Minimum Disclosure Counting for the Alternative Vote

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. Although there is a substantial body of work on preventing bribery and coercion of voters in cryptographic election schemes for plurality electoral systems, there are few attempts to construct such schemes for preferential electoral systems. The problem is preferential systems are prone to bribery and coercion via subtle signature attacks during the counting. We introduce a minimum disclosure counting scheme for the alternative vote preferential system. Minimum disclosure provides protection from signature attacks by revealing only the winning candidate.

Keywords: Preferential voting, alternative vote, instant runoff voting, online elections, counting schemes.

1 Introduction

Most cryptographic election schemes in the literature are designed for plurality (first past the post) electoral systems, where each voter chooses a single candidate and the winner is the candidate who receives the most votes. But using these schemes for preferential electoral systems exposes voters to potential bribery and coercion through signature attacks. We introduce a preferential counting scheme that protects voters from such attacks.

Preferential electoral systems are widespread in Australia. Indeed, all Australian parliamentary elections at national and state levels use preferential systems. In most cases elections for the lower house use the alternative vote and elections for the upper house use the single transferable vote. The single transferable vote is a generalisation of the alternative vote for electing multiple candidates rather than a single candidate. These preferential systems are also common in Ireland and Malta, and they are sometimes used for local government elections in parts of New Zealand, the UK, and the USA. In this paper we only consider the alternative vote.

The aim of preferential electoral systems is to give voters greater scope in expressing their choices. The distinguishing feature of these systems is that voters *rank* candidates in order of preference. The alternative vote is one of the more complex instances of preferential systems because the counting procedure has many rounds of counting. In each round a candidate is excluded and the votes

P.Y.A. Ryan and B. Schoenmakers (Eds.): VOTE-ID 2009, LNCS 5767, pp. 122–140, 2009.

for this candidate are transferred to the *remaining* (not yet excluded) candidates according to the preferences given on the ballots for that candidate. We elaborate below on the mechanics of the counting procedure.

1.1 The Alternative Vote

The alternative vote, also known as preferential voting or instant runoff voting, is a majoritarian system for filling a single vacancy. To be elected, a candidate must receive a *majority* (more than half) of the votes.

Each ballot contains a sequence of preference votes. A voter fills out a ballot by ranking every candidate in consecutive numerical order starting from 1 for the first preference. A common variant is optional preferences, where voters assign a minimum of one preference but need not assign all preferences.

The counting takes place in recursive rounds. Each round is a 'last' past the post election. The election authorities tally the votes considering only the most preferred remaining candidate in each ballot. Then they exclude the candidate with the lowest round tally and transfer each vote for that candidate to the next preferred remaining candidate on the corresponding ballot. The next round is in effect a sub-election for the remaining candidates. The process recursively repeats until a single candidate remains. The authorities announce this candidate as the winner.

Notice that it is possible to stop the counting as soon as a candidate obtains a majority. The counting algorithm we described performs a complete distribution of the votes. For a given number of candidates, this algorithm has a constant number of counting rounds.

In the event that multiple candidates have the lowest tally in a round, there are a variety of tie-breaking rules used in practice to determine the last candidate, for instance randomly or by comparing tallies from previous rounds. All such rules eventually resort to breaking ties randomly or arbitrarily when a 'true' tie occurs. In this paper we simply resolve all ties randomly and in future work we describe more elaborate techniques for other common methods.

1.2 The Signature Attack

The information-rich nature of the ballots in preferential systems introduces the possibility of a signature attack, commonly referred to as the Italian attack due to its infamous use in Italian elections [3]. A signature attack potentially compromises voter anonymity during the counting and is an effective technique for bribing and coercing voters. Any election is open to this attack when the number of possible voting options is relatively large compared to the number of voters. Preferential elections are particularly vulnerable because the number of possible preference permutations is factorial in the number of candidates.

To 'sign' a preferential ballot, a voter can for example allocate the first preference to a particular candidate and use the ordering of the remaining preferences as a covert channel that contains a signature. Even for a relatively modest number of candidates and a large voting population, such a signature is highly likely

to be unique. For any prescribed first preference candidate, an election with C candidates has $(C-1)!$ possible covert signatures. The national upper house election in Australia has about 80 candidates, and so there are 79! possible signatures. Even if every atom in the universe voted in this election, there would still be a negligible probability that any randomly chosen signature would also be cast by another voter.

A covert signature of this form is revealed when the ballots are exposed during the counting, and it links the voter to the vote. In traditional paper elections, only election authorities and independent scrutineers appointed by the candidates can observe the ballots. We can only hope they are trustworthy. Alarmingly in Australia, recent moves to improve the transparency of elections have inadvertently made it trivial to perform signature attacks. In order to allow independent scrutiny in elections that use electronic counting, a ruling under Freedom of Information legislation [26] has led election authorities to publish every ballot electronically!

Subtle variations of the signature attack may still be feasible with only partial knowledge of the votes. An adversary can embed uncommon sequences of preferences in the signatures. Then the adversary can glean any available information about these contrived sequences to narrow down the set of *possible signatures*. For example if a candidate's tally remains the same across two rounds, then that candidate cannot be the next preference in any of the votes for the candidate excluded in the first of those rounds. In this way even when the adversary cannot identify exact signatures in the votes cast, it is still possible to determine that particular signatures are not present. This possibility may well be sufficient to allow coercion.

