a standard preimage proof. The subtlety lies in the fact that u and $\hat{u}$ are both elements of $\mathbb{G}_q$, but to use u as input of $\mathrm{com}_p : \mathbb{Z}_p \times \mathbb{Z}_p \to \mathscr{G}_p$, it needs to be interpreted as an element of $\mathbb{Z}_p$. As a consequence, com_p is not a group homomorphism with respect to (α, β, γ), i.e., the preconditions for constructing a preimage proof are not satisfied.

[5] The bulletin board could also accept multiple copies of the same ballot, which then need to be eliminated in the tallying phase. But this makes preventing replay and board flooding attacks more complicated.

[6] Shuffling and decrypting a list of ciphertexts can be performed in a single serial process [38]. This is an optional implementation variant of our protocol, which we do not pursue here.

Vote Casting (Voter):

1. Pick $r \in_R \mathbb{Z}_p$. Compute $C = \mathrm{com}_p(u, r) \in \mathscr{G}_p$.
2. Pick $s \in_R \mathbb{Z}_q$. Compute $D = \mathrm{com}_q(\alpha, \beta, \gamma, s) \in \mathbb{G}_q$.
3. Compute election credential $\hat{u} = \hat{h}_1^{\alpha}\hat{h}_2^{\beta}$.
4. Pick $\rho \in_R \mathbb{Z}_q$. Compute $E = \mathrm{enc}_y^{\times}(\hat{u}, \rho) \in \mathbb{G}_q \times \mathbb{G}_q$.
5. Select vote $v \in \mathbb{Z}_q$.
6. Pick $\sigma \in_R \mathbb{Z}_q$. Compute $F = \mathrm{enc}_y^{+}(v, \sigma) \in \mathbb{G}_q \times \mathbb{G}_q$.
7. Compute the following non-interactive proofs:

$$\pi_1 = NIZKP[(u, r) : C = \mathrm{com}_p(u, r) \wedge P(u) = 0],$$
$$\pi_2 = NIZKP[(r, \alpha, \beta, \gamma, s) : C = \mathrm{com}_p(u, r) \wedge D = \mathrm{com}_q(\alpha, \beta, \gamma, s) \wedge u = h_1^{\alpha}h_2^{\beta}h_3^{\gamma}],$$
$$\pi_3 = NIZKP[(\alpha, \beta, \gamma, s, \rho, v, \sigma) : D = \mathrm{com}_q(\alpha, \beta, \gamma, s) \wedge E = \mathrm{enc}_y^{\times}(\hat{u}, \rho) \wedge F = \mathrm{enc}_y^{+}(v, \sigma) \wedge \hat{u} = \hat{h}_1^{\alpha}\hat{h}_2^{\beta}].$$

8. Post ballot $B = (C, D, E, F, \pi_1, \pi_2, \pi_3)$ to the bulletin board over an anonymous channel.

Fig. 3 Summary of the vote casting phase

In the next step, the trusted authority performs a cryptographic shuffle on the list $\mathbf{EF}$. The two ciphertexts of each element (E_i, F_i) of the input list are treated differently: E_i is disguised by raising it to the power of a single random value $\delta \in_R \mathbb{Z}_q \setminus \{0\}$, whereas F_i is disguised by re-encrypting it using an individual random value $\sigma_i \in_R \mathbb{Z}_q$. Therefore, the trusted authority applies the keyed one-way function

$$f_{\delta,\sigma_i}(E_i, F_i) = (E_i^{\delta}, \mathrm{reEnc}_y^{+}(F_i, \sigma_i))$$

to each input $(E_i, F_i) \in \mathbf{EF}$, where $k_i = (\delta, \sigma_i)$ represents the individual key. By selecting a random permutation $\phi \in_R \Phi_N$ and applying it to the resulting list, the trusted authority generates a new list

$$\mathbf{EF}' = \mathrm{shuffle}^{\phi}_{f_{(\delta,\sigma_1),\ldots,(\delta,\sigma_N)}}(\mathbf{EF})$$

of shuffled ciphertext pairs. Note that $\mathbf{EF}'$ inherits from $\mathbf{EF}$ the property that two distinct pairs $(E'_i, F'_i), (E'_j, F'_j) \in \mathbf{EF}'$ originate from the same eligible voter, whenever E'_i and E'_j contain the same plaintext.

To collect the votes that originate from a single voter, the trusted authority selects E'_i from every $(E'_i, F'_i) \in \mathbf{EF}'$ and decrypts the resulting list $\mathbf{E}' = (E'_1, \ldots, E'_N)$ into $\mathbf{H} = \mathrm{dec}_x^{\times}(\mathbf{E}')$. Each plaintext $H \in \mathbf{H}$ is a value of the form $H = \mathrm{dec}_x^{\times}(E'_i) = \hat{u}^{\delta} = (\hat{h}_1^{\alpha}\hat{h}_2^{\beta})^{\delta} = (\hat{h}_1^{\delta})^{\alpha}(\hat{h}_2^{\delta})^{\beta}$, where $\hat{h}_1^{\delta}$ and $\hat{h}_2^{\delta}$ are unique values for the current election and (α, β) belongs to some voter's private credential. In other words, all $(E'_i, F'_i) \in \mathbf{EF}'$ satisfying $H = \mathrm{dec}_x^{\times}(E'_i)$ for some $H \in \mathbf{H}$ originate from the same voter, which implies that $F_H = \prod F'_i$ is an encryption of the corresponding sum of votes. Recall that this sum is what counts in our protocol for a particular voter. Let $\mathbf{F}_H$ denote the list of all values F_H aggregated in this way and $N' = |\mathbf{F}_H| \leq N$ the size of the list.

