Masked Ballot Voting for Receipt-Free Online Elections

Roland Wen and Richard Buckland

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia
{rolandw,richardb}@cse.unsw.edu.au

Abstract. To prevent bribery and coercion attacks on voters, current online election schemes rely on strong physical assumptions during the election. We introduce Masked Ballot, an online voting scheme that mitigates these attacks while using a more practical assumption: untappable channels are available but only *before* the election. During the election voters cast ballots over completely public channels without relying on untappable channels, anonymous channels or trusted devices. Masked Ballot performs only the voting part of an election and is designed to integrate with counting schemes that compute the final election result.

Keywords: Receipt-freeness, online elections, voting schemes.

1 Introduction

The secret ballot is a fundamental instrument for protecting the freedom of choice of voters. It mitigates bribery and coercion because nobody knows whether voters are lying about how they voted. Replicating the secret ballot in a virtual setting is one of the greatest challenges of designing online election schemes. Although many innovative cryptographic solutions have been put forward, the required high level of secrecy forces these schemes to rely on strong physical assumptions during the election and sometimes also before the election. In this paper we develop a new approach to constructing online voting schemes. Our approach shifts strong assumptions from the election itself to a registration stage before the election so that the voting can take place over insecure public networks such as the Internet.

The underlying difficulty in online elections arises from reconciling confidentiality with verifiability. To prevent bribery and coercion, online elections must be receipt-free, which means voters cannot prove how they voted [2]. Receipt-freeness is an intrinsic property of most traditional elections, where the conventional secret ballot ensures that the votes are completely confidential and that voters have no evidence of how they voted. But online elections must also be both individually verifiable and universally verifiable. Individual verifiability means each voter can confirm the ballot cast corresponds to the intended vote, while universal verifiability means any observer can confirm that the authorities conducted the election

P.Y.A. Ryan and B. Schoenmakers (Eds.): VOTE-ID 2009, LNCS 5767, pp. 18–36, 2009.

correctly. To achieve these strong notions of verifiability, the voting process must publicly expose the ballots to some extent. Then to satisfy receipt-freeness the public ballots must appear ambiguous to conceal the votes. Given a public ballot, the voter's transcript of the ballot construction for a genuine vote must be *indistinguishable* from a fake transcript for any other possible vote.

Receipt-free voting schemes typically provide indistinguishability through the use of secret randomness. In general the ballot construction must contain an element of randomness that remains secret from the voter. Otherwise the ballot transcript suffices as a receipt that unequivocally corresponds to a unique vote. For example suppose that a voter constructs a ballot by probabilistically encrypting a vote. Then the ballot transcript contains the encrypted vote, the plaintext vote and the randomness for the encryption. Anyone can confirm whether the transcript is genuine simply by encrypting the vote with the given randomness and comparing the ciphertext with the ballot. But if the randomness is secret from the voter, then genuine and fake transcripts are indistinguishable.

The dilemma with using secret randomness is that while indistinguishability must hold, the voter must still learn the actual vote that corresponds to the ballot. The authorities could construct the ballots and then secretly reveal the votes to the voters. However a powerful coercive adversary who intercepts all communication between the voters and authorities would learn all the information that a voter knows. In this case the adversary could distinguish between genuine and fake ballot transcripts. To enable voters to generate fake ballot transcripts, receipt-free schemes require physical assumptions that limit the adversary's knowledge of the genuine transcripts. All previous schemes rely on one of the following three alternative assumptions.

1. During the election voters and election authorities have access to untappable channels. The adversary cannot intercept any secret data transmitted via these channels.
2. During the election voters have access to anonymous channels. In addition, before the election voters and election registrars have access to untappable channels.
3. During the election voters have secure access to trusted randomisers that generate secret randomness. In practice the randomisers are trusted devices such as smart cards.

Each approach has practical limitations, for instance untappable channels are difficult to implement over the Internet. To achieve receipt-freeness some strong assumption appears inevitable. However for practical Internet elections it is desirable to avoid such assumptions during the voting.

1.1 Contributions

We introduce Masked Ballot, an online voting scheme that achieves receipt-freeness under a more practical physical assumption: before the election there are one-way untappable channels from a registrar to the voters. As a registration stage in advance of an election is orders of magnitude longer than the

voting stage, offline implementations of untappable channels become feasible, for instance by post or face-to-face communication. During the election voters submit their ballots over insecure public channels. In this setting attacks such as forced abstention and forced random voting are unavoidable. The stronger property of coercion-resistance [11] prevents these attacks but additionally requires anonymous channels during the voting. Hence under our assumptions only receipt-freeness is possible. We prove receipt-freeness using Moran and Naor's formal model [15].

Masked Ballot is purely a voting scheme and as such is independent of the vote encoding and counting method. The idea behind the scheme is to disguise each vote with a secret mask. Each voter receives a single-use mask during a registration stage before the election. The registration is a trusted process and voters can generate fake mask transcripts for any possible mask. Then during the election voters construct ballots by combining their votes with their masks. Any party can subsequently unmask the votes in a universally verifiable manner.

The secret mask ensures receipt-freeness. An adversary can force a voter to reveal the masked vote. However the voter can generate a fake mask transcript such that the masked vote is consistent with any fake vote.

1.2 Organisation

Section 2 discusses contemporary approaches to constructing receipt-free online voting schemes. Section 3 defines the security model, Section 4 summarises Moran and Naor's definition of receipt-freeness and Section 5 covers the necessary cryptographic building blocks. Section 6 describes the details of the Masked Ballot voting scheme and Section 7 analyses the security and complexity of the scheme.

