Secure Internet Voting on Limited Devices with Anonymized DSA Public Keys

Rolf Haenni
Bern University of Applied Sciences
CH-2501 Biel, Switzerland

Oliver Spycher
Bern University of Applied Sciences
CH-2501 Biel, Switzerland
University of Fribourg
CH-1700 Fribourg, Switzerland

Abstract

This paper presents an Internet voting protocol, which is primarily designed for limited voting devices such as smart cards or mobile phones. The minimum requirement for these devices is the ability to compute one ElGamal encryption and one DSA signature in reasonable time. The protocol is based on an anonymization mechanism for DSA public keys, which can be installed on top of an existing DSA public-key infrastructure for each individual voting event. The protocol protects the privacy of the voters and offers public verifiability, but it is not receipt-free or coercion-resistant. As a counter-measure against vote buying or coercion attacks, it can be used as the electronic component of a hybrid voting system.

1 Introduction

The challenge of building secure Internet voting systems has attracted a great deal of attention among researchers in applied cryptography. Over the years, numerous voting protocols have been proposed in the literature. While the number of security properties of these protocols has increased steadily over the years, new requirements have been added to the list of desirable security features. Although an impressive security level has been reached today by increasingly complex voting protocols, the all-embracing "perfect" voting system is still missing. Some of the most important open problems are *long-term security* (What if today's cryptography becomes breakable in the future?), the *insecure platform* (What if the voter's computer is compromised by malware?), and *voter coercion* (What if voters are forced to vote in a particular way?). A certain family of protocols is particularly designed to address the coercion problem [3, 9, 25, 34], but due to the complexity of the involved cryptography, they are relatively inefficient and therefore not yet applicable for large-scale political elections. Another family of protocols is designed for electronic voting at protected polling stations [7, 14, 29], but these ideas are not directly transferable to conduct elections over the Internet.

1.1 Contribution

The Internet voting protocol presented in this paper does not directly address the above-mentioned security problems, it rather focuses on making the actual voting process as slim as possible. We assume thus the availability of enough computational power *before* and *after* the official voting period, whereas only minimal performance is assumed with regard to the voter's computational device, which runs some software for casting the votes. This seems to be a reasonable standpoint: while computational power for preparing and tallying an election can easily be scaled up to the actual size of the electorate and to meet the resulting computational needs, we must usually take it for granted that voters are equipped with devices of limited performance (e.g., smart cards, mobile phones, or older Javascript-based web browsers), and that they are not willing to wait for more than a few seconds to complete the vote casting process.

To meet the above requirement of a lightweight vote-casting client, we propose a protocol which involves on the voter's side a single ElGamal encryption and a single DSA signature. In more technical terms, only four modular exponentiations must be computed: two for the encryption, one for the digital signature, and one for another purpose. To avoid that the signature creates a direct link from the vote back to the voter, the protocol involves an anonymization mechanism for DSA public keys during the election preparation phase. The trick is to shuffle the public keys of the electorate while simultaneously replacing the generator of the underlying cyclic group [26, 32]. This mechanism can be installed on top of an existing DSA public-key infrastructure for each individual voting event. The protocol may therefore be considered to be applied in combination with existing eID cards that provide DSA key pairs for online authentication or

digital signatures.

Compared to existing protocols with similar properties (e.g., schemes based on homomorphic tallying [10, 21] or mix-nets [5, 23, 30]) and corresponding implementations (e.g., *Helios* [1, 2]), the novelty and main benefit of our approach is the possibility for voters to remain fully anonymous. This means that not only the content of somebody's vote remains secret, but also the existence of somebody's vote. This is a critical privacy property, which is needed to guarantee the fairness of the voting system. If a system provides the information on *who* already voted to the electorate, then preliminary conclusions on the expected turnout of certain electoral subgroups (members of political parties, employees of companies, voters of a given age or sex, etc.) can be drawn during the voting period. This could therefore influence the final result of the election.

1.2 Structure of Paper

In Section 2, we show the requirements on remote electronic voting systems as they are commonly postulated. We also describe the concept of a hybrid voting system, which may serve as a general counter-measure against the above-mentioned coercion problem. In Section 3, we introduce the voting protocol stepwise. We start with the above-mentioned anonymization mechanism for DSA public keys. Next, we describe a core version of the protocol that contains all indispensable ingredients. Finally, we propose three additional protocol components, which can be added independently to the core version. One of the additional components increases the number of modular exponentiations on the voter's side from four to five. We describe the security features offered by our protocol and the proposed extensions and compare them with two related protocols. In Section 4, we give some background information on the Selectio Helvetica system, an implementation of the protocol as a web application with a Javascript-based vote-casting client. We describe some practical constraints of this particular setting and outline possible solutions. We also report on our experience with the *Baloti.ch* voting platform, which uses Selectio Helvetica under the hood. In Section 5, we conclude the paper with some final remarks and an outlook to future work.

2 Internet Voting

An idealized voting systems guarantees the correctness of the election result and the secrecy of the vote under all possible circumstances. In traditional paper-based voting systems, correctness is established by supervising every single step of the voting and tallying procedures by independent observers. The only unsupervised step is the actual voting act, which may take place in a private voting booth to fully assure the secrecy of the vote. Both the supervision by independent observers and the privacy of the voting booth are simple and effective mechanisms, but they are hard to transfer into the context of remote electronic elections over the Internet. Nevertheless, *correctness* (together with *verifiability*) and *privacy* are widely accepted to be two of the most essential security requirements for electronic voting systems.

2.1 Security Requirements

The following non-exhaustive list defines some of the most important security requirements informally. We refer to [24] for a more systematic discussion and for further details.

Correctness. The final election result reflects exactly the electorate's choice. This embraces the following sub-requirements:

a) *Democracy.* Only eligible voters can vote, and each eligible voter can cast at most one vote that counts.

b) *Integrity.* After casting, votes cannot be altered, deleted, or substituted.

c) *Accuracy.* All valid votes are counted, invalid votes are not counted.

All aspects of correctness are crucial for an electronic voting system, because attacks are inherently more scalable than in traditional voting systems.

Privacy. No one can obtain more information about the votes and the participating voters than what can be inferred from the final election result. This embraces the following sub-requirements:

a) *Secrecy.* No one can tell how a particular voter or any possible subgroup of voters actually voted.

b) *Anonymity.* No one can tell who actually voted.

c) *Receipt-Freeness.* No one can prove to someone else (e.g., by handing over a voting receipt) whether and how he or she voted.

d) *Fairness.* No one can infer partial results before the election is closed.

Privacy with all its aspects is an essential integrity safeguard, because it allows voters to cast their votes in full independence.

Verifiability. The correctness of the final election result (which involves every single aspect in the above definition of correctness) can be verified. There are two different forms of verifiability:

- *Universal Verifiability.* Anyone (including the voter themselves) can verify the correctness of the election result.
- *Delegated Verifiability.* The verification task is delegated to a group of independent auditors.

A weaker form of verifiability, sometimes called *individual verifiability*, allows voters to check the inclusion of their votes in the final election result, but not the other aspects of correctness.

Robustness. A small set of broken, unavailable, or corrupt system components or a small group of conspiring parties (election authorities, system administrators, voters, external attackers, etc.) cannot disrupt the election process or compromise correctness or privacy.

Coercion-Resistance. No one can urge voters (neither by offering them a reward nor by intimidation) to vote in a particular way, to vote at random, not to vote at all, or to give away their private keying material. This implies that a coercer cannot decide whether a voter complies with the demands [25].