Election authorities frequently publish partial counting data such as the final placing of each candidate and all the round tallies for each candidate. But even releasing seemingly innocuous aggregate counting data has risks. Given the subtlety of signature attacks, it is not always immediately obvious whether disclosing particular information can have detrimental consequences.

Naturally much depends on the eventual distribution of the votes cast. Nevertheless an adversary can make some educated guesses, especially when there are few major candidates and many minor candidates. Several types of signature attacks on partial information are currently known [25]. But determining precisely what information is useful for mounting signature attacks and how effective are such attacks remains an open problem. Therefore revealing any information apart from the identity of the winning candidate can potentially expose voters to signature attacks.

Consequently, to eliminate the possibility of covert channels and intentional or accidental information leakage, the precautionary principle suggests that a conservative approach to secure counting is prudent. Ideally the counting process should be entirely secret and reveal only the winning candidate. The challenge for preferential systems lies in counting the votes in a secret yet universally verifiable manner.

1.3 Contributions

We introduce an alternative vote counting scheme that reveals only the identity of the winning candidate. We call this level of privacy minimum disclosure. This is the same notion of privacy used in Hevia and Kiwi's yes/no election scheme [12]. Minimum disclosure provides the strongest possible protection against attacks on counting information. In particular it prevents signature attacks on preferential systems.

Our scheme also satisfies the usual security requirements of correctness, universal verifiability, and robustness against corrupt authorities. The scheme can be used as an independent counting procedure or in conjunction with an existing online voting scheme.

The idea behind our counting scheme is to perform the counting on encrypted ballots. Each ballot is a list of encrypted preference votes in descending order of preference and each preference vote is for a distinct candidate.

The counting scheme uses a hide and seek paradigm to manipulate lists of ciphertexts without revealing anything about the order of a list. This approach repeatedly applies a three-step process.

1. Execute a distributed operation to conceal the ordering of the ciphertexts.
2. Execute another distributed operation to identify ciphertexts with certain properties.
3. Perform open operations, such as homomorphic addition, on the identified ciphertexts.

The distributed operations are cryptographic protocols that require the collaboration of multiple authorities. As such the main drawback of the scheme is the amount of work for the authorities. The extensive use of multiparty computation techniques is an inevitable trade-off in achieving both minimum disclosure and robustness, especially for electoral systems with elaborate counting algorithms, such as the alternative vote. In an election with A authorities, C candidates and V voters, the total computational and communication complexities for our scheme are $O(AC^2V)$.

1.4 Organisation

Section 2 discusses existing online voting schemes and preferential counting schemes. Section 3 defines the security model and Section 4 covers the necessary cryptographic building blocks. Section 5 describes the details of the minimum disclosure counting scheme and Section 6 proposes an optimised tallying protocol. Section 7 analyses the security and complexity of the scheme. Section 8 explains how to combine the counting scheme with common online voting schemes.

2 Related Work

In the general literature on cryptographic elections, preventing bribery and coercion centres on the requirements of receipt-freeness and coercion-resistance. Informally, receipt-freeness [2] means that voters who cast valid votes cannot be

bribed or coerced into proving how they voted because it is not possible for them to prove how they voted. Coercion-resistance [15] is the stronger requirement that voters cannot even prove whether they abstained, or cast invalid or random votes.

Receipt-free and coercion-resistant voting schemes focus on protecting voters from bribery and coercion during the voting itself. But they rarely consider the details of the counting. During the counting these schemes generally rely on statistical uncertainty in the votes to prevent voters from being identified by their votes. Every possible voting option must be likely to receive some votes from honest voters. For simple plurality elections, this is generally a reasonable assumption. But for preferential elections, it is not. This compromises receipt-freeness and coercion-resistance.

Contemporary online voting schemes have two main approaches to counting votes: public counting and private counting. Both methods reveal covert signatures and also absent signatures.

Voting schemes that perform public counting [15,17,19,20] implement only the voting stage of an election. Voters submit encrypted votes as their ballots. Then the authorities anonymise the ballots (generally through mix-nets) before decrypting them. To calculate the election result, any party can openly perform a known counting algorithm on the publicly revealed plaintext votes.

Conversely, voting schemes that perform private counting [1,2,13] implement both the voting and counting stages of a plurality election. Voters submit votes that are encrypted with an additively homomorphic cryptosystem. To calculate the election result, the authorities use the homomorphic property to combine all the encrypted votes into an encrypted tally for each possible voting option. Then they decrypt only these tallies. This approach maintains the privacy of individual votes because it publishes only the tallies. But as there are tallies for every voting option, it still reveals the same information about the votes as public counting. To calculate the result in a preferential election, any party can still openly perform the appropriate counting algorithm on the publicly revealed tallies.

To counter such signature attacks, Goh and Golle [7] propose an alternative vote counting scheme that only discloses the round tallies for each candidate. But there still remains some scope for signature attacks that exploit the round tallies to cull the set of possible signatures. Keller and Kilian [16] also propose a scheme with the same level of privacy.

Heather [11] describes a counting scheme for the more complex single transferable vote. In addition to revealing the candidates' round tallies, the transfer method also leaks partial sequences of preferences for previously excluded candidates. This extra information facilitates more effective signature attacks by significantly narrowing down the set of possible signatures.