2 Related Work

There is extensive literature on receipt-free online voting schemes. We classify the schemes according to the three different physical assumptions and analyse the practical challenges of each approach.

2.1 The Untappable Channels Approach

Many voting schemes use untappable channels during the election [2,9,18,20]. A voter constructs a ballot by interacting with the authorities. The voter and authorities can exchange secret information, such as the vote and zero-knowledge proofs, via untappable channels. Since voters can plausibly lie about this secret information, they can generate fake ballot transcripts for any vote.

Untappable channels pose several challenges. First, implementing completely untappable channels can be problematic. Although in some cases it could be reasonable to assume the Internet provides sufficient protection against eavesdropping, in general a powerful adversary can potentially intercept all communication. Hence it is hard to *guarantee* that eavesdropping is impossible.

Another difficulty is resolving disputes about messages sent via untappable channels. If a voter claims that the secret values sent by an authority are invalid, then only the voter and the authority know which party is dishonest. The inability to resolve such disputes can potentially disrupt the entire election process.

The final concern is the extent of trust in the authorities. Even when the trust is distributed among multiple authorities, a voter must know at least one honest authority in order to safely generate a fake ballot transcript [9]. An adversary potentially knows the genuine transcripts for all communication between the voter and the possibly corrupt authorities. Hence voters can only generate fake transcripts for their communication with known honest authorities. In some cases it might be reasonable to assume that each voter does know an honest authority. For example if each candidate acts as an authority, then a voter's preferred candidate is presumably trustworthy. However in general voters cannot trust any particular authority. For this reason schemes relying on untappable channels typically assume that an adversary cannot corrupt or collude with any authorities to compromise receipt-freeness.

2.2 The Anonymous Channels Approach

Another approach is to use anonymous channels during the election and untappable channels before the election [17]. In contrast to the untappable channels approach, the secret randomness is in an anonymous credential rather than the ballot. During a registration stage before the election, registrars send credentials to the voters via untappable channels. The voter can generate a fake credential transcript for any possible credential and an adversary cannot distinguish between genuine and fake credentials. The voter can use the same credential for multiple elections. During the election each voter submits a credential-ballot pair via an anonymous channel. The credential-ballot pair consists of an encrypted credential and an encrypted vote. Although voters cannot generate fake ballot transcripts, they can submit additional credential-ballot pairs with fake credentials. The anonymous channels ensure the adversary cannot otherwise trace credential-ballot pairs to the voters. All credential-ballot pairs are public but only those with genuine credentials contribute to the election result.

The existence of untappable channels before the election is still a fairly strong physical assumption but the possibility of offline implementations makes it more practical than the use of untappable channels during the election. Resolving disputes is still problematic but at least there is now an opportunity to find a solution before the election begins. However there remains an issue with trust in the registrars. As in the untappable channels approach, a voter must know at least one honest registrar, and so the general assumption is that all the registrars are honest. In addition distributing the trust among multiple registrars can pose an inconvenience for voters, for instance if the voter must exchange data in person with each registrar. For simplicity and convenience, schemes often assume that there is a single trusted registrar [11]. This is fairly reasonable as some degree of trust in the registrar is unavoidable even in traditional elections.

During the election the use of anonymous channels is more practical than the use of untappable channels. Solutions such as public terminals or mix-nets can often provide a sufficient degree of anonymity. However it is hard to guarantee complete anonymity in all cases. For example a mix-net can only conceal the correspondence between its input and output messages. Achieving receipt-freeness would still require the assumption of untappable channels between the voters and the mix-net. Otherwise an adversary who intercepts all communication between the voters and the mix-net can potentially identify the voters and force them to reveal their votes. Furthermore the first server in the mix-net can similarly identify the voters and their votes. Hence the first mix server must be honest.

A limitation with the anonymous channels approach is that a subtle coercion attack is possible by a fully adaptive adversary who can coerce voters at any time, including after the voting period. In contrast to the untappable channels approach, a voter cannot lie about any arbitrary vote. The voter can only submit an appropriate ballot with a fake credential in response to an adversary's instructions during the voting period. If the adversary only coerces a voter after the voting is complete, for instance to check who voted for the local Mafia boss, then the voter cannot reveal a plausible ballot. To counter such attacks a voter must always submit a credential-ballot pair for each possible vote, using the genuine credential only for the desired vote. While this may be acceptable when the number of possible votes is small, it is impractical in the general case.

2.3 The Trusted Randomisers Approach

The final approach is to use trusted randomisers during the election [1,12,13]. A voter constructs a ballot by interacting with a randomiser via an untappable channel. Then the voter submits the ballot over public channels. This is similar to the untappable channels approach, but now a randomiser provides the secret randomness.

In practice randomisers can be implemented by tamper-resistant devices such as smart cards. The untappable channel thus becomes a local channel rather than a network communication channel. Hence this approach can be much more feasible than the untappable channels approach. However smart cards have several drawbacks. Currently, suitable smart cards and card readers are relatively expensive and are not yet widespread. More concerning is the failure model for these devices. Equipment failure may prevent voters from voting, since it can take time to obtain a replacement smart card. Furthermore an adversary who compromises the devices could commit large-scale fraud.

3 Security Model

3.1 Participants

A voting scheme has three types of participants.

Voters. A voter is a participant who can vote in the election. Voters cast ballots and then take no further part in the election.

Registrar. A trusted registrar maintains the electoral roll and ensures that only eligible voters can participate. The registrar interacts with voters before the election but takes no part in the election itself. Although it is possible to distribute the trust among multiple registrars, for simplicity we generally consider only a single registrar.

Authorities. An authority helps to conduct the election. Multiple authorities collaborate to process the ballots and compute the election result. As Masked Ballot performs only the voting and not the subsequent counting, the authorities have a passive role in our scheme.