The protocol presented in this paper offers a solution to all of the above requirements, except for receipt-freeness and coercion-resistance. Therefore, we do not directly address the vote buying or voter coercion problems, but since the protocol contains all the prerequisites of a *hybrid voting system* [33], we offer at least an indirect solution.

2.2 Hybrid Voting Systems

A hybrid voting systems integrates a traditional and an electronic voting system and combines them with a vote revocation mechanism, which allows voters to overrule their electronic votes in the protecting environment of a polling station. Potential vote buyers or coercers must thus assume that corrupted voters will usually revoke their electronic votes. This clearly increases the price of a successful vote buying or coercion attack considerably.

For an e-voting protocol to be used as the electronic voting channel of a hybrid system, it needs to comply with two requirements [33]. First, registered voters that abstained from casting an electronic vote need to be able to unambiguously prove to the voting officials that they are still eligible for casting their vote. Second, if their electronic vote has been cast, voters need to be able to prove ownership of their electronic vote in the electronic ballot-box. The protocol presented in this paper satisfies these requirements. In particular, it guarantees that all eligible voters own respective *vote identifiers* for their votes, even if someone else has voted on their behalf. This is a subtle difference to other e-voting systems, which only guarantee that the parties who actually voted are in possession of vote identifiers (or full receipts). Thus, these systems do not meet the above requirements of a hybrid system.

Two different revocation procedures are given in [33]. Since the protocol that we present here guarantees vote identifiers to vote owners, but not full receipts, revocation is restricted to Procedure 2 of [33]. The general idea of this procedure is to have two electronic ballot-boxes, one for the votes and one for the revocations, and one physical ballot-box for the paper votes. To allow universal verifiability, both electronic ballot-boxes are realized by public bulletin boards. At the end of the voting period, each of the three ballot-boxes is counted independently, and the final result for a given candidate is computed by subtracting the candidate's revocations from the sum of the candidate's electronic and paper votes.

To preserve the correctness of the result, each revocation must exactly match with one electronic vote, but the link between them must be hidden. Procedure 2 in [33] suggests that voters re-encrypt their electronic votes, for which they have proven ownership using their vote identifiers, and that they prove towards the voting officials in the polling station the correctness of the re-encryption (in zero-knowledge and in a designated way). Upon successful verification, the officials generate a (multi-)signature for the re-encrypted vote. The re-encrypted vote together with the signature is then posted as a valid revocation onto the bulletin board, and the voter is granted access to the physical ballot-box for casting the final vote on paper. To prevent voters from revoking the same vote multiple times, the voting officials at the polling station need to keep track of their names (or of the votes they revoked). To protect the vote secrecy of those who revoked their electronic vote, the revocations need to be mixed in a re-encryption mix-net before decrypting them for the final tally (except in the case of a homomorphic tallying procedure).

While most of the security properties of the electronic system are maintained in the hybrid version, it creates some two minor problems. First, it is possible for a group of conspiring voting officials to post additional votes with valid signatures to the electronic ballot-box containing the revocations. By doing so, they cannot directly violate the integrity of somebody's electronic vote, but they can introduce additional votes, which will be subtracted from the final result. It is also possible for an official to divulge the links between votes and corresponding revo-

cations, which means that coercions-resistance in the hybrid system depends on the discretion of the voting officials. Since they need to be trustworthy anyway to properly run the traditional voting procedure, this appears to be a minor problem.

3 Protocol Description

In this section, we introduce our protocol proposal and discuss its properties. To facilitate an unfamiliar reader's first approach, we start by discussing some elementary cryptographic building blocks. A core component and the protocol itself are discussed next. We will then analyze the protocol's security properties and propose two possible enhancements. We conclude this section by discussing the relation to two existing protocols.

3.1 Cryptographic Building Blocks

The protocol assumes several modern cryptographic building blocks. Apart from standard ElGamal encryption/decryption, we also need threshold cryptosystems, non-interactive zero-knowledge proofs of knowledge, anonymous authentication, mix-nets, and anonymous channels. Some of these building blocks will be briefly described below.

ElGamal Cryptosystem. The *ElGamal cryptosystem* is based on a multiplicative cyclic group $(G,\cdot)$ of prime order q, for which the computational and the decisional Diffie-Hellman assumptions are believed to hold. The most common choice for such a group is a subgroup $G_q = \langle g \rangle \subseteq \mathbb{Z}_p^*$ of order $q|p-1$, where p and q are large primes (so-called *safe primes*). The public parameters of an ElGamal cryptosystem are thus p, q, and the generator g of G_q. An ElGamal key pair is a tuple (x,y), where the $x \in_R \mathbb{Z}_q$ is the randomly chosen private key and $y = g^x \in G_q$ the corresponding public key. If $m \in G_q$ denotes the message to encrypt, then the pair $(a,b) = \mathsf{Encrypt}_y(m,r) = (g^r, m \cdot y^r)$ is the ElGamal encryption of m with randomness $r \in_R \mathbb{Z}_q$. For a given encryption $e = (a,b)$, m can be recovered by computing $m = \mathsf{Decrypt}_x(e) = a^{-x} \cdot b$. Note that the ElGamal encryption function is *homomorphic* with respect to multiplication, which means that ciphertexts can be multiplied to get an encryption of the product of respective plaintext.

Threshold Cryptosystems. A cryptosystem such as ElGamal is called *threshold cryptosystem*, if the private decryption key is shared among n parties, and if the number of parties required to cooperate in the decryption protocol exceeds a certain threshold $t \leq n$. A threshold version of the ElGamal cryptosystem results from sharing the private decryption key using Shamir's secret sharing scheme [31]. To avoid the need for a trusted third party to generate the private key shares, it is possible to let the n parties execute a *distributed key generation protocol* [17].

Plaintext Equivalence Test. A distributed *plaintext equivalence test* (PET) in a threshold cryptosystem checks if two ciphertexts are encryptions of the same plaintexts, but without revealing any information about the two plaintexts [22]. For two ElGamal encryptions $e_1 = (g^{r_1}, m_1 \cdot y^{r_1})$ and $e_2 = (g^{r_2}, m_2 \cdot y^{r_2})$, a simple PET consists in checking if e_1/e_2 is an encryption of 1. Simply decrypting e_1/e_2 into m_1/m_2 would obviously reveal information about the plaintexts, but not if alternatively $(e_1/e_2)^z$ is decrypted into $(m_1/m_2)^z$ for some secret blinding value $z \in_R \mathbb{Z}_q$ shared by the holders of the private decryption key x. $PET_{x,z}(e_1,e_2) \in \{true, false\}$ denotes the application of a plaintext equivalence test for two encryptions e_1 and e_2.

DSA Signatures. The *Digital Signature Algorithm* (DSA) is a widely used US Federal Government standard for digital signatures. The public parameters (p,q,g) and the key pairs (x,y) are identical to an ElGamal cryptosystem. If m denotes the message to sign and $H(m) \in \mathbb{Z}_q$ a cryptographic hash code of m, then a DSA signature of m is a pair $(a,b) = \mathsf{Sign}_x(m,r) \in G_q \times G_q$ with

$$a = (g^r \bmod p) \bmod q,$$
$$b = (H(m) + a \cdot x) \cdot r^{-1} \bmod q,$$

and randomness $r \in_R \mathbb{Z}_q$. A given signature $s = (a,b)$ can be verified by checking if the equation $a = (g^u \cdot y^v \bmod p) \bmod q$ holds for $u = H(m) \cdot b^{-1} \bmod q$ and $v = a \cdot b^{-1} \bmod q$. The signature verification is denoted by $\mathsf{Verify}_y(s,m) \in \{true, false\}$.