Teague et al. [25] propose a single transferable vote counting scheme that achieves greater secrecy than the above schemes. When applied to the alternative vote, it conceals the round tallies and reveals only the order in which the

candidates are excluded. But the scheme relies on the trustworthiness of a single authority who can learn the plaintext contents of all the ballots.

3 Security Model

3.1 Participants and Adversary Model

The only participants in the counting scheme are the authorities who perform the counting. All communication is public and via an authenticated bulletin board. The security model is for a static, active adversary who can corrupt up to a threshold of the authorities.

3.2 Security Requirements

A secure counting scheme must satisfy the following requirements.

Minimum Disclosure. Apart from the identity of the winning candidate, no party or adversary gains any additional information about the candidates or the ballots than what was known before the counting commenced. The transcript of the counting is computationally indistinguishable from the transcript of the counting for any fake input list of the same number of valid votes that elects the same winning candidate. Revealing only the winning candidate prevents potential signature attacks including those that exploit partial counting information.

Correctness. All input votes are correctly counted and no other votes are counted.

Universal Verifiability. Any observer can confirm that the counting is correct.

Robustness. The counting tolerates the corrupt or faulty behaviour of any group of authorities up to a threshold.

Notice counting schemes do not consider requirements that only relate to voters during the preceding voting stage, for instance individual verifiability, robustness with respect to corrupt voters, and ensuring ballots are only cast by authentic voters. In some cases an additional integration procedure between the voting and counting may be necessary to transform the submitted ballots into a valid form for the counting.

3.3 Why So Secretive?

In current elections the authorities typically publish certain counting information for statistical purposes, and that published data alone is often sufficient for mounting signature attacks. So on the surface it might appear that in practice minimum disclosure is an unnecessarily strong requirement for online elections. However any more relaxed approach to secrecy in counting schemes can be problematic.

The amount and types of published data varies widely from election to election. But regardless of what information the authorities decide to reveal, a

counting scheme must not leak any partial information that aggravates signature attacks. In other words any leaked information must be insignificant.

The problem is there is currently no method to determine if specific partial information leakage is indeed insignificant. For example suppose a counting scheme leaks the identity of the excluded candidate in each round. Such information on its own would seem insignificant. Now suppose that the authorities decide to publish all the round tallies without identifying the candidates. Again such information would seem reasonably insignificant. But by combining these two types of partial information, an adversary can make strong correlations between all the tallies and candidates. This can substantially improve the chance of mounting successful signature attacks. Although this is a rather contrived example, it illustrates the difficulty in analysing the risk of partial information leakage. In fact the risk depends on context-specific factors such as the number of voters, the number of candidates and the a posteriori distribution of preference permutations.

In the absence of precise definitions of what partial information is sensitive, a cryptographic counting scheme should provide the strongest possible level of secrecy in order to suit any alternative vote election. Then if necessary the authorities can explicitly weaken the scheme to reveal exactly the desired counting data but nothing more. This approach mitigates the risk of additional unforeseen attacks.

4 Cryptographic Preliminaries

The minimum disclosure counting scheme relies on several distributed cryptographic protocols that provide privacy, universal verifiability and robustness. Rather than depending on specific instances of these protocols, we simply model them as ideal primitives. We state typical costs of the protocols in terms of a security parameter k.

4.1 Additively Homomorphic Threshold Cryptosystem

An additively homomorphic cryptosystem is a public-key cryptosystem that enables any party to efficiently compute an encryption of the sum of two messages given only the encryptions of the individual messages. For concreteness we describe the scheme using the Paillier cryptosystem [21], 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 has two homomorphic properties.

Addition. For plaintexts $m_1, m_2 \in \mathbb{Z}_n$,

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

Multiplication by a constant. For a plaintext $m \in \mathbb{Z}_n$ and constant $c \in \mathbb{Z}_n$,

$$\begin{aligned}c \boxdot \llbracket m \rrbracket &= (g^m r^n)^c \\ &= g^{cm} (r^c)^n \\ &= \llbracket cm \rrbracket \ .\end{aligned}$$

In the threshold version of Paillier [4,6], each authority has a share of the private key. A quorum of authorities must collaborate to decrypt any ciphertext. The decryption process is universally verifiable and reveals no additional information to any coalition of authorities smaller than the quorum.

To decrypt a ciphertext share and prove correctness, each authority performs $O(k)$ modular multiplications and broadcasts $O(k)$ bits. Publicly verifying and combining the shares of the A authorities costs $O(Ak)$.

4.2 Plaintext Equality and Inequality Tests

Plaintext equality and inequality tests compare the plaintexts of given ciphertexts without revealing the plaintexts. Given a pair of encrypted messages $\llbracket m_1 \rrbracket$ and $\llbracket m_2 \rrbracket$, a plaintext equality test [14] determines whether $m_1 = m_2$, and a plaintext inequality test [22,24] determines whether $m_1 > m_2$. In both cases the only public output is the boolean result of the test.

As for decryption in a threshold cryptosystem, the protocols to perform these tests are distributed operations that require the collaboration of a quorum of authorities, each of whom has a secret share of the private key. In fact the last step of these protocols requires a threshold decryption to reveal the result. The tests are universally verifiable and reveal no additional information to any coalition of authorities smaller than the quorum.