Before decrypting the aggregated votes, the trusted authority performs a final cryptographic shuffle on $\mathbf{F}_H$, in which the input values are re-encrypted. For this, random values $\sigma'_1, \ldots, \sigma'_{N'} \in_R \mathbb{Z}_q$ are selected to perform the re-encryption and a random permutation $\phi' \in_R \Phi_{N'}$ is selected for the shuffling. The result is a new list

$$\mathbf{F}'_H = \text{shuffle}^{\phi'}_{f_{\sigma'_1,\ldots,\sigma'_{N'}}}(\mathbf{F}_H)$$

of ciphertext votes, which can then be decrypted into a list $\mathbf{V} = \text{dec}^+_x(\mathbf{F}'_H)$ of plaintext votes. Note that each resulting plaintext vote is a value from $\mathbb{Z}_q$, but not necessarily all of them represent valid votes. We denote the process of filtering out invalid votes by $\mathbf{V}' = \mathbf{V} \cap \mathbb{V}$.[7] The resulting list $\mathbf{V}'$ is the election result.

In addition to the above computational steps, the trusted authority needs to provide convincing evidence that respective shuffling and decryption tasks have been performed properly. The following non-interactive proofs are necessary for this:

$$\pi_E = NIZKP[(\delta, \sigma_1, \ldots, \sigma_N, \phi) : \mathbf{EF}' = \text{shuffle}^{\phi}_{f_{(\delta,\sigma_1),\ldots,(\delta,\sigma_N)}}(\mathbf{EF})],$$

$$\pi_H = NIZKP[(x) : \mathbf{H} = \text{dec}^{\times}_x(\mathbf{E}') \wedge y = h^y],$$

$$\pi_F = NIZKP[(\sigma'_1, \ldots, \sigma'_{N'}, \phi') : \mathbf{F}'_H = \text{shuffle}^{\phi'}_{f_{\sigma'_1,\ldots,\sigma'_{N'}}}(\mathbf{F}_H)],$$

$$\pi_V = NIZKP[(x) : \mathbf{V} = \text{dec}^+_x(\mathbf{F}'_H) \wedge y = h^y].$$

By posting $(\mathbf{EF}, \mathbf{EF}', \mathbf{E}', \mathbf{H}, \mathbf{F}_H, \mathbf{F}'_H, \mathbf{V}, \mathbf{V}', \pi_E, \pi_H, \pi_F, \pi_V)$ to the bulletin board, the trusted authority completes the tallying process.[8]

3.2.5 Verification

To verify the election result, the election data must be retrieved from the bulletin board. It consists of everything that has been posted to the bulletin board during the election preparation, vote casting, and tallying phase:

$$(\mathbf{U}, \mathbf{A}, \hat{h}_1, \hat{h}_2, \mathbf{B}, \mathbf{EF}, \mathbf{EF}', \mathbf{E}', \mathbf{H}, \mathbf{F}_H, \mathbf{F}'_H, \mathbf{V}, \mathbf{V}', \pi_E, \pi_H, \pi_F, \pi_V).$$

The full verification process consists of the following steps:

- Verify the electoral roll $\mathbf{U}$ by checking the identities of all eligible voters.
- Reproduce the list of coefficients $\mathbf{A}$ of the polynomial defined by $\mathbf{U}$.
- Reproduce the election generators $\hat{h}_1$ and $\hat{h}_2$.
- For each ballot $B = (C, D, E, F, \pi_1, \pi_2, \pi_3) \in \mathbf{B}$, verify the proofs π_1, π_2, π_3.
- Reproduce $\mathbf{EF}$ from $\mathbf{B}$.
- Verify the shuffle proof π_E relative to $\mathbf{EF}$ and $\mathbf{EF}'$.
- Reproduce $\mathbf{E}'$ from $\mathbf{EF}'$.
- Verify the decryption proof π_H relative to $\mathbf{E}'$ and $\mathbf{H}$.
- Reproduce $\mathbf{F}_H$ from $\mathbf{H}$ and $\mathbf{EF}'$.
- Verify the shuffle proof π_F relative to $\mathbf{F}_H$ and $\mathbf{F}'_H$.
- Verify the decryption proof π_V relative to $\mathbf{F}'_H$ and $\mathbf{V}$.
- Reproduce $\mathbf{V}'$ from $\mathbf{V}$.

Verifying the correctness of the cryptographic setting would be another item in the above list, but usually one can assume that the setting has been checked by others.

3.3 Security properties

We will now look at our protocol from the perspective of its security properties. We provide an informal discussion of how correct election results, everlasting privacy, and receipt-freeness is achieved. Fairness is achieved in a trivial way by submitting votes encrypted.

Correctness For a present adversary not colluding with any of the trusted authorities and not in possession of a private credential, there are two principle ways of creating a ballot that will be accepted in the final tally. First, the adversary may try to find $(\alpha', \beta', \gamma')$ such that $u = h_1^{\alpha'} h_2^{\beta'} h_3^{\gamma'}$ for some u in U, but this is equivalent to solving the discrete logarithm problem. Second, the adversary may try to fake a proof transcript without knowing such values $(\alpha', \beta', \gamma')$, but this is impossible due to the computational soundness of the proofs π_1, π_2, and π_3.

If the present adversary is an eligible voter in possession of a valid private credential, then using it for submitting more than one ballot is explicitly allowed by the protocol and results in one final accumulated vote. The malicious voter could try to prevent the vote accumulation by submitting ballots with different election credentials, but the soundness of π_3 does not allow this. Without using the private credential, the voter is not more powerful than any other present adversary.