Zero-Knowledge Proofs of Knowledge. A zero-knowledge proof (ZKP) allows a party to demonstrate to another party that a mathematical statement is true, but without revealing anything other than the truth of the statement itself. A particular class of zero-knowledge proofs are *proofs of knowledge*, in which the prover demonstrates knowledge of the preimage $x \in X$ of a public value $y = \phi(x) \in Y$, where $\phi : X \to Y$ is a candidate one-way function. Such proofs can be constructed as interactive Σ-*protocols*, if ϕ is a homomorphism with a finite domain X [4]. Two of the simplest and most frequently used instances of such Σ-protocols are the proof of knowledge of a discrete logarithm $y = \log_g x$ in a multiplicative group $G_q = \langle g \rangle$ of finite order q, and similarly, the proof of equality of two discrete logarithms

$y = \log_g x = \log_{g'} x'$, where g' is another generator of G_q. Using the Fiat-Shamir heuristic [13], interactive proof can be turned into non-interactive ones (using the random oracle model).

Anonymous Channels. An anonymous channel hides the correspondence between senders and their messages, i.e., the senders of the messages remain anonymous or untraceable. The most common realization of anonymous channels is based on *mix-nets* [6]. A mix-net consists of a sequence of servers, each of which receives a batch of input messages and produces a batch of output messages in a permuted (mixed) order. One of today's most widely used implementation of an anonymous Internet channel is TOR [11, 18], which allows users to hide their identity while browsing the Web or using other Internet services. As a simple alternative to using such designated systems, people may protect the privacy of their online activities by accessing the Internet from public access points (Internet cafés, public libraries, public WLAN, etc.). The anonymity provided in that case may not be perfect under all possible circumstances, but as long as people avoid entering personal data, it should be acceptable for most purposes.

Public Bulletin Board. A public bulletin board is a broadcast channel with memory. This means that everybody is allowed to append new entries and to read its content, but nobody is allowed to delete or modify existing entries. Such a bulletin board may have the additional functionality of filtering out invalid or double entries, for example by checking the validity of an attached digital signature or proof of knowledge. For robustness, bulletin boards should be replicated [20, 28].

3.2 Shuffling DSA Public Keys

Before turning our attention to the actual e-voting protocol, let us first look at one of its core components. The task of this component is to shuffle a given list of DSA public keys such that the keys in the shuffled list cannot efficiently be linked to the keys in the original list. The goal thus is to create a list of *anonymized public keys* (or simply *anonymous keys*), which can no longer be attributed to individual parties, but which can still be used for verifying DSA signatures. The trick is to replace the generator of the underlying cyclic group.

Formally, suppose that (p,q,g) are the public DSA parameters and let $Y = (y_1,\ldots,y_n) = (g^{x_1},\ldots,g^{x_n})$ be a given list of public keys. If $\pi \in \Sigma_n$ is a permutation selected from the group Σ_n of permutations on $\{1,\ldots,n\}$, then $Y_\pi = (y_{\pi(1)},\ldots,y_{\pi(n)})$ denotes the permuted list of public keys. To unlink the elements of Y_π from Y, consider an additional random value $\alpha \in_R \mathbb{Z}_q$ and let $Y_\pi^\alpha = (y_{\pi(1)}^\alpha,\ldots,y_{\pi(n)}^\alpha)$ be the permuted list of public keys raised to the power of α. Note that each private key x_i together with

$$\hat{y}_i := y_{\pi(i)}^\alpha = (g^{x_i})^\alpha = (g^\alpha)^{x_i}$$

forms a valid DSA key pair $(x_i,\hat{y}_i)$ for the public parameters $(p,q,\hat{g})$, where $\hat{g} := g^\alpha$ denotes the new generator. Therefore, if $\hat{g}$ is used to sign m with private key x_i, then $s = \mathsf{Sign}_{x_i}(m,r)$ can be successfully verified with the anonymous key $\hat{y}_i$. In other words, $\mathsf{Verify}_{\hat{y}_i}(\mathsf{Sign}_{x_i}(m,r),m)$ returns *true*, if $\hat{g}$ is used instead of g to compute Sign and Verify. Note that $\hat{y}_i$ is usually unknown to the verifier, but since knowing x_i is sufficient to compute $\hat{y}_i = \hat{g}^{x_i}$, it can simply be attached to s. The verification then involves a preliminary step of checking whether $\hat{y}_i$ is an element of the permuted list of anonymous keys $\hat{Y} := \{\hat{y}_1,\ldots,\hat{y}_n\} = Y_\pi^\alpha$. The whole procedure can thus be seen as an instance of a *group signature scheme*, which allows each member of a group $\{1,\ldots,n\}$ to anonymously sign a message on behalf of the group [8].

Repeated Shuffling. If a single authority performs the shuffling of the public keys, then links between the members of Y and $\hat{Y}$ can easily be established by the authority itself. Even worse, publishing π would entirely wipe out the anonymity of the scheme. A single shuffling authority constitutes thus a severe single point of failure.

This problem can easily be solved by shuffling the public keys multiple times by independent authorities, so-called *anonymizers*. Suppose that there are $m \geq 2$ anonymizers A_j, $1 \leq j \leq m$, for which $\pi_j \in \Sigma_n$ denotes the selected permutation on $\{1,\ldots,n\}$ and $\alpha_j \in_R \mathbb{Z}_q$ the randomly chosen exponent. Let A_j compute $\hat{Y}_j$ and $\hat{g}_j$ by applying π_j,α_j to $\hat{Y}_{j-1}$ and α_j to g_{j-1}, starting with $\hat{Y}_0 := Y$ and $\hat{g}_0 := g$:

$$\hat{Y}_0 \xrightarrow{\pi_1,\alpha_1} \hat{Y}_1 \xrightarrow{\pi_2,\alpha_2} \hat{Y}_2 \quad \cdots\cdots \quad \xrightarrow{\pi_m,\alpha_m} \hat{Y}_m,$$
$$\hat{g}_0 \xrightarrow{\alpha_1} \hat{g}_1 \xrightarrow{\alpha_2} \hat{g}_2 \quad \cdots\cdots \quad \xrightarrow{\alpha_m} \hat{g}_m.$$

At the end of the shuffling process, we obtain from A_m the list of anonymous keys $\hat{Y} := \hat{Y}_m$ and the new generator $\hat{g} := \hat{g}_m$. Note that the whole repeated shuffling procedure corresponds to applying the composed permutation $\pi = \pi_m \circ \cdots \circ \pi_1$ and the product exponent $\alpha = \prod_{j=1}^m \alpha_j$ to the original list Y and the original generator g. The unlinkability between the two lists Y and $\hat{Y}$ is thus given, if π_j and α_j remain secret for at least one anonymizer (due to the assumed difficulty of the discrete logarithm problem).

Verifiable Shuffling. To guarantee the correctness of the repeated shuffling procedure under all possible circumstances, additional measures must be taken to ensure

that the anonymizers do not deviate from the protocol. In other words, we want the shuffling to be verifiably correct, which means that the anonymizers need to provide additional information to allow their computations to be publicly verifiable.