We assume that the registrar and authorities have large computational, communication and storage resources, but voters might only have limited resources.

3.2 Communication Model

The participants communicate by sending messages via two types of channels.

Untappable channel. An untappable channel ensures that communication is completely private and no eavesdropping is possible. We assume untappable channels provide authentication of both senders and receivers.

Bulletin board. A bulletin board is a public broadcast channel with memory. Participants can post messages but no party can delete or modify posted messages. Any party can read all posted messages. We assume the bulletin board provides authentication of senders.

Before the election each voter can receive messages from the registrar via a one-way untappable channel. During the election voters and authorities post messages to the bulletin board via public channels such as the Internet.

3.3 Adversary Model

We model cheating by a central adversary with the following powers.

Active corruption. The adversary has complete control of corrupt participants. It can privately communicate with corrupt participants and always knows their internal states. The adversary can instruct corrupt participants to arbitrarily deviate from the protocol in any way it desires.

Adaptive corruption of voters. The adversary can corrupt any voter at any time before, during or after the election.

Static corruption of authorities. The adversary can corrupt any authority only at the start of the election.

Threshold corruption. The adversary can corrupt any number of voters but only up to a threshold of authorities.

Adaptive coercion. The adversary can coerce a voter at any time before, during or after the election. It can privately communicate with coerced participants but has no knowledge of their internal states.

Eavesdropping. The adversary can intercept all communication apart from messages sent via untappable channels.

We assume coercion occurs remotely, and the voting environment and device are both secure. The adversary cannot observe voters physically (through cameras or shoulder surfing) or electronically (through malware).

3.4 Security Requirements

The voting scheme must satisfy the following requirements.

Receipt-Freeness. Voters cannot prove how they voted. We provide a formal definition in the next section.
Authenticity. Only eligible voters can participate.
Uniqueness. Each voter has only one vote.
Vote Independence. A voter cannot cast a vote that is some function of another voter's vote. For instance a voter cannot copy another vote without knowing the actual vote.
Individual Verifiability. Each voter can confirm the cast ballot corresponds to the intended vote.
Universal Verifiability. Any observer can confirm the voting is correct.
Robustness. The voting tolerates the corrupt or faulty behaviour of any group of authorities up to a threshold.

4 Receipt-Freeness

Moran and Naor provide a simulation-based definition of receipt-freeness. The definition explicitly permits null vote (abstention and invalid ballot) attacks and random vote attacks, but otherwise captures the full range of attacks by an adaptive adversary. This section summarises the definition. A detailed description appears in Chapter 5 of Moran's thesis [14].

The simulation paradigm establishes the security of a protocol by comparing an ideal specification of the protocol's functionality in an ideal world with the execution of the protocol in the real world. In the ideal world a trusted third party, known as the ideal functionality, accepts inputs from the participants via completely secure channels and then performs the specified computation. In the real world the participants follow the protocol to perform the computation. A protocol is secure if all attacks by an adversary in the real world are also possible in the ideal world. In other words the protocol securely emulates the ideal functionality.

Moran and Naor's definition extends the standard simulation model to capture receipt-freeness. Under this definition a receipt-free protocol requires a coercion-resistance strategy that specifies how coerced voters in the real world respond to the adversary's queries and commands. A scheme is receipt-free if the adversary cannot distinguish whether a coerced voter follows the coercion-resistance strategy or follows the adversary's instructions.

Note that an adversary can still potentially coerce voters by examining only the output of an ideal counting process. For example suppose the counting output includes all the (anonymised) votes. For a plurality election it might be

reasonable to assume that the output is receipt-free in almost all cases, but for a preferential election coercion is possible through signature attacks [6]. We address this issue in other work on preferential counting [21].

4.1 The Ideal World

There are n voters $V_1, \ldots, V_n$. Each voter has three secret inputs.

1. An intended vote that the voter wishes to cast.
2. A fake vote that the voter will reveal when resisting coercion.
3. A coercion-response bit that determines whether a voter complies with or resists coercion.

Each voter V_i submits a cast vote v_i to the ideal functionality, which then computes $f(v_1, \ldots, v_n)$ and broadcasts the result. For an honest voter the cast vote is the intended vote. For corrupt or coerced voters, the cast vote may be altogether different. The ideal adversary $\mathcal{I}$ can adaptively corrupt and coerce voters in the following manner.

Corrupting a voter V. V reveals the intended vote to $\mathcal{I}$ and follows $\mathcal{I}$'s instructions to cast any forced vote.

Coercing a voter V. If the coercion-response bit is $c = 1$ then V complies with coercion by revealing the intended vote to $\mathcal{I}$. If $c = 0$ then V resists coercion by revealing the fake vote to $\mathcal{I}$. After coercing the voter, $\mathcal{I}$ can also instruct V to cast any forced vote. If the forced vote is a null vote $\perp$ or random vote $*$, then V complies regardless of c. Otherwise if $c = 1$ then V complies by casting the forced vote and if $c = 0$ then V resists by casting the intended vote.

$\mathcal{I}$'s view contains the intended votes of corrupt voters, either the intended or fake votes of coerced voters (depending on their coercion-response bits), any forced votes of corrupt or coerced voters, the output of the ideal functionality, and the randomness it used.

4.2 The Real World

There are n voters $V_1, \ldots, V_n$ each with an intended vote, fake vote and coercion-response bit as in the ideal world. The voters follow the real-world protocol to submit the cast votes $v_1, \ldots, v_n$ and compute $f(v_1, \ldots, v_n)$. The real-world adversary $\mathcal{A}$ can adaptively corrupt and coercer voters in the following manner.