A present adversary colluding with one or several trusted authorities—or even the authorities themselves—may try to delete, modify, or add votes in the mixing or decryption steps of the protocol, but this is prevented by the computational soundness of the proofs π_E, π_H, π_F, and π_V. Their correctness can be verified by anyone.

Everlasting privacy A ballot posted over an anonymous channel to the bulletin board contains no information for identifying the voter. Clearly, the future adversary will be able to determine the private key x and use it to decrypt

[7] By mixing up list and set operations in one expression, we slightly abuse standard mathematical notation.

[8] Some lists are implicitly given: $\mathbf{EF}$ (follows from $\mathbf{B}$), $\mathbf{E}'$ (follows from $\mathbf{EF}'$), $\mathbf{F}_H$ (follows from $\mathbf{H}$ and $\mathbf{EF}'$), and $\mathbf{V}'$ (follows from $\mathbf{V}$). As such, they need not to be published, but we include them for improved clarity.

E into $\hat{u} = \hat{h}_1^{\alpha}\hat{h}_2^{\beta}$, but this value is perfectly hiding with respect to both α and β. Similarly, $u = h_1^{\alpha}h_2^{\beta}h_3^{\gamma}$ is perfectly hiding with respect to α, β, and γ. Therefore, knowing x does not establish a link from $E = \text{enc}_y^{\times}(\hat{u}, \rho)$ to u. Since the proofs π_1, π_2, and π_3 are zero-knowledge and therefore of no additional help, even a future adversary is unable to break vote privacy.

Receipt-freeness A voter may send multiple ballots with valid proofs during the vote casting phase, and all of them will count in the final tally. By disclosing the randomizations used in the encryptions F, the authorship of a single ballot can be proven and its plaintext vote can be revealed. However, it is impossible to prove that every other ballots was cast by somebody else (this would mean to prove *not* knowing corresponding randomizations). As a single additional ballot can cancel out or overrule all precedent votes, proving the authorship of one or multiple ballots does not give a conclusive receipt.

In case the voter reveals the encryption randomizations for some (but not all) ballots to the vote buyer, the link to the voter's other ballots must remain hidden during tallying. By disclosing the randomization of E in addition to the randomization of F, the voter can also reveal the election credential $\hat{u} = \hat{h}_1^{\alpha}\hat{h}_2^{\beta}$. In the first shuffle, by raising $E = \text{enc}_y^{\times}(\hat{u}, \rho)$ to the power of δ, the elections credentials are disguised under encryption. Since δ is only known to the trusted authorities (by holding corresponding shares of δ), decrypting the resulting value E^{δ} into $\hat{u}^{\delta}$ does not reveal a link to $\hat{u}$ to anyone not in possession of δ and unable to compute discrete logarithms efficiently. The outcome of the first shuffle and the subsequent decryption is therefore not uncovering the voter's remaining ballots. This could only happen if all trusted authorities collude or by someone capable of solving the discrete logarithm problem, but the trust assumptions in our adversary model excludes this.

Another potential way of uncovering the ballots of a given voter during tallying is by marking the submitted votes with some additional information that the vote buyer would accept as a receipt. This could be done in three different ways: by submitting a unique combinations of annihilating values $v \in \mathbb{Z}_q$ in addition to the real vote, by submitting a unique total number of votes, or by submitting a unique valid vote when $\mathbb{V}$ is large enough. Each of these cases would give a conclusive receipt.

- To prevent the first type of receipt, the ciphertext votes obtained from the first mix-net are not decrypted directly. Instead, the ciphertext votes are accumulated and shuffled in a second mix-net. In this way, the selected combination of annihilating votes remains hidden between the two mix-nets, only the sum of all votes is decrypted.
- Regarding the second type of receipt, the protocol as described in the previous section does not include any counter-measures. The voter and the vote buyer could therefore agree in advance on the total number of submitted votes. To receive the reward, the voter will then disclose the same number of encryption randomizations. If the list **H** from step 4 contains exactly one value with again the same number of identical copies, the vote buyer will accept the disclosed randomizations as receipt. Because the agreed amount of votes must be unique for each voter, the scalability of constructing receipts of this type is limited for a large electorate. In Section 3.4, we propose a possible protocol extension to limit the scalability even for a small electorate.
- The third type of receipt is known as the *Italian attack*. An Italian attack is always possible if the final votes are decrypted individually and if $\mathbb{V}$ is much larger than the electorate. The problem is therefore limited to elections with many options and complex rules or to elections with a small electorate. The protocol as presented so far does not include any counter-measures to prevent the attack in these cases, but we will present in Section 3.4 an extended tallying procedure that avoids the decryption of the accumulated votes of individual voters.

3.4 Extensions

In the basic version of the protocol as presented in Section 3.2, we have ignored some problems regarding receipt-freeness and some important practical aspects. The following discussion of corresponding extensions rounds off the description of our protocol.

Null votes This extension addresses the problem that voters and vote buyers may agree in advance on the total number of submitted votes. The idea is to artificially increase the number of votes of a given voter by supplying the bulletin board with additional null votes. This idea has been proposed in [27] for a receipt-free version of Helios, but it needs to be adjusted to the particularities of our protocol. Generally, ballots containing additional null votes must satisfy two properties: (1) they are indistinguishable from regular ballots; (2) anyone can generate them. To avoid that the bulletin board gets flooded with a large amount of null votes, we propose that only trusted authorities can generate them. This can be realized by combining an additional proof

$$\pi_4 = NIZKP[(\sigma, x) : F = \text{enc}_y^{+}(0, \sigma) \wedge y = h^x]$$

disjunctively with $\pi_{123} = (\pi_1, \pi_2, \pi_3)$. A voter will therefore generate π_{123} and simulate π_4, whereas the trusted

authorities will simulate π_{123} and generate π_4.[9] To connect a ballot containing a null vote to a real ballot from an eligible voter, the trusted authorities copies and re-encrypts $E = \text{enc}_y^{\times}(\hat{u}, \rho)$ from a ballot already published on the bulletin board. The remaining problem then is to decide about the number of submitted null votes. Finding an optimal strategy that maximizes the obfuscation of the total number of votes of a single voter, and therefore minimizes to risk of a vote buying attack based on receipts of this type, is an open question for further research.