This problem is similar to *re-encryption mix-nets*, which are designed to transform an input list of (usually ElGamal) ciphertexts $E = \{e_1, \ldots, e_n\}$ into another list of ciphertexts $\hat{E} = \{\hat{e}_1, \ldots, \hat{e}_n\}$ with the same plaintexts in permuted order [6]. To avoid a single point of failure, a mix-net consists of at least two *mix-servers* $M_1, \ldots, M_m$, $m \geq 2$, which perform each a single shuffling and re-encryption step. In the same way as explained above, i.e., starting with $\hat{E}_0 := E$, we get a sequence $\hat{E}_0 \rightarrow \hat{E}_1 \rightarrow \cdots \rightarrow \hat{E}_m$ of encryption lists, from which we obtain the final result $\hat{E} := \hat{E}_m$. Each mix-server M_j, after publishing the output list $\hat{E}_j$, provides a non-interactive zero-knowledge proof ZKP_j of correct shuffling. The exact shape and the efficiency of the zero-knowledge proof depend on the chosen approach [16, 19, 27, 36]. The best techniques available today require between $6n$ and $8n$ modular exponentiations for generating and between $6n$ and $10n$ modular exponentiations for verifying the proof [19]. This is a tremendous improvement over the naïve approach based on general Boolean proof composition techniques, where the resulting proof size is quadratic. Subsequently, we discuss three different approaches that are all based on existing proof techniques from the literature.

Approach 1. We could directly apply any of the existing mix-net techniques to our problem by considering each public key $y_i \in Y$ as a trivial ElGamal encryption $e_i = (1, y_i)$ with randomness 0 and by letting each mix-server M_j additionally raise the input encryptions in $\hat{E}_{j-1}$ and the generator $\hat{g}_{j-1}$ to the power of the randomly chosen exponent $\alpha_j \in_R \mathbb{Z}_q$. An additional zero-knowledge proof ZKP'_j of doing so correctly must be provided together with ZKP_j. At the end of the mixing process, the output encryptions are decrypted into the list $\hat{Y}$ of anonymous keys. For the re-encryption during the mixing process and the decryption at the end, this method requires an additional key pair (x, y) of a homomorphic threshold cryptosystem.

The number of additional modular exponentiations for generating ZKP'_j is $n+1$, i.e., generating the combined proof $ZKP_j \wedge ZKP'_j$ requires $7n+1$ modular exponentiations in the best case, i.e., with the protocol proposed in [19]. We do not give further details on this method, because the computational overhead for dealing with ElGamal encryptions (two values instead of one) and for the final threshold decryption seems not to be appropriate.

Approach 2. A more explicit solution for our problem is described in Neff's paper on *verifiable secret shuffles* [26]. He provides a protocol that solves the so-called *general n-shuffle problem*. In the terminology of our paper, a simplified version of this problem can be stated as follows. Suppose that g, $\hat{g}$, and the values of two sequences $Y = \{y_1, \ldots, y_n\}$ and $\hat{Y} = \{\hat{y}_1, \ldots, \hat{y}_n\}$ are all publicly known values of G_q. The prover knows $\alpha = \log_g \hat{g}$, but α is unknown to the verifier. The prover wants to convince the verifier that there is some permutation $\pi \in \Sigma_n$ such that $\hat{y}_{\pi(i)} = y_i^{\alpha}$ holds for all $1 \leq i \leq n$, but without revealing any information about α or π.

As already pointed out in [26], this is clearly a reformulation of our problem of verifying the correctness of a shuffled list of DSA public keys. Neff's general protocol solves this problem by reducing it to a simpler problem, in which it is assumed that the prover also knows the discrete logarithms $x_i = \log_g y_i$ of the input values y_i.

According to [26], the number of modular exponentiations required to construct the full proof is $8n+5$, and the number of modular exponentiations to verify it is $9n+2$. From an efficiency point of view, Neff's approach is thus comparable to the first approach described above. The drawback of this protocol is that it is a 7-move proof, which is not very practicable. Note that the protocol as presented in [26] contained some flaws, but they have been addressed in [27]. Unfortunately, the focus of [27] is no longer on solving the n-shuffle problem.

Approach 3. Rather than providing a proof of absolute correctness, it may be sufficient if each mix-server provides strong evidence of not having deviated from the protocol. In the mix-net literature, such an alternative approach is known as *randomized partial checking*, or RPC for short [23]. RPC-based mix-nets are exceptionally efficient in comparison with ZKP-based mix-nets. The underlying idea is simple but very effective. After performing the mix, the mix-servers are challenged in revealing a random subset of their input/output relations, but such that the unlinkability remains in place. A corrupt mix-server is then likely to be caught, even if it attempts to tamper only a few outputs. This assures a very high overall probability of correct shuffling.

To realize RPC-based mixing of DSA public keys, every anonymizer applies consecutively two permutations π' and π'' (to simplify the formal notation, we omit the anonymizer's index j). They also select two random exponents α' and α''. This implies that $\pi = \pi'' \circ \pi'$ and $\alpha = \alpha' \cdot \alpha''$ are the anonymizer's secret shuffling parameters. Let $\hat{Y}' := Y_{\pi'}^{\alpha'}$ be the intermediate list and $\hat{Y} := Y_{\pi}^{\alpha}$ the output list of DSA keys obtained from shuffling the input list Y. After publishing $\hat{Y}'$ and $\hat{Y}$, the challenge results from dividing $\hat{Y}'$ randomly into two portions of size $n/2$. Let $I', I'' \subseteq \{1, \ldots, n\}$, $I' \cup I'' = \{1, \ldots, n\}$, be cor-

responding disjoint subsets of indices. The anonymizer must then reveal the values $\pi'^{-1}(i)$ for each $i \in I'$ and $\pi''(i)$ for each $i \in I''$. In other words, either the link "to the left" or the link "to the right" must be revealed, but never both links at the same time. This guarantees the unlinkability between the elements of Y and $\hat{Y}$, and the probability that tampering κ outputs remains unnoticed is $1/2^{\kappa}$.

For each revealed link, the anonymizer must provide a zero-knowledge proof of knowledge of a secret exponent α' or α'', for which the two values match. More precisely, two proofs of equality of $n/2$ discrete logarithms are required, one for the links "to the left" or one for the link "to the right". The generation (and verification) of such a partial proof requires $n/2$ modular exponentiations. Note that two additional exponentiations are needed for proving that $\hat{g} = g^{\alpha' \cdot \alpha''}$, i.e., a total number of $n+2$ modular exponentiations is required for the full proof. This is obviously far better than each of the two exact proofs described above.

3.3 Protocol Description

The security of our protocol depends heavily on the anonymity provided by shuffling the DSA public keys as described in the previous subsection. As we will show now, the remaining elements of the protocol are rather simple. Casting a vote, for example, essentially consists in signing the encrypted candidate choice with the anonymized DSA public key. To determine the final election result, the votes carrying a valid signature are decrypted and the resulting plaintext votes are counted.

The protocol involves an *election authority*, which is responsible for setting up an election (specifying the list of candidates, compiling the electoral register, defining the official voting period, etc.). Other responsibilities are shared among parties from three different groups:

a) The group of eligible *voters*, $\mathscr{V} = \{V_1, \ldots, V_n\}$, none of which are are assumed to be trustworthy.

b) The group of *anonymizer*, $\mathscr{A} = \{A_1, \ldots, A_m\}$, of which at least $t_m \leq m$ are assumed to be trustworthy.

c) The group of *talliers*, $\mathscr{T} = \{T_1, \ldots, T_r\}$, of which at least $t_r \leq r$ are assumed to be trustworthy.

We collectively refer to anonymizers and talliers by the term *trustee*. For the sake of simplicity, we assume the trustees to be individuals, but in reality, they could also be independent organizations. Any intersection of these groups is explicitly allowed. Particularly, voters can work as anonymizers or talliers. We also assume the presence of adversaries and coalitions of adversaries, but without explicitly formalizing them as a group.

We distinguish in our protocol five consecutive steps, of which the first two need not to be repeated for every election. Let us now describe these steps one after another.