Plaintext equality tests have the same complexity as threshold decryptions. Plaintext inequality tests are more expensive, with the dominant additional cost being a bit extraction step described below. The total complexity for A authorities is $O(Alk)$ multiplications when the plaintexts are in a known range $\left[0, 2^l\right)$. In the counting scheme inequality tests are only used to compare encrypted tallies, and for V voters each tally is at most $l = \lceil \log_2 V \rceil$ bits.

4.3 Secure Bit Extraction

A bit extraction protocol [24] converts an encrypted message into separate encryptions of the individual bits of the message. Given an encrypted message $\llbracket m \rrbracket$, where $0 \leq m < 2^l$ and the binary representation is $m = m_0, \ldots, m_{l-1}$, the output is $\llbracket m_0 \rrbracket, \ldots, \llbracket m_{l-1} \rrbracket$. The bit extraction is a universally verifiable threshold

protocol and reveals no additional information to any coalition of authorities smaller than the quorum.

In the bit extraction protocol each authority privately performs $O(lk)$ multiplications including proofs of correctness. Publicly verifying and combining the individual results of the A authorities costs $O(Alk)$.

4.4 Verifiable Mix-Nets and Rotators

A verifiable re-encryption mix-net [8,10,18] is a series of servers that each randomly mix (by permuting and re-encrypting) a list of messages. In the case that each message is a tuple of ciphertexts rather than a single ciphertext, such as with preferential ballots, the mix-net re-encrypts every ciphertext in the tuple individually and preserves the structure of the tuple.

A verifiable rotator cascade [5] is similar to a mix-net. The difference is that each server randomly rotates (by cyclically shifting and re-encrypting) a list of messages. Rotation is particularly useful when it is necessary to preserve the relative ordering of the messages. Although it is possible to construct rotators using mix-nets [23], a direct implementation is more efficient.

Both mix-nets and rotator cascades conceal the correspondence between input and output messages as long as at least one server is honest. The mixing and rotating are both universally verifiable. For l ciphertexts, re-encrypting and proving correctness typically costs $O(lk)$ multiplications for each server. Publicly verifying the entire protocol when there are A servers costs $O(Alk)$.

5 The Minimum Disclosure Counting Scheme

The minimum disclosure counting scheme implements secure counting for alternative vote elections. It commences after the voting has finished and the authorities have performed all necessary ballot processing including the removal of invalid ballots.

We describe the counting scheme as a series of high-level protocols. Multiple authorities collaborate to execute the protocols. They post the result of every operation on an authenticated bulletin board with full revision tracking.

Each step in the protocol execution is either a distributed operation that requires a quorum of authorities to collaborate or a completely open operation that *any party* can compute from posted messages. The distributed operations are the distributed protocols described in Section 4, as well as operations constructed from those protocols. All other operations are open operations. A single arbitrary authority posts the results of the open operations but each authority individually performs the operations and verifies the correctness of the posted results.

Some of the open steps require a known encryption of a known plaintext message. In such cases, rather than probabilistically encrypting the plaintext with a secret randomness value, the authority *deterministically* encrypts the plaintext with a known randomness value of 1 and subsequent operations are used to add any necessary secret randomness.

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

5.1 Data Structures and Auxiliary Protocols

The counting stores the following encrypted information in list-based data structures:

Candidates	A list of encrypted remaining (non-excluded) candidates in random order.
Ballot	A list of encrypted preference votes. Each preference vote is for a remaining candidate, and the list ordering represents a voter's preferences for the candidates in descending order.
Ballots	A list of ***Ballot*** objects each of which corresponds to a valid vote cast by a voter.
Counters	A dictionary of encrypted candidate-tally mappings each of the form $([\![c]\!], [\![t]\!])$, where the key c is a candidate and t is the tally for c in the current round. We represent the dictionary as a list of encrypted pairs.

In addition to the `count`, `tally` and `exclude` protocols specified in the following subsections, several auxiliary protocols are used to manipulate the encrypted data:

`pet` $([\![m_1]\!], [\![m_2]\!])$	Perform a plaintext equality test on the input ciphertexts.
`pit` $([\![m_1]\!], [\![m_2]\!])$	Perform a plaintext inequality test on the input ciphertexts.
`mix` $(\boldsymbol{List})$	Randomly permute and re-encrypt a list of messages. Each message can be a single ciphertext or a pair of ciphertexts.
`rotate` $(\boldsymbol{List})$	Randomly cyclically shift and re-encrypt a list of messages. Each message can be a single ciphertext or a pair of ciphertexts.
`append` $(\boldsymbol{List}, m)$	Append the message m to ***List***. The message can be a single ciphertext or a pair of ciphertexts.
`remove` $(\boldsymbol{List}, [\![m]\!])$	Remove all ciphertexts matching $[\![m]\!]$ from ***List***. In the counting scheme only one element will be removed by this protocol. We implement the `remove` protocol by executing `pet` $([\![m]\!], [\![item]\!])$ for each $[\![item]\!] \in \boldsymbol{List}$. If ***List*** is a dictionary, then remove the mapping corresponding to the key m. In this case we use `pet` to compare $[\![m]\!]$ with the encrypted keys in ***List***.

These protocols reveal no information about their encrypted inputs apart from the returned values, except that the `remove` protocol also reveals the *position* of the removed item in the list. In this case prior `mix` or `rotate` operations ensure the revealed position is random.