Extended homomorphic tallying To avoid the decryption of accumulated votes of individual voters, which is responsible for the Italian attack, we can modify the last steps of the tallying procedure. Instead of performing $\mathbf{V} = \text{dec}_x^{+}(\mathbf{F}'_H)$ and publishing $\mathbf{V}$, the trusted authorities prove for every $F'_H \in \mathbf{F}'_H$ one of the two following proofs:

$$NIZKP[(x) : \text{dec}_x^{+}(F'_H) \in \mathbb{V}],$$

if F'_H contains a valid vote, or

$$NIZKP[(x) : \text{dec}_x^{+}(F'_H) \notin \mathbb{V}],$$

if F'_H contains an invalid vote. The list of these proofs is published together with $\mathbf{F}'_H$. To obtain the final election results, all values F'_H containing a valid vote are accumulated under encryption and the resulting sum of votes is decrypted in one single step.[10] The above proofs can be constructed in different ways, for example by decrypting F'_H into v, encrypting v with a fresh randomization into F''_H, proving the plaintext equivalence of F'_H and F''_H, and demonstrating the validity (or invalidity) of the vote in F''_H using methods from [22].

Multiple elections If the protocol as presented so far is used for multiple elections, but without requiring voters to renew their credentials, then a future adversary will be able to link the votes from the same voter by finding pairs (α, β) that match with election credentials from different elections. This does not provide a direct link to the voters' identities, but it allows creating voter profiles which will eventually leak information. To overcome this problem, the protocol must be modified to ensure that a pair (α, β) is used for only one election. This can be achieved by extending the private and public credentials to $(\alpha, \beta_1, \ldots, \beta_L, \gamma)$ and $u = h_1^{\alpha} h_{2,1}^{\beta_1} \ldots h_{2,L}^{\beta_L} h_3^{\gamma}$, respectively, where L is the maximal number of elections the credentials can be used for. The corresponding commitment to the extended private credential, $D = \text{com}_q(\alpha, \beta_1, \ldots, \beta_L, \gamma, s)$, implies that π_2 needs to be extended to a representation proof of size $L + 1$. Finally, a modified election credential $\hat{u} = \hat{h}_1^{\alpha} \hat{h}_2^{\beta_l}$ and a modified proof π_3 are computed for $l = (\varepsilon \bmod L) + 1$, where $\varepsilon = 1, 2, \ldots$ is the *election number* published beforehand by the elections administration.

[9] In case x is shared among multiple authorities, the literal $y = h^x$ in π_4 can be replaced by a disjunction $\bigvee_j (y_j = h^{x_i})$, where x_i is a single private key share of an individual trusted authority and y_j are corresponding public values of all trusted authorities. In this way, null votes can be generated individually by a single trusted authority.

[10] An additive vote encoding capable of representing all possible election results is necessary for this.

Table 1 Ballot size as a function of M and K. Elements of $\mathbb{Z}_p$ and $\mathbb{G}_q$ are counted together

Ballot component	Elements of $\mathscr{G}_p$	Elements of $\mathbb{Z}_p$ or $\mathbb{G}_q$	Elements of $\mathbb{Z}_q$
C, D	1	1	–
E, F	–	4	–
π_1	$4\lfloor \log M \rfloor + 2$	$3\lfloor \log M \rfloor + 3$	–
π_2	$K+1$	$2K+2$	$4K$
π_3	–	5	7
Entire ballot	$4\lfloor \log M \rfloor + K + 4$	$3\lfloor \log M \rfloor + 2K + 15$	$4K+7$

4 Performance and implementation

Given the complexity of generating and verifying the necessary set membership, representation, and shuffle proofs, we need to look closely at the computational resources required by our voting protocol. We will first analyze the generation of a single ballot during vote casting. Then, we will determine the cost of the tallying procedure, and finally examine the verification of an entire election. The subject of our analysis is the basic protocol version from Section 3.2 without any extension.

4.1 Vote casting

The size and the cost of generating of a ballot in our protocol is mainly determined by the sizes of π_1 and π_2. Note that π_1 depends on the number of eligible voters M, whereas π_2 depends on a security parameter K.[11] In Table 1, we recapitulate the number of group elements for $\mathscr{G}_p$, $\mathbb{Z}_p$, $\mathbb{G}_q$, and $\mathbb{Z}_q$ and sum them up for an entire ballot. Since $\mathbb{Z}_p$ and $\mathbb{G}_q$ share the same modulo p, their elements are counted together. Similarly, Table 2 recapitulates and sums up the number of exponentiations in $\mathscr{G}_p$ and $\mathbb{G}_q$ and multiplications in $\mathbb{Z}_p$ required for generating a single ballot.

Compared to the complexity analysis for the predecessor protocol as presented in [28], the results given here are very similar. The only differences come from the encryptions E and F and the proof π_3, which require a few more

[11] The security parameter K determines the soundness of the proof. We adopt the recommendation of $K \geq 80$ from [4].