1. Setup. During the setup, the election authority defines sufficiently secure ElGamal/DSA parameters (p,q,g). These values are published and can be used across multiple voting events. The election authority also installs a public bulletin board $\mathscr{B}$ as a public communication channel, and an anonymous channel $\mathscr{C}$ for posting the votes to the board. The talliers employ a distributed key generation algorithm to obtain a threshold ElGamal key pair (x,y). The common public key y is published, and the shares of the private key x are kept secret. All values published during the setup can be used across multiple elections.

2. Registration. The election authority sets up a DSA public-key infrastructure (PKI) for potential voters. We do not specify all the details of this step, but it must certainly involve some sort of personal authentication. Upon successful authentication, potential voters are equipped with a DSA key pair (x_i, y_i). The key generation procedure must guarantee that the private key is only known to its owner. Corresponding public-key certificates are published to confirm the binding between the public key and the identity of the person behind it. Note that the same PKI may be used for multiple elections or even for purposes other than voting, i.e., we do not require the registration to involve the verification of someone's eligibility. This allows us to install the protocol on top of an existing DSA or ElGamal PKI, e.g., in countries with existing electronic identity cards.

3. Election Preparation. To prepare an election, the election authority publishes the set C of possible choices, from which voters may choose exactly one value. In multi-candidate elections, we consider C to be set of all admissible combinations of candidates. The election authority must also compile and publish the *electoral register*, which includes the certificates of all eligible voters. Note that the electoral register is specific to a particular election, precinct, or district, and may thus change from election to election. The election authority must also announce the begin and the end of official voting period.

To conclude the election preparation, the DSA public keys of all eligible voters are copied from the certificates in the electoral register. Let $Y = \{y_1, \ldots, y_n\}$ be the resulting list, which is taken by the anonymizers as input to the shuffling procedure of the previous subsection. After performing the shuffling, the anonymized output list

$\hat{Y} = \{\hat{y}_1, \ldots, \hat{y}_n\}$ is published together with the new generator $\hat{g}$.

4. Vote Casting. Let $c_i \in C$ be the preferred choice of voter V_i. To cast c_i as V_i's vote, the following steps need to be performed:

1. Encrypt c_i with the talliers' common public key y and randomness $r_1 \in_R \mathbb{Z}_q$:
$$e_i = \mathsf{Encrypt}_y(c_i, r_1).$$
2. Sign e_i with V_i's private key x_i and randomness $r_2 \in_R \mathbb{Z}_q$:
$$s_i = \mathsf{Sign}_{x_i}(e_i, r_2).$$
3. Compute the anonymous key using the new generator $\hat{g}$:
$$\hat{y}_i = \hat{g}_i^{x_i}.$$
4. Submit the *ballot* $B_i = (e_i, s_i, \hat{y}_i)$ over the anonymous channel $\mathscr{C}$ to the public bulletin board $\mathscr{B}$.

After casting the vote, V_i may want to check if B_j has appeared correctly on the public bulletin board. If both the bulletin board and the anonymous channel are constructed in a robust way, and if the V_i is using a secure platform, this will always be the case. If not, V_i may either try to resubmit B_i or to get into contact with the election authority to solve the problem. Note that the option of resubmitting ballots may provide a desirable feature of the voting system called *re-voting*. Voters are then allowed to change their votes during the voting period (see below).

5. Tallying. The tallying procedure starts as soon as the election is closed. Let $B = (e, s, \hat{y}) \in \mathscr{B}$ be an entry from the public bulletin boards, which has been recorded during the official voting period. To be considered in the final tally, it must satisfy the following conditions:

1. $\hat{y}$ is a valid anonymous key, i.e., $\hat{y} \in \hat{Y}$.
2. s is a valid signature for e, i.e., $\mathsf{Verify}_{\hat{y}}(e, s) = \mathit{true}$.
3. B is the only, the first, or the last valid entry for $\hat{y}$ in $\mathscr{B}$. The actual choice between the first or the last ballot depends on whether re-voting is allowed or not.

Checking these conditions requires only publicly available data. It could therefore be delegated to the public board itself.

The encrypted votes of the remaining ballots satisfying these conditions are finally decrypted by the talliers (using their shares of the private key x and by zero-knowledge providing proofs of correct decryption). The final election result is then determined by counting the resulting plaintext votes $c = \mathsf{Decrypt}_x(e)$ for each $c \in C$. Invalid plaintext votes are dropped.

3.4 Discussion

Despite its simplicity, the protocol described in the previous section possesses some interesting properties. Most importantly, it provides an exceptionally simple and lightweight vote casting procedure, which consists of "standard cryptography" (ElGamal, DSA, modular exponentiation) only. Implementing the voting application is therefore not a great challenge and can be realized with standard cryptographic libraries and easily deployed on devices with limited performance. Note that the number of modular exponentiations is always exactly four (two for the ElGamal encryptions, one for the DSA signature, one for computing the anonymous key), which means that the running time of the voting application is independent of the number of choices $|C|$. This is an important advantage over the family of protocols relying on homomorphic tallying [10, 21], for which proving the validity of the votes is indispensable. Note that the modular exponentiations needed for encrypting the vote and generating the signature are independent of the vote and can therefore be computed ahead of time.

Let us now look at the security properties of the proposed protocol. We will address all the requirements introduced in Subsection 2.1. Our informal arguments are based on typical cryptographic assumptions such as the difficulty of computing discrete logarithms problem (DLP), the decisional Diffie-Hellman assumption (DDH), or the existence of collision-resistant cryptographic hash functions. We also take it for granted, that the security properties provided by the protocol components (ElGamal encryption, DSA signatures, shuffling, anonymous channel, public bulletin board) hold under all possible circumstances. Establishing a more formal security model for the whole voting system and corresponding formal security proofs for the desired properties is still work in progress at the time of writing.

Correctness and Verifiability The protocol offers universal verifiability with respect to all aspects of correctness. a) *Democracy*. The eligibility of the voters can be checked by inspecting the public electoral register. The certificates included in the electoral register guarantee that all public keys belong to eligible voters. The proof of correct shuffling assures then that every anonymous key is linked to a public key from the electoral register and therefore belongs to an eligible voter. Finally, the signatures included in the ballots confirm that each vote belongs to an eligible voter. Multiple ballots from the same eligible voter contain the same anonymous key and are

therefore detected during the tallying phase. b) *Integrity*. The signature included in the ballot assures that votes cannot be altered or substituted in a ballot. Altering or deleting ballots before reaching the bulletin board is possible in principle, but this could easily be detected by the voters themselves by inspecting the board. Missing ballot could easily be re-submitted. Deleting ballots from the board is in conflict with the assumed append-only property of the board. c) *Accuracy*. After decrypting the votes, valid votes can easily be separated from invalid ones, and the tallying of the valid votes can easily be verified by repeating the counting procedure.

Privacy Our protocol also provides a high degree of privacy. The key mechanism for providing privacy is the public key shuffling procedure in the election preparation phase. With receipt-freeness as the only exception, all privacy aspects are guaranteed. a) *Secrecy*. Every plaintext vote is unambiguously linked to an anonymous key, but linking the anonymous key back to its owner is prohibited by the anonymous channel and the unlinkability property of the public key shuffling procedure. The same argument prevents linking a set of plaintext votes to the corresponding group of voters. b) *Anonymity*. To find out whether a particular eligible voter has actually voted, we must again establish a link to an actual plaintext vote. This is impossible for the same reasons given above. c) *Receipt-Freeness*. After casting a vote, the voter possesses three secret values: the private signature key, the randomness of the encryption, and the randomness of the signature. Each of them allows the voter to prove ownership of the ballot on the bulletin board. The private key even allows to prove vote abstention (without revealing the key itself). Therefore, our protocol provides receipts for all possible cases. d) *Fairness*. According to the protocol, votes are decrypted after the official voting period. Under the assumption that at least $t_r \leq r$ talliers follow the protocol, it is impossible to infer partial results before the election is closed.