Corrupting a voter V. $\mathcal{A}$ can send commands and queries to V, who follows $\mathcal{A}$'s instructions exactly. Hence $\mathcal{A}$ can learn V's entire internal view.

Coercing a voter V. $\mathcal{A}$ can send commands and queries to V. If the coercion-response bit is $c = 1$ then V behaves exactly as a corrupt voter. If $c = 0$ then V follows the coercion-resistance strategy to respond to commands and queries.

$\mathcal{A}$'s view consists of the views of corrupt voters, all its communication with coerced voters, all public communication, and the randomness it used.

4.3 Definition of Receipt-Freeness

Definition 1 (Receipt-Freeness). *A protocol is receipt-free if for every real adversary $\mathcal{A}$ there exists an ideal adversary $\mathcal{I}$ who corrupts and coerces exactly the same participants as $\mathcal{A}$, and the following condition holds: for any intended votes $v_1, \ldots, v_n$, fake votes $v'_1, \ldots, v'_n$ and coercion-response bits $c_1, \ldots, c_n$, $\mathcal{I}$ can simulate a view in the ideal world that is indistinguishable from $\mathcal{A}$'s view in the real world, where the distributions are over the randomness used by $\mathcal{I}$, $\mathcal{A}$ and the voters.*

5 Cryptographic Preliminaries

The Masked Ballot voting scheme requires a threshold homomorphic cryptosystem and compatible zero-knowledge proofs to prove certain properties of encrypted messages.

5.1 Threshold Homomorphic Cryptosystem

A homomorphic cryptosystem is a public-key cryptosystem that enables any party to efficiently compute an encryption of the sum or product of two messages given only the encryptions of the individual messages. Suitable candidates are the Paillier cryptosystem [19] and the ElGamal cryptosystem [7].

For concreteness we describe the scheme using Paillier, which is semantically secure under the Decisional Composite Residuosity Assumption. The public key is (g, n), where $n = pq$ is an RSA modulus and $g = n + 1$. All plaintext operations are modulo n and all ciphertext operations are modulo n^2. For simplicity we omit the modular reduction in the notation.

A message $m \in \mathbb{Z}_n$ is encrypted by randomly generating $r \in \mathbb{Z}_n^*$ and computing the ciphertext

$$[\![m]\!] = g^m r^n \in \mathbb{Z}_{n^2}^* \ .$$

The Paillier cryptosystem is additively homomorphic. For the plaintexts $m_1, m_2 \in \mathbb{Z}_n$,

$$\begin{aligned} [\![m_1]\!] \boxplus [\![m_2]\!] &= (g^{m_1} r_1^n) \times (g^{m_2} r_2^n) \\ &= g^{m_1+m_2} (r_1 r_2)^n \\ &= [\![m_1 + m_2]\!] \ . \end{aligned}$$

In the threshold version of Paillier [5,8], each authority has a share of the private key. A quorum of authorities must collaborate to decrypt any ciphertext.

5.2 Non-interactive Zero-Knowledge Proofs

Masked Ballot uses two types of non-interactive zero-knowledge proofs: a proof of plaintext knowledge [3] and a designated-verifier proof of correct encryption. A proof of correct encryption [5] shows that a given ciphertext $[\![m]\!]$ is an encryption

of the given plaintext m. Converting this into a designated-verifier proof [10] enables the prover to convince only a specific verifier that the proof is valid. The prover constructs the proof using the verifier's public key and the verifier can generate a fake proof using its private key.

6 The Masked Ballot Voting Scheme

The Masked Ballot voting scheme has three stages: registration, voting and unmasking. The registration stage takes place in advance of the election, and the election itself consists of the voting and unmasking stages. Apart from the use of one-way untappable channels during the registration stage, all communication is via the authenticated bulletin board.

At a conceptual level Masked Ballot is essentially a hybrid of the Juels-Catalano-Jakobsson (JCJ) scheme [11] and the Cramer-Gennaro-Schoenmakers (CGS) scheme [4]. During the registration stage voters obtain secret masks in the same way as credentials in the JCJ scheme. Then during the voting stage voters cast ballots as in the CGS scheme. The important difference is that rather than submitting encrypted votes, voters submit encrypted masked votes. The masked vote combines a mask and a vote using a standard additive secret sharing technique, much like in Moran and Naor's paper-based election scheme [16]. To unmask a ballot the authorities use the homomorphic property of the cryptosystem to combine the encrypted mask from the registration stage with the masked ballot from the voting stage.

Note in the following protocol descriptions we sometimes abuse notation to have $[\![x]\!]$ refer to a variable that contains an encryption of x.

6.1 Initialisation

First the authorities perform the necessary initialisation steps.

1. Set up an authenticated bulletin board and establish access mechanisms for the registrar, authorities and voters.
2. Set up the threshold cryptosystem. Each authority has a secret share of the private key.
3. Publish the public key and any system parameters.

6.2 Registration Stage

In advance of the election, the trusted registrar provides each voter with a secret mask using Protocol 1. The input is the voter's identifier V and public key pk. Voters must only use their masks for a single election.

The untappable channels and designated-verifier proofs prevent other parties from learning any information about the masks. We assume that voters know their private keys, and so they can generate fake mask transcripts for any possible mask. Alternatively voters can provide proofs of knowledge of their private keys [9]. A more simple option is that voters could generate a single-use key pair for

1: **register**(V, pk)
2: $m \leftarrow$ random element of $\mathbb{Z}_n$
3: $[\![m]\!] \leftarrow$ `encrypt`(m)
4: $d \leftarrow$ designated-verifier proof for pk that $[\![m]\!]$ is an encryption of m
5: `post`$(V, [\![m]\!])$ to the bulletin board
6: `send`(m, d) to the voter via a *one-way untappable channel*

Protocol 1: Registering a voter

the election and then reveal their private keys to the registrar immediately after they receive their masks.