Table 2 Number of exponentiations and multiplications required to generate a single ballot

Ballot component	Exponentiations in $\mathscr{G}_p$	Exponentiations in $\mathbb{G}_q$	Multiplications in $\mathbb{Z}_p$
C, D	2	4	–
E, F	–	7	–
π_1	$8\lfloor \log M \rfloor + 4$	–	$2M\lfloor \log M \rfloor$
π_2	$2K+2$	$4K$	–
π_3	–	11	–
Entire ballot	$8\lfloor \log M \rfloor + 2K + 8$	$4K+22$	$2M\lfloor \log M \rfloor$

exponentiations in $\mathbb{G}_q$ and corresponding group elements. Since K will usually be a value ≥ 80, the estimated ballot sizes and running times given in [28, Table 2 and 4] do not change much (for example ten additional exponentiations in $\mathbb{G}_q$). We will therefore not repeat the discussion of the analysis. We only conclude that we expect our protocol to work reasonably well except for a a very large electorate.

4.2 Tallying

The tallying procedure is what makes this protocol more complex compared to its predecessor. To analyze its complexity, let $N \geq N'$ be the total number of submitted ballots and $N' \leq M$ the number of eligible voters submitting at least one ballot, i.e, N/N' is the average number of ballots per voter and N'/M the voter turnout. We assume that all submitted ballots contain valid proofs, which implies $N = |\mathbf{EF}|$ and $N' = |\mathbf{F}_H|$. In Table 3, we show the total number of group elements generated during tallying and the number of necessary exponentiations in $\mathbb{G}_q$. We assume that the two shuffle proofs π_E and π_F are generated using Wikstrm's method [37, 38]. The results are given for a single authority. In case of multiple authorities, the numbers need to be multiplied accordingly.

Together with the results for the ballot size in Table 1, which can be multiplied by N to obtain the size of $\mathbf{B}$, we can calculate an estimation of the size of the election data published on the bulletin board. Note that the lists $\mathbf{EF}$, $\mathbf{E}'$, $\mathbf{F}_H$, and $\mathbf{V}'$ need not to be stored explicitly (see footnote on 8). To simplify the setting, we assume that every eligible voter submits one single ballot, which implies $M = N = N'$. Furthermore, to allow comparison with the results given in [28], we adopt the security parameters $K = 80$, $|p| = 1024$, and $|q| = 160$. The results of this calculation are given in Table 4.

Table 4 Size of the election data for different numbers of voters and parameters $K = 80$, $|p| = 1024$, and $|q| = 160$, and assuming that $M = N = N'$

Eligible voters	Elements of $\mathscr{G}_p$	Elements of $\mathbb{Z}_p$, $\mathbb{G}_q$	Elements of $\mathbb{Z}_q$	Total size
10	$1.10 \cdot 10^3$	$1.92 \cdot 10^3$	$3.60 \cdot 10^3$	0.44 MB
100	$1.21 \cdot 10^4$	$2.00 \cdot 10^4$	$3.59 \cdot 10^4$	4.61 MB
1,000	$1.33 \cdot 10^5$	$2.09 \cdot 10^5$	$3.59 \cdot 10^5$	48.60 MB
10,000	$1.49 \cdot 10^6$	$2.21 \cdot 10^6$	$3.59 \cdot 10^6$	0.51 GB
100,000	$1.61 \cdot 10^7$	$2.30 \cdot 10^7$	$3.59 \cdot 10^7$	5.33 GB
1,000,000	$1.77 \cdot 10^8$	$2.42 \cdot 10^8$	$3.59 \cdot 10^8$	56.64 GB

We conclude from the results of Table 4 that the total size of the election data increases only slightly compared to the predecessor protocol. An overhead of 10 % is needed for $M = 10$, but this number gets even smaller when M gets larger, for example less than 7 % for $M = 1{,}000{,}000$. This observation reflects the fact that the tallying procedure requires only $O(N)$ space and therefore contributes much less to the total election data than the $O(N \log M)$ space of the ballots (assuming that K is a constant value), especially when M gets large.

A similar conclusion can be drawn with respect to computation time. Recall from step 2 in Fig. 4 that verifying all ballots is an important part of the tallying procedure. As we will see in Table 5 (upper part), the verification of the proofs in each of the N ballots requires $O(N \log M)$ exponentiations in $\mathscr{G}_p$ and $\mathbb{G}_q$ and $O(MN)$ multiplications in $\mathbb{Z}_p$,

Table 3 Number of group elements and exponentiations required during tallying in case of a single authority and assuming that all ballots are valid

Component	Elements of $\mathbb{G}_q$	Elements of $\mathbb{Z}_q$	Exps. in $\mathbb{G}_q$
$\mathbf{EF}' = \text{shuffle}^{\phi}_{f_{(\delta,\sigma_1),\ldots,(\delta,\sigma_N)}}(\mathbf{EF})$	$4N$	–	$4N$
π_E	$N+8$	$2N+5$	$12N+7$
$\mathbf{H} = \text{dec}^{\times}_x(\mathbf{E}')$	N	–	N
π_H	$2N$	N	$2N$
$\mathbf{F}'_H = \text{shuffle}^{\phi'}_{f_{\sigma'_1,\ldots,\sigma'_{N'}}}(\mathbf{F}_H)$	$2N'$	–	$2N'$
π_F	$N'+6$	$2N'+4$	$8N'+5$
$\mathbf{V} = \text{dec}^{+}_x(\mathbf{F}'_H)$	–	N'	N'
π_V	$2N'$	N'	$2N'$
Total	$8N+5N'+14$	$3N+4N'+9$	$19N+13N'+12$

Tallying (Trusted Authority):