Robustness The repeated shuffling procedure and the sharing of the decryption key makes our protocol robust against a minority of corrupt anonymizers or talliers. To guarantee full robustness, we need to assume that the anonymous channel and the public bulletin board are themselves constructed in a robust way, for example by replicating any crucial component or data.

Coercion-Resistance The missing receipt-freeness of our protocol inherently prohibits coercion-resistance. The possibility of allowing re-votes may seem to be an appropriate countermeasure against coercion attacks, but secretly casting a re-vote with the same anonymous key could immediately be detected by the coercer.

As a real countermeasure for the missing coercion-resistance property, we propose our protocol to be used as the electronic component of a hybrid voting system (see discussion in Subsection 2.2). The private signature keys serve as guaranteed voter identifiers, which are needed to prove ownership of a particular encrypted vote towards the officials at the polling station. If we assume that revoking the electronic vote in the protecting environment of a polling station remains unobservable to coercers, then the system becomes coercion-resistant.

3.5 Additional Components

As shown in the previous subsection, our protocol satisfies most security requirements to a satisfactory degree. However, there are some minor issues, which do not directly affect the security requirements under the stated assumptions, but which may still be bothersome. Let us discuss three such issues and solve them with corresponding protocol enhancements.

Discarding Invalid Votes. In the protocol as presented in Subsection 3.3, voters may encrypt values different from the available choices in C. After decrypting these votes in the tallying phase, they turn out to be invalid and are thus removed from the final tally. The problem of processing such invalid votes is that it allows voters to use the voting system to broadcast a message to the authorities or even the electorate, for example with the goal of discrediting the system. This would be an attack against the voting system itself, not against the correctness or privacy of the election. To avoid such an attack, we propose the following enhancement of the tallying phase (vote casting remains unchanged).

Each valid choice $c \in C$ is considered as a trivial ElGamal encryption $e_c = (1, c)$ with randomness 0. With $E_C = \{e_c : c \in C\}$ we denote the set of all "encrypted" choices. Before decrypting the votes, the talliers must now perform plaintext equivalence tests between the encrypted votes and the encrypted choices. If $PET_{x,z}(e, e_c) = \mathit{false}$ for all $e_c \in E_C$, then e is obviously an invalid vote and can be discarded without decrypting it.

Note that the presence of a trivial encryption enables a simplified PET. Because $c = \mathsf{Decrypt}_x(e_c)$ is publicly known, it must not be protected. Instead of decrypting $(e/e_c)^z = e^z/e_c^z$, it is thus sufficient to decrypt the nominator e^z (the denominator decrypts into c^z). As a consequence, $PET_{x,z}(e, e_c)$ degenerates into testing $\mathsf{Decrypt}_x(e^z)/c^z = 1$, which can be simplified even further into $\mathsf{Decrypt}_x(e^z) = c^z$. This means that the talliers can pre-compute the values c^z for all $c \in C$. Invalid votes

can then be detected efficiently (in linear time) by verifying if $\mathsf{Decrypt}_x(e^z)$ corresponds with one of these values.

Preventing Vote Duplication. In the protocol as presented in Subsection 3.3, nothing prevents voters from duplicating encrypted votes from the public board and submitting them as their own votes. In other words, the system allows voters to submit unknown votes, which is certainly an unpleasant feature. Such cases may even be undetectable, if the voters re-encrypt the encrypted votes before submitting them. To solve this problem, we propose the following protocol enhancement, which makes the vote casting process slightly more expensive.

Duplicating somebody else's vote means that the encryption randomness is unknown. To avoid vote duplication, the ballot must thus be extended with a non-interactive zero-knowledge proof of knowing the randomness. Note that the challenge included in the proof must depend on the voter's anonymous public key, because otherwise votes and proofs could be copied together. The proof itself is a simple proof of knowledge of discrete logarithm, which requires one additional modular exponentiation. If the public board does not publish ballots with invalid proofs, then proving knowledge of the encryption randomness also prevents voters from writing arbitrary cleartext messages onto the board.

Imperfect Anonymous Channel. In practice, the anonymous channel required for submitting the ballot to the bulletin board may be hard to implement. Additionally, it may be difficult to enforce its usage, as some voters may simply decide not to use the anonymous channel at all. To protect the privacy of the voters in such situations, the ballots on the bulletin board may be mixed in a verifiable re-encryption mix-net before being decrypted for the final tallying. This is an additional anonymization step, which is redundant in case of a perfect anonymous channel. In case of an imperfect anonymous channel, however, it prohibits an attacker from seeing somebody's vote in cleartext (the attacker still learns *that* somebody voted, but not *how*). Inserting this additional mixing step makes the tallying procedure more expensive, but since it helps to improve the overall system security in some situations, it is generally recommendable.

3.6 Relation to Existing Work

The idea of a voting scheme based on anonymous keys is relatively unexplored in the literature. The idea first appeared in Neff's paper on verifiable shuffles [26], but only an incomplete sketch of the voting protocol was given. The most obvious difference to our approach is Neff's idea of letting the voters doing their own shuffle before casting a vote. This implies that voters are responsible for their own privacy, while in our protocol privacy is delegated to the set of anonymizers (and thus relies on trust assumptions). On the other hand, Neff's idea implies that vote casting gets extremely expensive, whereas our protocol provides a lightweight voting procedure. Neff briefly mentions the idea that mixing could be done by a set of authorithies [26, Subsection 1.2], but without working it out.

Another closely related work is the preliminary version of this protocol, which recently appeared in [32]. There are various major and minor differences, but the most obvious difference in the preliminary version is the missing DSA signature in the ballot (a zero-knowledge proof has been used instead) and the necessity of using the voter's secret as encryption randomness. As a result, vote casting in the preliminary version requires a total of five modular exponentiations and is thus slightly more expensive than in the current protocol version. The protocol description as provided in [32] did also not mention the extensions discussed in the previous subsection.

Compared to other protocols not providing receipt-freeness, we encountered several important advantages of our scheme. One advantage over the family of protocols relying on homomorphic tallying has already been mentioned at the beginning of Subsection 3.4. Another important advantage over those protocols is the fact that they do not provide full anonymity (everyone can tell who has actually voted). Anonymity is generally a problem in protocols which voters are authenticated directly, for example in schemes based on blind signatures [15, 35, 37] or re-encryption mix-nets [5, 23, 30]. As already discussed in Subsection 1.1, this could have some negative impact with respect to the fairness provided by the voting system. Our protocol is immune against this type of problem, because voters are authenticated anonymously as members of the electorate.

4 Selectio Helvetica

The *Selectio Helvetica* (SH) project aims at developing an Internet voting application based on the protocol described in this paper [12]. It is meant to constitute a proof of concept for building an easy-to-setup and easy-to-use voting service to non-political vote organizers. The implementation differs from the protocol as described here in some major and minor points. We will briefly discuss some of these points and then give an overview of the employed technologies in our implementation. We will also report on our experience with the *Baloti.ch* voting platform, which uses SH under the hood.

Differences. In the modified protocol underlying the SH system, we explicitly introduce two additional play-

ers. The *vote organizer* assesses the voter's right to vote, and the *voting provider* acts as an intermediary among voters and authorities, and writes to the public board.