5.2 Count Protocol

The `count` protocol (Protocol 1) is the top-level protocol for calculating the election result. It invokes several sub-protocols to count the ballots. The inputs are the lists $\boldsymbol{Candidates}$ and $\boldsymbol{Ballots}$. The output is the identity of the winning candidate.

```
 1: count(Candidates, Ballots)
 2:     if Candidates has 1 remaining candidate [[c]]
 3:         decrypt([[c]]) and post c
 4:     else
 5:         Counters ← tally(Candidates, Ballots)
 6:         Counters ← mix(Counters)
 7:         [[c_ex]] ← min(Counters)
 8:         Ballots ← exclude(Ballots, [[c_ex]])
 9:         Candidates ← remove(Candidates, [[c_ex]])
10:         count(Candidates, Ballots)
```

Protocol 1: Counting for the alternative vote

Before the counting commences the authorities create the list $\boldsymbol{Candidates}$. To do this they deterministically encrypt each valid candidate and then mix the ciphertexts. Additionally each ballot in $\boldsymbol{Ballots}$ must contain an encrypted preference vote for each valid candidate. In Section 8 we discuss how to ensure the input ballots are valid.

The `count` protocol is a recursive procedure that performs a complete distribution of the votes, continuing until there is only one remaining candidate. Each recursive step corresponds to a round of counting that invokes the `tally` protocol to privately calculate the round tally and then the `exclude` protocol to privately exclude a candidate.

The excluded candidate c_{ex} is the candidate with the minimum round tally. The `min` protocol locates $[\![c_{ex}]\!]$ in $\boldsymbol{Counters}$ by executing `pit` $([\![t_i]\!], [\![t_j]\!])$ for pairs of candidate-tally mappings $([\![c_i]\!], [\![t_i]\!]), ([\![c_j]\!], [\![t_j]\!]) \in \boldsymbol{Counters}$. Tracking the counter with the current minimum tally and updating the minimum counter according to the result of `pit` requires $(C-1)$ invocations of `pit` for C counters. As `min` reveals a partial ordering of the counters, a preceding `mix` operation is necessary to ensure the revealed ordering is random. The `min` protocol resolves ties randomly and avoids revealing whether any ties occur.

5.3 Tally Protocol

The `tally` protocol (Protocol 2) calculates the round tally for each remaining candidate without revealing the tallies, the candidates, or the contents of the ballots. The inputs are the lists $\boldsymbol{Candidates}$ and $\boldsymbol{Ballots}$. The output is the list $\boldsymbol{Counters}$, which contains the round tally for each remaining candidate.

The protocol starts by initialising $\boldsymbol{Counters}$. Each encrypted key is an exact copy of an encrypted candidate $[\![c]\!] \in \boldsymbol{Candidates}$, and each encrypted tally is

```
 1: tally(Candidates, Ballots)
 2:     Counters ← {}
 3:     for each ⟦c⟧ ∈ Candidates
 4:         ⟦t⟧ ← ⟦0⟧
 5:         Counters ← append(Counters, (⟦c⟧, ⟦t⟧))
 6:     for each Ballot ∈ Ballots
 7:         Counters ← mix(Counters)
 8:         ⟦v⟧ ← the highest preference vote in Ballot
 9:         (⟦c⟧, ⟦t⟧) ← lookup(Counters, ⟦v⟧)
10:         ⟦t⟧ ← ⟦t⟧ ⊞ ⟦1⟧
11:     return Counters
```

Protocol 2: Calculating the round tallies

the deterministic encryption of the initial tally 0. Subsequent mixing introduces secret randomness into all the ciphertexts.

The protocol iteratively calculates the tallies using the highest preference vote (the head of the list) in each ballot. For each ballot the protocol locates the correct counter and then increments it by homomorphically adding the deterministic encryption of 1. The `lookup` protocol locates the matching counter $(\llbracket c \rrbracket, \llbracket t \rrbracket) \in \boldsymbol{Counters}$ by executing `pet` $(\llbracket v \rrbracket, \llbracket c \rrbracket)$ for each $(\llbracket c \rrbracket, \llbracket t \rrbracket) \in \boldsymbol{Counters}$. Since `lookup` reveals the position of the incremented counter, a prior `mix` operation is necessary to ensure the revealed position is random.

5.4 Exclude Protocol

The `exclude` protocol (Protocol 3) deletes the excluded candidate from each ballot without revealing the identity of the excluded candidate, or the contents of any ballot. The inputs are the list $\boldsymbol{Ballots}$ and the encrypted excluded candidate $\llbracket c_{ex} \rrbracket$. The output is the updated list $\boldsymbol{Ballots}$.

```
 1: exclude(Ballots, ⟦c_ex⟧)
 2:     for each Ballot ∈ Ballots
 3:         Ballot ← append(Ballot, ⟦m⟧)
 4:         Ballot ← rotate(Ballot)
 5:         Ballot ← remove(Ballot, ⟦c_ex⟧)
 6:         Ballot ← rotate(Ballot)
 7:         Ballot ← remove(Ballot, ⟦m⟧)
 8:         Ballot ← restore(Ballot)
 9:     return Ballots
```

Protocol 3: Deleting the excluded candidate from each ballot

For each $\boldsymbol{Ballot}$ the protocol executes `remove` to delete the encrypted preference vote $\llbracket v \rrbracket$ with $v = c_{ex}$. However, as the `remove` protocol leaks the position

of the removed item, it is necessary to conceal the position of $[\![v]\!]$. Hence the `exclude` protocol first executes a `rotate` operation. Then although the randomly shifted position of $[\![v]\!]$ is known at the instant of deletion, there is no correlation with its original position in the ballot.