Distributing the registration among multiple registrars is possible. In this case each registrar follows the single registrar protocol to provide a voter with a mask share. The voter's combined mask is simply the sum of the shares. The homomorphic cryptosystem enables any party to compute the encrypted mask from the posted encryptions of the mask shares. As long as at least one registrar is honest, the mask remains secret. However, as discussed in Section 2, the voter must still know an honest registrar and there must be some procedure for resolving disputes.

6.3 Voting Stage

During the voting stage each voter casts a masked ballot using Protocol 2. The inputs are the voter's secret mask $m \in \mathbb{Z}_n$ and vote $v \in \mathbb{Z}_n$.

1: **vote**(m, v)
2: $[\![v - m]\!] \leftarrow$ `encrypt`$(v - m)$
3: $p \leftarrow$ proof of plaintext knowledge of $[\![v - m]\!]$
4: `post`$([\![v - m]\!], p)$ to the bulletin board

Protocol 2: Casting a masked ballot

The intuition for receipt-freeness is that $(v - m)$ is identical to any fake vote v' and fake mask m' such that $v' - m' = v - m$. A coerced voter responds to the adversary using the coercion-resistance strategy in Protocol 3.

Since the voting is non-interactive and the adversary cannot observe the voter during the voting, we can consider the voting protocol as an atomic operation. The adversary only knows a voter has completed the voting when the voter posts the ballot. Hence the coercion-resistance strategy varies according to whether coercion of a voter starts before or after the ballot is cast.

Before the ballot is cast the adversary can specify the coerced voter's ballot. The adversary can instruct the voter to abstain or cast an invalid ballot, resulting in a null vote attack. Alternatively the adversary can instruct the voter to cast a valid ballot for a prescribed random value instead of $(v - m)$, resulting in a

Coercion-resistance strategy

1: **if** before ballot is cast
2: $m' \leftarrow$ random element of $\mathbb{Z}_n$
3: **else**
4: $m' \leftarrow v' - (v - m)$
5: $d' \leftarrow$ fake designated-verifier proof that $[\![m]\!]$ is an encryption of m'
6: follow the adversary's instructions exactly except with fake (m', d') instead of genuine (m, d)

Protocol 3: Coercion-resistance strategy

random vote attack. Otherwise the adversary can instruct the voter to cast a ballot for a forced vote. In this case the voter uses a fake random mask, and so the ballot again contains a random vote. Notice attempts to coerce the voter to cast a forced vote without having the voter commit to the mask are futile.

After the ballot is cast the adversary cannot influence the coerced voter's ballot. The adversary can only learn the fake mask for any given fake vote.

6.4 Unmasking Stage

After the voting stage any party can verify the ballots and unmask the valid ballots using Protocol 4. The input is the voting transcript $\mathbb{T}$, which is the list of public voter transcripts. Each public voter transcript is of the form $(V, [\![m]\!], [\![v - m]\!], p)$, where V is a voter's identifier, $[\![m]\!]$ is the encrypted mask, $[\![v - m]\!]$ is the masked ballot and p is the proof of plaintext knowledge.

1: **unmask**($\mathbb{T}$)
2: **for** each $(V, [\![m]\!], [\![v - m]\!], p) \in \mathbb{T}$
3: **if** p is incorrect
4: `post`$(V, \text{invalid})$ to the bulletin board
5: **else**
6: $[\![v]\!] \leftarrow [\![v - m]\!] \boxplus [\![m]\!]$
7: `post`$(V, [\![v]\!])$ to the bulletin board

Protocol 4: Unmasking the ballots

6.5 Counting

After unmasking the ballots the authorities use an appropriate counting scheme to compute the election result in accordance with a prescribed electoral system. Intermediate integration steps may be necessary to transform the ballots into a valid form for the counting. The precise integration procedure is specific to each counting scheme.

7 Analysis

7.1 Security

The Masked Ballot scheme satisfies the common security requirements for online voting schemes. We prove receipt-freeness in the next subsection. The proof also implies correctness and robustness.

Authenticity and Uniqueness. The authenticated bulletin board ensures that only eligible voters can submit ballots and that each voter submits only a single ballot.

Vote Independence. The proof of plaintext knowledge ensures that the voter knows the plaintext masked vote.

Individual Verifiability. In the registration stage the proof of correct encryption convinces the voter that the mask is correct. In the voting stage the voter can use the private data to reconstruct the ballot and compare it to the posted ballot.

Universal Verifiability. The unmasking stage requires only deterministic operations on posted messages. Any observer can perform these operations and verify the correctness of the posted results.

7.2 Proof of Receipt-Freeness

To prove receipt-freeness under Definition 1, we construct a simulator in the ideal world and show that an adversary's real-world view is indistinguishable from the simulated view.

In the ideal world, an ideal adversary $\mathcal{I}$ can interact with the ideal voting functionality $\mathcal{F}_{\text{VOTING}}$ (Protocol 5). As Masked Ballot is purely a voting scheme and does not consider the counting, the output of $\mathcal{F}_{\text{VOTING}}$ is simply the list of voters who cast non-null votes.