From the decision of building an Internet voting application, which should be available to users from all around the world, it follows that voters will not be able to properly register in person. There is also no global public-key infrastructure, on which the system could build up. The most important difference in SH is therefore the lack of a proper PKI. Instead, we decided to realize the registration based on e-mail addresses, i.e., proving ownership of an e-mail address is sufficient for registering as a voter. Furthermore, in contrast to the assumption of the protocol, vote organizers may not necessarily be in possession of a final electoral register prior to the beginning of the voting phase. The adapted protocol underlying the SH system provides a solution for this restriction.

Accordingly, we propose the following modification of the registration phase. A potential voter first asks the vote organizer to sign the provided e-mail address in order to confirm the inclusion in the electoral register. The voter then sends the signed e-mail address to the voting provider, which in return sends the voter a registration credential by e-mail. The voter choses a password and sends the voting provider the registration credential along with a set of encrypted hash values of the chosen password, each one designated to a distinct authority. Upon reception of the expected registration credential, the voting provider associates the voter's e-mail address with an unused public key y on the public bulletin board and sends the authorities the encrypted hash values. Then, whenever voters need to access their private signature key x, they request it directly from the authorities, simply by entering their password. Thus, all they need to remember is their password, which they can re-use at subsequent voting events.

Clearly, the convenience of this registration procedure comes with a price. Based on the voters' requests, the authorities would be able to elicit how they voted. Therefore, all votes need to be mixed prior to tallying. Also the voters' e-mail provider could register on their behalf. However, such an attack would be noticed, as the registration credential can only be used once. In such a case, voters would need to make a claim to the vote organizer and have him repeat the registration. The former public key is marked as spoiled and the corresponding anonymous public key revealed. Thus, ballot stuffing is not possible using spoiled credentials.

In order to increase participation, vote organizers can ask the voting provider to include voters in the electoral register even after the vote casting phase has begun. In that case, the voting provider needs to provide at least as many voting credentials as expected voters. In order to prevent ballot stuffing by a sufficiently large group of authorities using unassigned credentials, the anonymous public keys that correspond with unassigned public keys are revealed after the vote casting phase.

The other phases of the SH system follow directly from the protocol presented in this paper.

Employed Technologies. The SH system is implemented using only well-defined, widely used, and standardized technologies. Components communicate through web services. Since web services are based on XML, the components can be implemented and operated on any platform, such as Java EE or .NET. Furthermore, communication channels are secured on the transport layer using HTTPS.

The usability and performance features of the components used by the voters are crucial. At the same time, a technology must be used which is available on virtually all potential computers used by voters. This is addressed by letting voters use standard web browsers running JavaScript.

The server-side components are implemented using the Java EE platform and operated on a JBoss application server. In addition to the core functionality, each component has been enhanced by a management console, which allows to initialize and monitor the components during operation.

Baloti.ch. On the Internet platform *Baloti.ch*, the migrant population living in Switzerland can cast votes with the help of Selectio Helvetica. A public call for integration projects by the Swiss Federal Commission for Migration Issues allowed an interdisciplinary consortium to design and test a multilingual Internet platform mimicking Swiss referendum politics as a two year pilot starting in 2010. Besides politically neutral information on current referendum votes, the website offers an electronic replica of a ballot vote for all issues at stake on the Swiss national level. It thus provides an interesting test bed environment for electronic voting. Because of the political nature of the project and the sensitive information (political preferences) provided by the voters, it was important to provide a secure Internet voting system. In order to build up trust in the system, we opted against having permanently stored user profiles.

The Baloti website is activated three weeks before a national referendum. This corresponds to the period Swiss citizens are allowed to cast their vote by postal mail. During the voting period, the electronic ballot box is open and information on all national votes is displayed. With the help of press releases, coverage on Swiss TV and radio stations, Facebook and Google Ads, etc., *Baloti.ch* was advertised and went online for the first

time during the September 2010 referendum on a revision of the Swiss Unemployment Insurance Law. During the voting period, the website had 3'300 single visitors. Roughly 10 percent of all visitors cast a vote. For the second Baloti vote in November 2010 the website had 4'500 visitors, but fewer votes than in September 2010. The decrease of cast votes could partly be attributed to the complicated nature of the bills and several pending usability problems. During the remaining time of the pilot until the end of 2011, we will address these issues and constantly improve the site.

5 Conclusion

Shuffling and mix-net techniques are known as useful tools in the construction of secure Internet voting systems. While some protocols are based on shuffling the votes and others on shuffling the candidates, this paper introduces a relatively new type of voting protocol based on shuffling the voting credentials. In the proposed protocol, the voting credentials consist of simple DSA public keys. The shuffling of these keys creates a list of anonymous keys, which can no longer be attributed to individual voters, but which can still be used to verify their signatures. We have shown how the shuffling procedure works and how to construct corresponding proofs of correct shuffling.

The resulting Internet voting protocol is extremely simple to explain and very efficient on the voter's side: the voter only needs to encrypt and sign the vote. The protocol is therefore of particular interest for voting devices with limited capacities. Despite its simplicity, the protocol still possesses most of the commonly required security properties. It is not receipt-free and therefore not coercion-resistant, but it could be used as the electronic component of a hybrid voting system, and thus achieve an acceptable degree of coercion-resistance.

Acknowledgments

Research supported by the *Hasler Foundation* (project No. 09037) and the *Mittelbauförderung* of the Bern University of Applied Sciences. We thank all reviewers for their valuable comments.

References

[1] ADIDA, B. Helios: Web-based open-audit voting. In *SS'08, 17th USENIX Security Symposium* (San Jose, USA, 2008), P. Van Oorschot, Ed., pp. 335–348.

[2] ADIDA, B., DE MARNEFFE, O., PEREIRA, O., AND QUISQUATER, J. J. Electing a university president using open-audit voting: Analysis of real-world use of Helios. In *EVT/WOTE'09, Electronic Voting Technology Workshop/Workshop on Trustworthy Elections* (Montreal, Canada, 2009), D. Jefferson, J. L. Hall, and T. Moran, Eds.

[3] ARAÚJO, R., FOULLE, S., AND TRAORÉ, J. A practical and secure coercion-resistant scheme for internet voting. In *Towards Trustworthy Elections: New Directions in Electronic Voting*, D. Chaum, M. Jakobsson, R. Rivest, P. Ryan, J. Benaloh, M. Kutylowski, and B. Adida, Eds., LNCS 6000. Springer, 2010, pp. 330–342.

[4] BANGERTER, E. *Efficient Zero-Knowledge Proofs of Knowledge for Homomorphisms*. PhD thesis, Fakultät für Elektrotechnik und Informationstechnik, Ruhr-Universität Bochum, Germany, 2005.

[5] BENALOH, J. Simple verifiable election. In *EVT'06, USENIX/ACCURATE Electronic Voting Technology Workshop* (Vancouver, Canada, 2006), D. S. Wallach and R. L. Rivest, Eds.

[6] CHAUM, D. Untraceable electronic mail, return addresses and digital pseudonyms. *Communications of the ACM 24*, 2 (1981), 84–88.

[7] CHAUM, D., ESSEX, A., CARBACK, R., CLARK, J., POPOVENIUC, S., SHERMAN, A., AND VORA, P. Scantegrity: End-to-end voter-verifiable optical-scan voting. *IEEE Security & Privacy 6*, 3 (2008), 40–46.

[8] CHAUM, D., AND VAN HEYST, E. Group signatures. In *EUROCRYPT'91, Workshop on the Theory and Application of Cryptographic Techniques* (Brigthon, U.K., 1991), D. W. Davies, Ed., LNCS 547, pp. 257–265.