1. Retrieve the list $\mathbf{B}$ of all ballots from the bulletin board.
2. For each $B = (C, D, E, F, \pi_1, \pi_2, \pi_3) \in \mathbf{B}$, verify π_1, π_2, π_3. Select the pairs (E, F) from ballots with valid proofs. Let $\mathbf{EF}$ denote the list of such pairs and $N = |\mathbf{EF}|$ its size.
3. Pick values $\delta, \sigma_1, \ldots, \sigma_N \in_R \mathbb{Z}_q$ and a permutation $\phi \in_R \Phi_N$. Compute $\mathbf{EF}' = \text{shuffle}^{\phi}_{f_{(\delta,\sigma_1),\ldots,(\delta,\sigma_N)}}(\mathbf{EF})$.
4. Select the ciphertexts E' from all pairs $(E', F') \in \mathbf{EF}'$. Let $\mathbf{E}'$ denote the list of such ciphertexts. Compute plaintexts $\mathbf{H} = \text{dec}^{\times}_x(\mathbf{E}')$.
5. For each distinct $H \in \mathbf{H}$, compute $F_H = \prod F'$ using all pairs $(E', F') \in \mathbf{EF}'$ satisfying $H = \text{dec}^{\times}_x(E')$. Let $\mathbf{F}_H$ denote the list of such values and $N' = |\mathbf{F_H}|$ its size.
6. Pick values $\sigma'_1, \ldots, \sigma'_{N'} \in_R \mathbb{Z}_q$ and a permutation $\phi' \in_R \Phi_{N'}$. Compute $\mathbf{F}'_H = \text{shuffle}^{\phi'}_{f_{\sigma'_1,\ldots,\sigma'_{N'}}}(\mathbf{F}_H)$.
7. Compute plaintext votes $\mathbf{V} = \text{dec}^{+}_x(\mathbf{F}'_H)$.
8. Compute valid plaintext votes $\mathbf{V}' = \mathbf{V} \cap \mathbb{V}$.
9. Compute the following non-interactive proofs:

$$\pi_E = NIZKP[(\delta, \sigma_1, \ldots, \sigma_N, \phi) : \mathbf{EF}' = \text{shuffle}^{\phi}_{f_{(\delta,\sigma_1),\ldots,(\delta,\sigma_N)}}(\mathbf{EF})],$$

$$\pi_H = NIZKP[(x) : \mathbf{H} = \text{dec}^{\times}_x(\mathbf{E}') \wedge y = h^x],$$

$$\pi_F = NIZKP[(\sigma'_1, \ldots, \sigma'_{N'}, \phi') : \mathbf{F}'_H = \text{shuffle}^{\phi'}_{f_{\sigma'_1,\ldots,\sigma'_{N'}}}(\mathbf{F}_H)],$$

$$\pi_V = NIZKP[(x) : \mathbf{V} = \text{dec}^{+}_x(\mathbf{F}'_H) \wedge y = h^x].$$

10. Post $(\mathbf{EF}, \mathbf{EF}', \mathbf{E}', \mathbf{H}, \mathbf{F}_H, \mathbf{F}'_H, \mathbf{V}, \mathbf{V}', \pi_E, \pi_H, \pi_F, \pi_V)$ into the designated area of the bulletin board.

Fig. 4 Summary of the tallying phase with a single trusted authority

which is much more expensive than $O(N)$ exponentiations in $\mathbb{G}_q$ required to execute the remaining steps of the tallying procedure. For estimating the running times of verifying all ballots, we refer to [28, Table 6]. For $M = 1{,}000{,}000$, for example, we calculated an negligible 0.1 % overhead for the whole tallying procedure. Therefore, we refer again to the discussion and conclusions given in [28] and do not repeat them here.

Table 5 Number of exponentiations and multiplications required to verify the election data for M eligible voters, N submitted ballots, N' participating voters, security parameter K, and a single trusted authority

Component	Exponentiations in $\mathcal{G}_p$	Exponentiations in $\mathbb{G}_q$	Multiplications in $\mathbb{Z}_p$
π_1	$6N\lfloor \log M \rfloor + 6N$	–	$2NM$
π_2	$2KN + N$	$4KN$	–
π_3	–	$16N$	–
π_E	–	$13N + 15$	–
π_H	–	$4N$	–
π_F	–	$9N' + 11$	–
π_V	–	$4N'$	–
Total	$6N\lfloor \log M \rfloor + 2KN + 7N$	$4KN + 33N + 13N' + 26$	$2NM$

4.3 Verification

Compared to the predecessor protocol from [28], a complete verification of the election data as presented at the end of Section 3.2 requires a number of steps in addition to the verification of the proofs contained in the submitted ballots. Table 5 summarizes the number of exponentiations in $\mathcal{G}_p$ and $\mathbb{G}_q$ and the number of multiplications in $\mathbb{Z}_p$. Note that π_1 requires a linear number of multiplications, which cannot be neglected when M gets very large [6].

Again, it turns out that the additional work to verify the proofs generated during tallying is marginal compared to the verification of the ballots. In terms of asymptotic running times, $O(N)$ exponentiations in $\mathbb{G}_q$ are required for verifying the proofs from the tallying procedure, whereas $O(N \log M)$ exponentiations in $\mathcal{G}_p$ and $\mathbb{G}_q$ and $O(NM)$ multiplications in $\mathbb{Z}_p$ are required for verifying the ballots. For example, we calculated that the overall verification takes only 3 % longer for $M = 10$ and less than 0.05 % longer for $M = 1{,}000{,}000$. In [28, Table 6], we estimated the time to verify the proofs included in 1,000,000 ballots on a single ordinary machine to be more than 4000 h, which seems to be at the limit of what is feasible today, possibly with multiple and more powerful machines. The same conclusion holds for the extended protocol from this paper.