$\mathcal{I}$ runs a black-box simulation (Protocol 6) using oracle access to the real adversary $\mathcal{A}$. Whenever $\mathcal{A}$ corrupts a voter in the real world, $\mathcal{I}$ corrupts the corresponding voter in the ideal world and learns the intended vote. Whenever $\mathcal{A}$ coerces a voter in the real world, $\mathcal{I}$ coerces the corresponding voter in the ideal world and, depending on the voter's secret coercion-response bit, learns either the intended or fake vote. For corrupt and coerced voters, $\mathcal{I}$ also learns any forced vote provided by $\mathcal{A}$. $\mathcal{I}$ simulates the necessary parts of the voters' real-world views using its knowledge of the votes for corrupt and coerced voters, and all communication with $\mathcal{A}$.

In the real world a voter's view consists of the mask transcript $\mathbb{M}$ and the ballot transcript $\mathbb{B}$. The mask transcript is $\mathbb{M} = (\llbracket m \rrbracket, m, d, pk, sk)$ where:

$\llbracket m \rrbracket$	is the encrypted mask,
m	is the mask,

d	is the designated-verifier proof that $[\![m]\!]$ is an encryption of m, and
(pk, sk)	is the key pair for the proof d.

The ballot transcript is $\mathbb{B} = ([\![v - m]\!], p, v - m, r)$ where:

$[\![v - m]\!]$	is the masked ballot,
p	is the proof of plaintext knowledge,
v	is the vote,
m	is the mask, and
r	is the randomness for the encryption $[\![v - m]\!]$ and the proof p.

The voter's public transcript is $\mathbb{P} = ([\![m]\!], [\![v - m]\!], p)$ and the remaining values in $\mathbb{M}$ and $\mathbb{B}$ form the voter's internal view. We need not explicitly consider the unmasking transcript because it is a known, deterministic function of $\mathbb{P}$.

We assume the registration is secure. To generate fake proofs of correct encryption, the designated-verifier property of the proofs must hold. Furthermore the registrar must be honest, otherwise receipt-freeness would be broken. However the encryption of the masked vote and the zero-knowledge property of the proof of plaintext knowledge ensure honest voters would still retain privacy of their votes.

We also assume that the threshold cryptosystem is semantically secure and that some threshold of authorities is corrupted statically but a quorum remains honest. Then the corrupt authorities cannot compromise the protocol execution in any way.

Lemma 1 (Indistinguishability). *$\mathcal{I}$'s simulated view in the ideal world and $\mathcal{A}$'s view in the real world are computationally indistinguishable.*

Functionality $\mathcal{F}_{\text{VOTING}}$

Vote, V, v	Accept this command from an honest voter V or the adversary if V is corrupt. Store (V, v) and disregard subsequent **Vote** commands for V. If the command is from the voter then notify the adversary that V has voted.
Vote, $V, *$	Accept this command from the adversary if the voter V is coerced. This represents a random vote so randomly choose a vote v, store (V, v) and disregard subsequent **Vote** commands for V.
Vote, $V, \perp$	Accept this command from the adversary if the voter V is corrupt or coerced. This represents an abstention so store a null vote $(V, \perp)$ and disregard subsequent **Vote** commands for V.
BeginVoting	Start accepting **Vote** commands.
EndVoting	Stop accepting **Vote** commands and then output the list of voters who cast non-null votes.
RevealVotes	Reveal the stored non-null votes pairs (V, v) to an ideal counting functionality.

Protocol 5: The ideal voting functionality $\mathcal{F}_{\text{VOTING}}$

Ideal world simulation

Initialisation

$\mathcal{I}$ simulates the generation of the authorities' public key and their shares of the private key. Notice $\mathcal{I}$ can decrypt messages using the authorities' shares of the private key.

Registration Stage

For each voter, $\mathcal{I}$ simulates the generation of the key pair (pk, sk). Notice $\mathcal{I}$ can generate fake proofs with the voter's private key. In addition $\mathcal{I}$ simulates the registration process using Protocol 1 to generate the mask m. Then $\mathcal{A}$ learns $[\![m]\!]$.

Voting Stage

At the beginning of the voting stage send **BeginVoting** to $\mathcal{F}_{\text{VOTING}}$ and then simulate the voters' views.

Honest voter. As the intended vote is unknown, $\mathcal{I}$ randomly selects a vote v and uses the mask m to simulate V's view using Protocol 2. Then $\mathcal{A}$ learns $([\![v - m]\!], p)$ and hence V's public transcript $\mathbb{P} = ([\![m]\!], [\![v - m]\!], p)$.

Corrupt voter. $\mathcal{A}$ can corrupt a voter V either before or after V casts a vote.

1. *Before the vote is cast.* At this point $\mathcal{A}$ only knows $[\![m]\!]$. $\mathcal{I}$ uses the mask m to simulate V's view according to $\mathcal{A}$'s instructions. If any of the instructions cause a null or random vote, then $\mathcal{I}$ submits $(\mathbf{Vote}, V, \perp)$ or $(\mathbf{Vote}, V, *)$ to $\mathcal{F}_{\text{VOTING}}$. Otherwise $\mathcal{A}$ instructs V to submit a valid ballot $([\![v - m]\!], p)$. Then $\mathcal{I}$ derives the forced vote $v = \texttt{decrypt}\,([\![v - m]\!]) + m$ and submits $(\mathbf{Vote}, V, v)$ to $\mathcal{F}_{\text{VOTING}}$. $\mathcal{A}$ learns V's mask transcript $\mathbb{M} = ([\![m]\!], m, d, pk, sk)$ and ballot transcript $\mathbb{B} = ([\![v - m]\!], p, v, m, r)$.
2. *After the vote is cast.* Originally $\mathcal{I}$ simulated V's view using a random vote v and the mask m. The original transcripts are $\mathbb{M} = ([\![m]\!], m, d, pk, sk)$ and $\mathbb{B} = ([\![v - m]\!], p, v, m, r)$ but at this point $\mathcal{A}$ only knows $\mathbb{P} = ([\![m]\!], [\![v - m]\!], p)$. Now $\mathcal{I}$ learns the intended vote v' and must update the transcripts. It constructs a fake mask $m' = v' - (v - m)$ and a fake proof d' that $[\![m]\!]$ is an encryption of m'. Then $\mathcal{A}$ learns V's updated transcripts $\mathbb{M}' = ([\![m]\!], m', d', pk, sk)$ and $\mathbb{B}' = ([\![v' - m']\!], p, v', m', r)$. Since $v' - m' = v - m$, the updated transcripts match the original public transcript.