[9] CLARK, J., AND HENGARTNER, U. Selections: Internet voting with over-the-shoulder coercion-resistance. In *FC'11, 15th International Conference on Financial Cryptography* (St. Lucia, 2011).

[10] CRAMER, R., GENNARO, R., AND SCHOENMAKERS, B. A secure and optimally efficient multi-authority election scheme. *European Transactions on Telecommunications 8*, 5 (1997), 481–490.

[11] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. Tor: The second-generation onion router. In *SS'04, 13th USENIX Security Symposium* (San Diego, USA, 2004), M. Blaze, Ed., pp. 303–320.

[12] DUBUIS, E., FISCHLI, S., HAENNI, R., SERDÜLT, U., AND SPYCHER, O. Selectio Helvetica: A verifiable remote e-voting system. In *CeDEM'11, Conference for E-Democracy and Open Government* (Krems, Austria, 2011).

[13] FIAT, A., AND SHAMIR, A. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86, 6th Annual International Cryptology Conference on Advances in Cryptology* (Santa Barbara, USA, 1986), A. M. Odlyzko, Ed., pp. 186–194.

[14] FISHER, K., CARBACK, R. T., AND SHERMAN, A. T. Punchscan: Introduction and system definition of a high-integrity election system. In *WOTE'06, IAVoSS Workshop On Trustworthy Elections* (Cambridge, U.K., 2006).

[15] FUJIOKA, A., OKAMOTO, T., AND OHTA, K. A practical secret voting scheme for large scale elections. In *ASIACRYPT'92, Workshop on the Theory and Application of Cryptographic Techniques* (Gold Coast, Australia, 1992), J. Seberry and Y. Zheng, Eds., LNCS 718, pp. 244–251.

[16] FURUKAWA, J., AND SAKO, K. An efficient scheme for proving a shuffle. In *CRYPTO'01, 21st Annual International Cryptology Conference on Advances in Cryptology* (Santa Barbara, USA, 2001), J. Kilian, Ed., LNCS 2139, pp. 368–387.

[17] GENNARO, R., JARECKI, S., KRAWCZYK, H., AND RABIN, T. Secure distributed key generation for discrete-log based cryptosystems. In *EUROCRYPT'99, International Conference on the Theory and Application of Cryptographic Techniques* (Prague, Czech Republic, 1999), J. Stern, Ed., LNCS 1592, pp. 295–310.

[18] GOLDBERG, I. On the security of the Tor authentication protocol. In *PET'06, 6th Workshop on Privacy Enhancing Technologies* (Cambridge, U.K., 2006), G. Danezis and P. Golle, Eds., pp. 316–331.

[19] GROTH, J. A verifiable secret shuffle of homomorphic encryptions. *Journal of Cryptology 23*, 4 (2010), 546–579.

[20] HEATHER, J., AND LUNDIN, D. The append-only web bulletin board. In *FAST'08, 5th International Workshop on Formal Aspects in Security and Trust* (Malaga, Spain, 2008), P. Degano, J. Guttman, and F. Martinelli, Eds., LNCS 5491, pp. 242–256.

[21] HIRT, M., AND SAKO, K. Efficient receipt-free voting based on homomorphic encryption. In *EUROCRYPT'00, International Conference on the Theory and Applications of Cryptographic Techniques* (Bruges, Belgium, 2000), G. Goos, J. Hartmanis, and J. van Leeuwen, Eds., LNCS 1807, pp. 539–556.

[22] JAKOBSSON, M., AND JUELS, A. Mix and match: Secure function evaluation via ciphertexts. In *ASIACRYPT'00, 6th International Conference on the Theory and Application of Cryptographic Techniques* (Kyoto, Japan, 2000), T. Okamoto, Ed., LNCS 1976, pp. 162–177.

[23] JAKOBSSON, M., JUELS, A., AND RIVEST, R. L. Making mix nets robust for electronic voting by randomized partial checking. In *SS'02, 11th USENIX Security Symposium* (San Francisco, USA, 2002), D. Boneh, Ed., pp. 339–353.

[24] JONKER, H. L. *Security Matters: Privacy in Voting and Fairness in Digital Exchange*. PhD thesis, Eindhoven University of Technology and University of Luxembourg, 2009.

[25] JUELS, A., CATALANO, D., AND JAKOBSSON, M. Coercion-resistant electronic elections. In *WPES'05, 4th ACM Workshop on Privacy in the Electronic Society* (Alexandria, USA, 2005), V. Atluri, S. De Capitani di Vimercati, and R. Dingledine, Eds., pp. 61–70.

[26] NEFF, C. A. A verifiable secret shuffle and its application to e-voting. In *CCS'01, 8th ACM Conference on Computer and Communications Security* (Philadelphia, USA, 2001), P. Samarati, Ed., pp. 116–125.

[27] NEFF, C. A. Verifiable mixing (shuffling) of ElGamal pairs. Tech. rep., VoteHere, Inc., 2004.

[28] PETERS, R. A. A secure bulletin board. Master's thesis, Department of Mathematics and Computing Science, Technische Universiteit Eindhoven, The Netherlands, 2005.

[29] RYAN, P. Y. A., BISMARK, D., HEATHER, J., SCHNEIDER, S., AND ZHE, X. Prêt à voter: a voter-verifiable voting system. *IEEE Transactions on Information Forensics and Security 4*, 4 (2009), 662–673.

[30] SAKO, K., AND KILIAN., J. Receipt-free mix-type voting scheme: A practical solution to the implementation of a voting booth. In *EUROCRYPT'95, 15th International Conference on the Theory and Applications of Cryptographic Techniques* (Saint-Malo, France, 1995), L. C. Guillou and J. J. Quisquater, Eds., LNCS 921, pp. 393–403.

[31] SHAMIR, A. How to share a secret. *Communications of the ACM 22*, 11 (1979), 612–613.

[32] SPYCHER, O., AND HAENNI, R. A novel protocol to allow revocation of votes in a hybrid voting system. In *ISSA'10, 9th Annual Conference on Information Security – South Africa* (Sandton, South Africa, 2010).

[33] SPYCHER, O., HAENNI, R., AND DUBUIS, E. Coercion-resistant hybrid voting systems. In *EVOTE'10, 4th International Workshop on Electronic Voting* (Bregenz, Austria, 2010), R. Krimmer and R. Grimm, Eds., no. P-167 in Lecture Notes in Informatics, Gesellschaft für Informatik E.V., pp. 269–282.

[34] SPYCHER, O., KOENIG, R., HAENNI, R., AND SCHLÄPFER, M. A new approach towards coercion-resistant remote e-voting in linear time. In *FC'11, 15th International Conference on Financial Cryptography* (St. Lucia, 2011).

[35] SUNG, S. H. Y., AND LEE, J. An electronic voting scheme based on undeniable blind signature. In *ICCST'03, 37th Annual International Carnahan Conference on Security Technology* (Taipei, Taiwan, 2003), pp. 163–167.

[36] WIKSTRÖM, D. A commitment-consistent proof of a shuffle. In *ACISP'09, 14th Australasian Conference on Information Security and Privacy* (Brisbane, Australia, 2009), C. Boyd and J. González Nieto, Eds., LNCS 5594, pp. 407–421.

[37] XIA, Z., AND SCHNEIDER, S. A new receipt-free e-voting scheme based on blind signature. In *WOTE'06, IAVoSS Workshop on Trustworthy Elections* (Cambridge, U.K., 2006), pp. 127–135.