5 Conclusion

In this paper, we introduced the first practical cryptographic voting protocol offering everlasting vote privacy and receipt-freeness simultaneously. Everlasting privacy is realized with perfectly hiding commitments and zero-knowledge proofs of knowledge, and hence does not depend on trusted authorities or computational intractability assumptions. Receipt-freeness, on the other hand, is achieved by a combination of cryptographic mixing and homomorphic tallying, for which trusted authorities and computational intractability assumptions are obviously required. These characteristics of our protocol exclude vote buying attacks, if we assume them to take place at the time of an election and that rewards are paid off shortly afterwards, but not many years later. Attacks against vote privacy will always remain impossible.

We presented the protocol in two steps. For the basic version, which includes all central mechanisms, some restrictions must be applied to both everlasting privacy and receipt-freeness. To eliminate these problems, we proposed corresponding protocol extensions, which can be added individually or jointly. The protocol is captivating in its

relatively simple vote casting procedure, but the complex tallying and verification procedures—especially if all proposed extensions are implemented—might be subject of further research.

To check if our protocol is practicable for real-world elections, we analyzed the computational resources in terms of memory space consumption and computation time. Compared to the predecessor protocol, it turned out that the overhead for the extended tallying procedure is marginal. The results of the analysis and the conclusions are therefore very similar to those given in [28]. For a medium-sized or even a large electorate (up to approximately 1 million voters), our protocol is feasible with today's technology. An even larger electorate can always be divided into smaller partitions without severe consequences.

Some problems remain unsolved in the current version of our protocol. First, some aspects of coercion-resistance are not addressed, for example forced-abstention, impersonation, or randomization attacks. Another open issue is the problem of flooding the bulletin board with a very large number of valid ballots. This problem is a direct consequence of our mechanism of achieving receipt-freeness by allowing voters to vote multiple times and consider them all in the final tally. Finally, our protocol provides no solution for the problem of a malicious voting platform. Enhancing our protocol with existing solutions to the open problems is another subject for future research.

Acknowledgments Research supported by the Swiss National Science Foundation (project No. 200021L_140650).

References

1. Arapinis M, Cortier V, Kremer S, Ryan M (2013) Practical everlasting privacy. In: Basin D, Mitchell J (eds) POST'13, 2nd conference on principles of security and trust, LNCS 7796, Rome, pp 21–40
2. Araújo R, Foulle S, Traoré J (2007) A practical and secure coercion-resistant scheme for remote elections. In: Chaum D, Kutylowski M, Rivest RL, Ryan PYA (eds) FEE'07, Workshop on frontiers in electronic elections. Schloss Dagstuhl, Germany, pp 330–342
3. Araújo R, Foulle S, Traoré J (2010) A practical and secure coercion-resistant scheme for internet voting. In: Chaum D, Jakobsson M, Rivest R, Ryan PYA, Benaloh J, Kutylowski M, Adida B (eds) Towards trustworthy elections: new directions in electronic voting, LNCS 6000. Springer, pp 330–342
4. Au MH, Susilo W, Mu Y (2010) Proof-of-knowledge of representation of committed value and its applications. In: Steinfeld R, Hawkes P (eds) ACISP'10, 15th Australasian conference on information security and privacy, LNCS 6168, Sydney, pp 352–369
5. Bayer S, Groth J (2012) Efficient zero-knowledge argument for correctness of a shuffle. In: Pointcheval D, Johansson T (eds) EUROCRYPT'12, 31st annual international conference on theory and applications of cryptographic techniques, LNCS 7237, Cambridge, pp 263–280
6. Bayer S, Groth J (2013) Zero-knowledge argument for polynomial evaluation with application to blacklists. In: Johansson T, Nguyen PQ (eds) EUROCRYPT'13, 32nd annual international conference on the theory and applications of cryptographic techniques, LNCS 7881, Athens, pp 646–663
7. Benaloh J, Tuinstra D (1994) Receipt-free secret-ballot elections. In: STOC'94, 26th Annual ACM symposium on theory of computing. Montréal, pp 544–553
8. Brands S (2000) Rethinking public key infrastructures and digital certificates: building in privacy. MIT Press
9. Brands S, Demuynck L, De Decker B (2007) A practical system for globally revoking the unlinkable pseudonyms of unknown users. In: Pieprzyk J, Ghodosi H, Dawson E (eds) ACISP'07, 12th Australasian conference on information security and privacy, LNCS 4586, Townsville, pp 400–415
10. Buchmann J, Demirel D, van de Graaf J (2013) Towards a publicly-verifiable mix-net providing everlasting privacy. In: Sadeghi AR (ed) FC'13, 17th International conference on financial cryptography, LNCS 7859, Okinawa, pp 197–204
11. Camenisch J, Chaabouni R, Shelat A (2008) Efficient protocols for set membership and range proofs. In: Pieprzyk J (ed) ASIACRYPT'08, 14th International conference on the theory and application of cryptology and information security, LNCS 5350, Melbourne, pp 234–252
12. Camenisch J, Stadler M (1997) Efficient group signature schemes for large groups. In: Kaliski BS Jr (ed) CRYPTO'97, 17th Annual international cryptology conference on advances in cryptology, LNCS 1294, Santa Barbara, pp 410–424
13. Canard S, Traoré J (2003) List signature schemes and application to electronic voting. In: Augot D, Charpin P, Kabatianski G (eds) WCC'03, 3rd International workshop on coding and cryptography, Versailles, pp 81–90
14. Chaum D (1988) The dining cryptographers problem: unconditional sender and recipient untraceability. J Cryptol 1(1):65–75
15. Clark J, Hengartner U (2011) Selections: internet voting with over-the-shoulder coercion-resistance. In: Danezis G (ed) FC'11, 15th International conference on financial cryptography, LNCS 7035, St. Lucia, pp 47–61
16. Cranor LF, Cytron RK (1996) Design and implementation of a practical security-conscious electronic polling system. Tech. Rep. WUCS-96-02. Washington University
17. Cuvelier E, Pereira O, Peters T (2013) Election verifiability or ballot privacy : Do we need to choose? In: Crampton J, Jajodia S, Mayes K (eds) ESORICS'13, 18th European conference on research in computer security, LNCS 8134, Egham, pp 481–498
18. Demirel D, Henning M, van de Graaf J, Ryan PYA, Buchmann (2013) Prêt à Voter providing everlasting privacy. In: Heather J, Schneider S, Teague V (eds) VoteID'13, 4th International conference on e-voting and identity, LNCS 7985, Guildford, pp 156–175
19. Demirel D, van de Graaf J, Araújo R (2012) Improving Helios with everlasting privacy towards the public. In: Halderman JA, Pereira O (eds) EVT/WOTE'12, Electronic voting technology workshop/workshop on trustworthy elections, Bellevue
20. Fiat A, Shamir A (1986) How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko AM (ed) CRYPTO'86, 6th Annual international cryptology conference on advances in cryptology, Santa Barbara, pp 186–194
21. Fujioka A, Okamoto T, Ohta K (1992) A practical secret voting scheme for large scale elections. In: Seberry J, Zheng Y (eds) ASIACRYPT'92, Workshop on the theory and application of cryptographic techniques, LNCS 718, Gold Coast, pp 244–251
22. Groth J (2005) Non-interactive zero-knowledge arguments for voting. In: Ioannidis J, Keromytis A, Yung M (eds) ACNS'05, 3th International conference on applied cryptography and network security, LNCS 3531, New York, pp 467–482