At this point `exclude` must undo the rotation to return the ballot to its original ordering. To permit this, prior to the original `rotate` the protocol appends a deterministically encrypted marker $[\![m]\!]$ (where m is a publicly known and invalid preference) to the end of the ballot. Then afterwards it executes `remove` to delete $[\![m]\!]$. This also reveals the end of the ballot and the `restore` operation simply shifts the list of preferences back to its original ordering. Note the `rotate` before the marker is removed conceals the relative positions of $[\![v]\!]$ and $[\![m]\!]$.

5.5 Optional Preferences Variant

A common variation in preferential systems is that voters are only required to assign one preference, and the remaining preferences are optional. The minimum disclosure counting scheme can also accommodate this situation. In this case we still require that ballots contain an encrypted preference vote for each valid candidate. Every ballot simply contains an additional encrypted null candidate $[\![\perp]\!]$ as a terminator after the last desired preference. The voter, or possibly the voting application, enters the remaining preferences in arbitrary order after $[\![\perp]\!]$.

The only change needed in the counting scheme is in the `count` protocol. The list ***Candidates*** now contains $[\![\perp]\!]$ and the recursion terminates when there are two remaining candidates (the winner and $\perp$). To conceal exhausted ballots the `tally` protocol treats the null candidate the same as any other candidate. But the counting must disregard the null candidate's tally in order to avoid excluding the null candidate. Hence immediately before executing `min`, the `count` protocol must perform an additional `remove` (***Counters***, $[\![\perp]\!]$) step.

6 Optimised Tally Protocol

We can optimise the `tally` protocol by using a radix M representation to encode each candidate as in Baudron et al.'s voting scheme [1]. Let C be the number of candidates, V be the number of voters, $L = \lceil \log_2 V \rceil$ and $M = 2^L$. Then we encode the ith candidate as $c_i = M^{i-1}$ for $i \in \{1, \ldots, C\}$.

The optimised `tally` protocol (Protocol 4) homomorphically adds the highest preference vote in each ballot to compute a single encryption of the sum $s = \sum_{i=1}^{C} t_i c_i$, where t_i is the tally for c_i. Under the radix $M = 2^L$ encoding, s is an integer of length CL bits where the ith block of L bits corresponds to the bit representation of t_i.

The protocol uses an `extract` operation (see Section 4.3) to convert $[\![s]\!]$ into an encrypted bit representation $([\![b_0]\!], \ldots, [\![b_{CL-1}]\!])$. The protocol reconstructs each encrypted tally $[\![t_i]\!]$ from its bit representation and forms the encrypted candidate-tally pairs as in the original `tally` protocol. The only difference is that ***AllCounters*** contains a counter for each valid candidate including previously

```
 1: tally(Candidates, Ballots)
 2:     [[s]] ← [[0]]
 3:     for each Ballot ∈ Ballots
 4:         [[v]] ← the highest preference vote in Ballot
 5:         [[s]] ← [[s]] ⊞ [[v]]
 6:     ([[b_0]], ..., [[b_{CL-1}]]) ← extract([[s]], CL)
 7:     AllCounters ← {}
 8:     for each i ∈ {1, ..., C}
 9:         [[t_i]] ← [[0]]
10:         for each j ∈ {0, ..., L − 1}
11:             [[t_i]] ← [[t_i]] ⊞ (2^j ⊡ [[b_{(i−1)L+j}]])
12:         AllCounters ← append(AllCounters, ([[c_i]], [[t_i]]))
13:     AllCounters ← mix(AllCounters)
14:     Counters ← {}
15:     for each remaining candidate [[r]] ∈ Candidates
16:         ([[c]], [[t]]) ← lookup(AllCounters, [[r]])
17:         Counters ← append(Counters, ([[c]], [[t]]))
18:     return Counters
```

Protocol 4: Optimised tallying

excluded candidates. The final part of the protocol filters out the counters for excluded candidates to produce ***Counters*** for only the remaining candidates.

Note that this optimisation is only appropriate when the sum s fits in the plaintext space, that is $C \lceil \log_2 V \rceil < k$ for a k-bit length public key. Of course it is always possible to increase k but the increased work in performing operations under a larger key may not be worthwhile. In most cases there should be no problem because C tends to be reasonably small (typically less than 20) in alternative vote elections.

The optimised `tally` protocol is essentially an efficient minimum disclosure counting scheme for plurality systems. All that remains is to mix the counters then locate the counter with the maximum tally and decrypt the winning candidate. Locating the maximum counter is analogous to the `min` protocol in Section 5.2.

7 Analysis

7.1 Security

The counting scheme satisfies minimum disclosure, correctness, universal verifiability and robustness.

Minimum Disclosure. Minimum disclosure follows from the privacy properties of the underlying primitives and the application of the hide and seek paradigm. Apart from the final decryption to reveal the winning candidate, only the plaintext equality and inequality tests potentially leak any information.

The equality tests are used to construct the `remove` and `lookup` protocols (Sections 5.1 and 5.3). Both protocols reveal only the position of an encrypted message in a list of ciphertexts. The preceding `mix` or `rotate` ensures the revealed position is random.