Coerced voter. $\mathcal{A}$ can coerce a voter V either before or after V casts a vote.

1. *Before the vote is cast.* The simulation is the same as for a corrupt voter before the vote is cast. The only difference is instead of submitting $(\mathbf{Vote}, V, v)$ to $\mathcal{F}_{\text{VOTING}}$ for a forced vote, $\mathcal{I}$ submits $(\mathbf{Vote}, V, *)$.
2. *After the vote is cast.* The simulation is the same as for a corrupt voter after the vote is cast. The only difference is the revealed vote could be either the intended or fake vote.

At the end of the voting stage send **EndVoting** to $\mathcal{F}_{\text{VOTING}}$.

Protocol 6: Ideal world simulation

Proof (Sketch). For each voter, the simulated transcripts in Protocol 6 are computationally indistinguishable from the real-world transcripts.

Case 1 (Honest voter). The simulated and real-world public transcripts are $\mathbb{P}_S = (\llbracket m_S \rrbracket, \llbracket v' - m_S \rrbracket, p_S)$ and $\mathbb{P}_R = (\llbracket m_R \rrbracket, \llbracket v - m_R \rrbracket, p_R)$. The difference is that $\mathbb{P}_S$ is for a random vote v' whereas $\mathbb{P}_R$ is for the intended vote v. The semantic security of the cryptosystem ensures the transcripts are computationally indistinguishable.

Case 2 (Corrupt voter: before vote cast). The simulated transcripts are $\mathbb{M}_S = (\llbracket m_S \rrbracket, m_S, d_S, pk_S, sk_S)$ and $\mathbb{B}_S = (\llbracket v - m_S \rrbracket, p_S, v, m_S, r_S)$. These transcripts are consistent with the forced vote v. Then the simulated and real-world transcripts are identical because they are both consistent transcripts for the same forced vote.

Case 3 (Corrupt voter: after vote cast). The simulated ballot transcript $\mathbb{B}_S = (\llbracket v - m'_S \rrbracket, p_S, v, m'_S, r_S)$ is consistent with the intended vote v. However the simulated mask transcript $\mathbb{M}_S = (\llbracket m_S \rrbracket, m'_S, d'_S, pk_S, sk_S)$ is inconsistent because it contains an encryption of m_S instead of m'_S, and a fake proof of correct encryption d'_S. The semantic security of the cryptosystem ensures that even given m'_S, the encryption $\llbracket m_S \rrbracket$ is computationally indistinguishable from any $\llbracket m'_S \rrbracket$. In addition the designated-verifier property of the proof ensures that d'_S is computationally indistinguishable from a genuine proof. Hence such inconsistent transcripts are computationally indistinguishable from consistent transcripts. Then the simulated transcripts are computationally indistinguishable from the consistent real-world transcripts.

Case 4 (Coerced voter: before vote cast). The simulated transcripts are $\mathbb{M}_S = (\llbracket m_S \rrbracket, m_S, d_S, pk_S, sk_S)$ and $\mathbb{B}_S = (\llbracket v - m_S \rrbracket, p_S, v, m_S, r_S)$. The transcripts are consistent with the forced vote v. There are two different sets of real-world transcripts depending on the voter's coercion-response bit c.

If $c = 1$ then the voter behaves exactly as a corrupt voter and casts the forced vote v. As in Case 2 the simulated and real-world transcripts are identical because they are both consistent transcripts for the same forced vote.

If $c = 0$ then the voter follows the coercion-resistance strategy and casts a random vote. The real-world transcripts $\mathbb{M}_R = (\llbracket m_R \rrbracket, m'_R, d'_R, pk_R, sk_R)$ and $\mathbb{B}_R = (\llbracket v - m'_R \rrbracket, p_R, v, m'_R, r_R)$ are for the forced vote v. This is the reverse of Case 3: here the simulated transcripts are consistent but the real-world transcripts have the same inconsistency as the simulated transcripts in that case. Then for the same reasons the simulated and real-world transcripts are computationally indistinguishable.

Case 5 (Coerced voter: after vote cast). The simulated transcripts are $\mathbb{M}_S = (\llbracket m_S \rrbracket, m'_S, d'_S, pk_S, sk_S)$ and $\mathbb{B}_S = (\llbracket v - m'_S \rrbracket, p_S, v, m'_S, r_S)$. There are two different sets of real-world transcripts depending on the voter's coercion-response bit c.

If $c = 1$ then the voter behaves exactly as a corrupt voter and reveals the intended vote v. Since the real-world transcripts are consistent with v, the simu-

lated and real-world transcripts are exactly the same as in Case 3. Then for the same reasons they are computationally indistinguishable.

If $c = 0$ then the voter follows the coercion-resistance strategy and reveals the fake vote v. The real-world transcripts are $\mathbb{M}_R = (\llbracket m_R \rrbracket, m'_R, d'_R, pk_R, sk_R)$ and $\mathbb{B}_R = (\llbracket v - m'_R \rrbracket, p_R, v, m'_R, r_R)$. Now both simulated and real-world transcripts have the same inconsistency as the simulated transcripts in Case 3. Then such transcripts are computationally indistinguishable from consistent transcripts, and hence indistinguishable from each other.