23. Haenni R, Koenig RE (2013) A generic approach to prevent board flooding attacks in coercion-resistant electronic voting schemes. Comput Secur 33:59–69
24. Hirt M, Sako K (2000) Efficient receipt-free voting based on homomorphic encryption. In: Goos G, Hartmanis J, van Leeuwen J (eds) EUROCRYPT'00, 19th International conference on the theory and applications of cryptographic techniques, LNCS 1807, Bruges, pp 539–556
25. Juang WS, Liaw HT (2004) Fair blind threshold signatures in wallet with observers. J Syst Softw 72(1):25–31
26. Juels A, Catalano D, Jakobsson M (2005) Coercion-resistant electronic elections. In: Atluri V, De Capitani di Vimercati S, Dingledine R (eds) WPES'05, 4th ACM workshop on privacy in the electronic society, Alexandria, pp 61–70
27. Kulyk O, Teague V, Volkamer M (2015) Extending helios towards private eligibility verifiability. In: Haenni R, Koenig RE, Wikström D (eds) VoteID'15, 5th International conference on e-voting and identity, LNCS 9269, Bern, pp 57–73
28. Locher P, Haenni R (2015) Verifiable internet elections with everlasting privacy and minimal trust. In: Haenni R, Koenig RE, Wikström D (eds) VoteID'15, 5th International conference on e-voting and identity, LNCS 9269, Bern, pp 74–91
29. Moran T, Naor M (2006) Receipt-free universally-verifiable voting with everlasting privacy. In: Dwork C (ed) CRYPTO'06, 26th Annual international cryptology conference on advances in cryptology, LNCS 4117, Santa Barbara, pp 373–392
30. Moran T, Naor M (2007) Split-ballot voting: everlasting privacy with distributed trust. In: Ning P, de Capitani di Vimercati S, Syverson P (eds) CCS'07, 14th ACM conference on computer and communications security, Alexandria, pp 246–255
31. Moran T, Naor M (2010) Split-ballot voting: everlasting privacy with distributed trust. ACM Trans Inf Syst Secur 13(2):16:1–16:43
32. Okamoto T (1997) Receipt-free electronic voting schemes for large scale elections. In: Christianson B, Crispo B, Lomas TMA, Roe M (eds) 5th International security protocols workshop, LNCS 1361, Paris, pp 25–35
33. Rivest RL, Smith WD (2007) Three voting protocols: ThreeBallot, VAV, and Twin. In: EVT'07, USENIX/ACCURATE Electronic voting technology workshop. Boston
34. Sako K, Kilian J (1995) Receipt-free mix-type voting scheme: a practical solution to the implementation of a voting booth. In: Guillou LC, Quisquater JJ (eds) EUROCRYPT'95, 14th International conference on the theory and applications of cryptographic techniques, LNCS 921, Saint-Malo, pp 393–403
35. Schläpfer M, Haenni R, Koenig RE, Spycher O (2011) Efficient vote authorization in coercion-resistant internet voting. In: Kiayias A, Lipmaa H (eds) VoteID'11, 3rd International conference on e-voting and identity, LNCS 7187, Tallinn, pp 71–88
36. Spycher O, Koenig RE, Haenni R, Schläpfer M (2011) A new approach towards coercion-resistant remote e-voting in linear time. In: Danezis G (ed) FC'11, 15th International conference on financial cryptography, LNCS 7035, St. Lucia, pp 182–189
37. Terelius B, Wikström D (2010) Proofs of restricted shuffles. In: Bernstein DJ, Lange T (eds) AFRICACRYPT'10, 3rd International conference on cryptology in Africa, LNCS 6055, Stellenbosch, pp 100–113
38. Wikström D (2009) A commitment-consistent proof of a shuffle. In: Boyd C, González Nieto J (eds) ACISP'09, 14th Australasian conference on information security and privacy, LNCS 5594, Brisbane, pp 407–421
39. Xia Z, Schneider S (2006) A new receipt-free e-voting scheme based on blind signature. In: WOTE'06, IAVoSS Workshop on trustworthy elections. Cambridge, pp 127–135