The inequality tests are used to construct the `min` protocol (Section 5.2). This protocol reveals a partial ordering of a list of ciphertexts according to the plaintexts. The preceding `mix` ensures the revealed partial ordering is random.

Therefore all the leaked information is random and reveals nothing about the private counting state.

Correctness. The high-level description of the counting scheme doubles as a specification of a (non-cryptographic) counting algorithm for the alternative vote.

Universal Verifiability and Robustness. The authenticated bulletin board enables any party to examine and verify every step of the protocol execution. Each step is universally verifiable and robust. There are two types of steps: distributed operations and open operations.

1. A distributed operation requires the authorities to post non-interactive zero-knowledge proofs that explicitly provide verifiability and robustness. Any party can then check whether the operation is correct.
2. An open operation is a known deterministic function on previously posted messages. An arbitrary authority posts the result. Any party can verify the open operation by independently computing the function as specified in the protocol and then comparing the result to the posted result. Robustness follows as each authority can also compute the result and compare it to the posted result. In the event of inconsistencies, all the authorities post their results. The correct result is the one that is identical for the quorum of honest authorities.

7.2 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. For all the primitives used, 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.

In an election with A authorities, C candidates and V voters, the total cost of performing or verifying the counting is $O(AC^2Vk)$. Each authority individually performs $O(C^2Vk)$ operations. Publicly verifying and combining the individual results of the authorities has an additional cost of $O(AC^2Vk)$.

The total $O(AC^2Vk)$ complexity arises from the $O(ACVk)$ cost per counting round, with $\big(C-1\big)$ rounds in total. In each round the `tally` protocol costs each authority $O(V)$ mix operations on $O(C)$ ciphertexts, $O(CV)$ plaintext equality tests, and $O(ACVk)$ verification operations, resulting in an asymptotic complexity of $O(ACVk)$. The `exclude` protocol has the same cost. The `min` protocol does

not affect the complexity as it only performs $O(C)$ plaintext inequality tests, each at a cost of $O((A \log V)\, k)$.

Although the optimised `tally` protocol only costs $O(V + (AC \log V)\, k)$ due to the $O(V)$ homomorphic additions and the $O((AC \log V)\, k)$ extraction of $C \log V$ bits, the overall complexity remains the same because the `exclude` protocol still costs $O(ACVk)$ per round.

8 Integration with Voting Schemes

We can construct an end to end solution for cryptographic preferential elections by combining the minimum disclosure counting scheme with an existing receipt-free or coercion-resistant online voting scheme (see Section 2). A common approach in voting schemes is for the ballot to contain an encrypted vote for a single candidate. The voter provides a non-interactive zero-knowledge proof of vote validity so that anyone can verify the ballot is for a valid candidate. Adapting such a voting scheme for preferential voting requires the following modifications.

1. Each voter casts a ballot containing a list of encrypted preference votes in descending order of preference. As in the optimised `tally` protocol (Section 6) we use the radix M representation to encode candidates. The voter also provides an explicit proof of *preferential* vote validity.
2. After removing all unauthentic ballots and ballots with incorrect proofs of vote validity, the authorities use the minimum disclosure counting scheme to compute the election result.

Since a ballot must now contain an encrypted vote for each valid candidate, the proof of preferential vote validity is more complex than for a plurality scheme. First the voter must prove that each encrypted preference vote $[\![v_i]\!]$ in the ballot is for a valid candidate, for instance using Damgård and Jurik's proof [4].

Next the voter must show that each preference vote v_i is for a distinct candidate. An efficient solution is Groth's proof of vote validity for the Borda vote [9]. The proof is for a ballot that consists of a single encrypted preferential vote $[\![v]\!]$. A valid vote is of the form $v = \sum_{i=1}^{C} \pi(i)\, v_i$, where C is the number of candidates, π is a permutation of the rankings $1, \ldots, C$, and v_i is a preference vote for a valid candidate.

To use this proof the voter must first convert the list of encrypted preference votes $[\![v_1]\!], \ldots, [\![v_C]\!]$ into a single encrypted preferential vote $[\![v]\!]$. In addition the conversion must be universally verifiable. The homomorphic cryptosystem provides a natural solution as anyone can compute $[\![v]\!] = (1 \boxdot [\![v_1]\!]) \boxplus (2 \boxdot [\![v_2]\!]) \boxplus \ldots \boxplus (C \boxdot [\![v_C]\!])$.

Casting a ballot is efficient for the voter. The cost of creating a ballot is $O(Ck)$ and the total cost of constructing or verifying a proof of vote validity is $O((C \log C)\, k)$.

9 Conclusion

We introduced a minimum disclosure counting scheme for secure counting in alternative vote elections. Its main contribution is that it achieves privacy in the counting by performing operations only on ciphertexts and decrypting only the winning candidate. Hence it thwarts signature attacks for bribery and coercion. The scheme provides stronger security than both contemporary cryptographic counting schemes and traditional manual counting. It can function as a standalone counting scheme or can be combined with an online voting scheme to form a complete online election scheme.

Even if the election authorities deliberately weaken the counting scheme to reveal specific counting data, minimum disclosure in the protocol is still important in order to ensure there is no additional and unintended information leakage. This can be especially relevant when initially adopting cryptographic counting. For instance it may be desirable to use the counting scheme in parallel with manual counting and then compare the results. Since the manual count must resolve any ties using the same random choices as the counting scheme, then in this case it would be necessary to reveal some counting data such as the order of exclusions.