Finally the corrupt and coerced voters who cast null votes are identical in the ideal and real worlds. Hence the voting output (the list of voters who cast non-null votes) is identical in the ideal and real worlds. □

Corollary 1 (Receipt-Freeness of Masked Ballot). *Under the stated assumptions, the Masked Ballot voting scheme is receipt-free.*

7.3 Complexity

Using typical costs of the underlying cryptographic primitives, we provide estimates of the computational and communication complexity. We use modular multiplication as the unit of measure and assume that a modular exponentiation costs $O(k)$ multiplications for a security parameter k. Encrypting a message and constructing (or verifying) a proof each requires a constant number of exponentiations. The number of modular multiplications performed has the same asymptotic complexity as the number of bits transferred, and so the computational complexity below also refers to the communication complexity.

The cost for each voter is $O(k)$. In the registration stage a voter verifies a designated-verifier proof, which costs $O(k)$. In the voting stage the voter encrypts a vote and constructs a proof of knowledge, which each costs $O(k)$.

In an election with V voters, the cost for the registrar is $O(Vk)$. The cost of performing or verifying the unmasking is also $O(Vk)$.

8 Conclusion

We introduced the Masked Ballot online voting scheme, which achieves receipt-freeness under the physical assumption that there are one-way untappable channels only before the election. Such channels can be realised through offline communication.

Masked Ballot presents a different set of trade-offs from previous receipt-free schemes. The main advantage of our approach is that it particularly suits Internet voting from any light-weight device with network access. A drawback is the need for the voter to securely obtain a single-use mask before each election.

Although several different physical assumptions can be used to construct receipt-free election schemes, none is ideal given the current state of widespread technology. Each approach has its own trade-offs, with advantages in certain settings but practical shortcomings in the general case. In the future tamper-proof

smart cards, completely anonymous channels or untappable channels may be easier to implement. But at present these physical assumptions are not generally practical for large-scale elections. Given that there are cryptographic solutions, the challenge is to make them more practical for widespread use.

Acknowledgements

We are grateful to Berry Schoenmakers and Tal Moran for very helpful discussions. We also thank the anonymous referees for many valuable comments.

References

1. Baudron, O., Fouque, P.A., Pointcheval, D., Stern, J., Poupard, G.: Practical multi-candidate election system. In: PODC, pp. 274–283 (2001)
2. Benaloh, J.C., Tuinstra, D.: Receipt-free secret-ballot elections (extended abstract). In: STOC, pp. 544–553 (1994)
3. Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty Computation from Threshold Homomorphic Encryption. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 280–299. Springer, Heidelberg (2001)
4. Cramer, R., Gennaro, R., Schoenmakers, B.: A Secure and Optimally Efficient Multi-Authority Election Scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
5. Damgård, I., Jurik, M.: A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)
6. Di Cosmo, R.: On Privacy and Anonymity in Electronic and Non Electronic Voting: the Ballot-As-Signature Attack (2007), `http://www.pps.jussieu.fr/~dicosmo/E-Vote/`
7. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31, 469–472 (1985)
8. Fouque, P.A., Poupard, G., Stern, J.: Sharing Decryption in the Context of Voting or Lotteries. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 90–104. Springer, Heidelberg (2001)
9. Hirt, M., Sako, K.: Efficient Receipt-Free Voting Based on Homomorphic Encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000)
10. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated Verifier Proofs and Their Applications. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996)
11. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: Atluri, V., di Vimercati, S.D.C., Dingledine, R. (eds.) WPES, pp. 61–70. ACM, New York (2005)
12. Lee, B., Boyd, C., Dawson, E., Kim, K., Yang, J., Yoo, S.: Providing Receipt-Freeness in Mixnet-Based Voting Protocols. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 245–258. Springer, Heidelberg (2004)
13. Lee, B., Kim, K.: Receipt-Free Electronic Voting Scheme with a Tamper-Resistant Randomizer. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 389–406. Springer, Heidelberg (2003)

14. Moran, T.: Cryptography by the People, for the People. PhD thesis, The Weizmann Institute of Science (2008), http://people.seas.harvard.edu/~talm/papers/thesis.pdf
15. Moran, T., Naor, M.: Receipt-Free Universally-Verifiable Voting with Everlasting Privacy. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 373–392. Springer, Heidelberg (2006)
16. Moran, T., Naor, M.: Split-ballot voting: everlasting privacy with distributed trust. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) ACM Conference on Computer and Communications Security, pp. 246–255. ACM, New York (2007)
17. Niemi, V., Renvall, A.: How to Prevent Buying of Votes in Computer Elections. In: Safavi-Naini, R., Pieprzyk, J.P. (eds.) ASIACRYPT 1994. LNCS, vol. 917, pp. 164–170. Springer, Heidelberg (1995)
18. Okamoto, T.: Receipt-Free Electronic Voting Schemes for Large Scale Elections. In: Christianson, B., Crispo, B., Lomas, T.M.A., Roe, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 25–35. Springer, Heidelberg (1998)
19. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
20. Sako, K., Kilian, J.: Receipt-Free Mix-Type Voting Scheme - A Practical Solution to the Implementation of a Voting Booth. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
21. Wen, R., Buckland, R.: Minimum Disclosure Counting for the Alternative Vote. In: Ryan, P.Y.A., Schoenmakers, B. (eds.) VOTE-ID 2009. LNCS, vol. 5767. Springer, Heidelberg (2009)