Worldwide, plurality electoral systems are the most common for government elections. Interestingly, preferential electoral systems are gradually becoming more widespread. New Zealand and Scotland have recently adopted preferential systems for some elections. In parts of Canada, the UK and the USA, there is currently a push to switch to preferential systems.

Historically, a barrier to the adoption of preferential systems has been the complexity of manual counting. But now computers can automate the counting. Indeed many preferential elections already use electronic counting, where election authorities manually enter votes from paper ballots into an electronic database and a computer calculates the result. In fact for some preferential systems, such as the version of the single transferable vote recently introduced for local elections in New Zealand, the counting algorithm is so complicated that manual counting is infeasible.

Electronic counting offers many advantages. However the shift towards naive electronic counting without cryptographic safeguards is an alarming trend. One serious concern is unauthorised access to the voting data. Compromising the electronic database of plaintext votes opens the door to the potential for large-scale bribery and coercion of voters through signature attacks. Another issue is the lack of verifiability. It is notoriously difficult to detect flaws in the software implementation and the hardware. Publicly releasing the complete voting data for independent verification, as required in Australia, violates the secret ballot and jeopardises effective democracy. Therefore verifiably secure cryptographic approaches to preferential counting have an important role to play in both paper and electronic elections.

Cryptographic counting for the alternative vote raises two open problems. First, what is the optimal complexity? For C candidates and V voters, the lower bound is at least $O(CV)$ from the $O(C)$ rounds and $O(V)$ distributed

ballot operations per round. Intuitively the limiting factor is the exclusion of a candidate without revealing its identity or ranking in any ballot. Regardless of the ballot representation, this seems to require $O(C)$ work per ballot. Then the optimal cost would be $O(C^2V)$.

Second, is it possible to precisely define what counting information is sensitive? In the context of signature attacks it appears very difficult to develop an appropriate definition. However a weaker requirement than minimum disclosure might enable coercion-resistant schemes of lower cost.

Acknowledgements

We thank Berry Schoenmakers for suggesting the use of verifiable rotators to greatly simplify an earlier version of our scheme. We also thank Vanessa Teague for helpful discussions. We are grateful to 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. 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/`
4. 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)
5. de Hoogh, S., Schoenmakers, B., Skoric, B., Villegas, J.: Verifiable Rotation of Homomorphic Encryptions. In: Jarecki, S., Tsudik, G. (eds.) Public Key Cryptography. LNCS, vol. 5443, pp. 393–410. Springer, Heidelberg (2009)
6. 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)
7. Goh, E.J., Golle, P.: Event Driven Private Counters. In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 313–327. Springer, Heidelberg (2005)
8. Groth, J.: A Verifiable Secret Shuffle of Homomorphic Encryptions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 145–160. Springer, Heidelberg (2002)
9. Groth, J.: Non-interactive Zero-Knowledge Arguments for Voting. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 467–482. Springer, Heidelberg (2005)
10. Groth, J., Ishai, Y.: Sub-linear Zero-Knowledge Argument for Correctness of a Shuffle. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 379–396. Springer, Heidelberg (2008)
11. Heather, J.: Implementing STV securely in Pret a Voter. In: CSF, pp. 157–169. IEEE Computer Society, Los Alamitos (2007)

12. Hevia, A., Kiwi, M.A.: Electronic jury voting protocols. Theor. Comput. Sci. 321, 73–94 (2004)
13. 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)
14. Jakobsson, M., Juels, A.: Mix and Match: Secure Function Evaluation via Ciphertexts. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 162–177. Springer, Heidelberg (2000)
15. 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)
16. Keller, J., Kilian, J.: A Linked-List Approach to Cryptographically Secure Elections Using Instant Runoff Voting. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 198–215. Springer, Heidelberg (2008)
17. 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)
18. Nguyen, L., Safavi-Naini, R., Kurosawa, K.: Verifiable shuffles: a formal model and a Paillier-based three-round construction with provable security. Int. J. Inf. Sec. 5, 241–255 (2006)
19. 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)
20. 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)
21. 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)
22. Parkes, D.C., Rabin, M.O., Shieber, S.M., Thorpe, C.A.: Practical secrecy-preserving, verifiably correct and trustworthy auctions. In: Fox, M.S., Spencer, B. (eds.) ICEC. ACM International Conference Proceeding Series, vol. 156, pp. 70–81. ACM, New York (2006)
23. Reiter, M.K., Wang, X.: Fragile mixing. In: Atluri, V., Pfitzmann, B., McDaniel, P.D. (eds.) ACM Conference on Computer and Communications Security, pp. 227–235. ACM, New York (2004)
24. Schoenmakers, B., Tuyls, P.: Efficient Binary Conversion for Paillier Encrypted Values. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 522–537. Springer, Heidelberg (2006)
25. Teague, V., Ramchen, K., Naish, L.: Coercion-Resistant Tallying for STV Voting. In: Dill, D.L., Kohno, T. (eds.) EVT, USENIX Association (2008)
26. Victorian Civil and Administrative Tribunal: van der Craats v Melbourne City Council [2000] VCAT 447 (2000), `http://www.austlii.edu.au/au/cases/vic/VCAT/2000/447.html`