Information-Theoretically Secure Voting Without an Honest Majority

Anne Broadbent and Alain Tapp

Département d'informatique et de recherche opérationnelle
Université de Montréal, C.P. 6128, Succ. Centre-Ville
Montréal (QC), H3C 3J7 Canada
{broadbea, tappa}@iro.umontreal.ca

Abstract. We present three voting protocols with unconditional privacy and information-theoretic correctness, without assuming any bound on the number of corrupt voters or voting authorities. All protocols have polynomial complexity and require private channels and a simultaneous broadcast channel. Our first protocol is a basic voting scheme which allows voters to interact in order to compute the tally. Privacy of the ballot is unconditional, but any voter can cause the protocol to fail, in which case information about the tally may nevertheless transpire. Our second protocol introduces voting authorities which allow the implementation of the first protocol, while reducing the interaction and limiting it to be only between voters and authorities and among the authorities themselves. The simultaneous broadcast is also limited to the authorities. As long as a single authority is honest, the privacy is unconditional, however, a single corrupt authority or a single corrupt voter can cause the protocol to fail. Our final protocol provides a safeguard against corrupt voters by enabling a verification technique to allow the authorities to revoke incorrect votes. We also discuss the implementation of a simultaneous broadcast channel with the use of temporary computational assumptions, yielding versions of our protocols achieving everlasting security.

Keywords: multiparty computation, election protocol, dining cryptographers, information-theoretic security, election authorities, ballot verification.

1 Introduction

Multiparty secure computation enables a group of n participants to collaborate in order to compute a function on their private inputs. Assuming that private random keys are shared between each pair of participants, every function can be securely computed if and only if less than $n/3$ participants are corrupt; this fundamental result is due to David Chaum, Claude Crépeau and Ivan Damgård [CCD88] and to Michael Ben-Or, Shafi Goldwasser and Avi Wigderson [BOGW88]. When a broadcast channel is available, the results of Tal Rabin and Michael Ben-Or [RBO89] tell us that this proportion can be improved to $n/2$.

Among all functions that can be computed with these general-purpose protocols, perhaps the one that has the most obvious application is voting. If we have a guarantee on the proportion of honest participants, a secure voting protocol based only on pairwise private channels can be implemented (if, in addition to this, we have a broadcast channel, then we can tolerate more cheaters). Here, we are interested in the case where no such guarantee is available. The first protocol for voting that is information-theoretically secure even in a presence of a majority of dishonest participants was presented in [BT07]. Along with the use of private communication, the protocol uses a simultaneous broadcast channel. In this extended abstract, we first give a new presentation of the original protocol, followed by two protocols which present significant improvements on the original one. Although our initial motivation was of theoretical nature, we believe that this work may lead to interesting practical applications.

All three protocols are obtained from two simple yet powerful observations. First, if the dinning cryptographer's protocol [Cha88] is used to compute the parity function and is implemented with a simultaneous broadcast channel, then it is perfect. The second observation is that if a string of n bits is shared among n participants is such that the parity of the n bits is random (and unknown), then it is impossible for any strict subset of participants to locally derandomize this parity.

In our first protocol, we assume that each pair of voters is connected by a private authentic channel. In our second and third protocols, we relax this assumption by introducing *voting authorities*. The assumption then becomes that there are private and authentic channels only between voters and the authorities and among the authorities themselves.

All three protocols require a simultaneous broadcast channel [CGMA85,HM05], which, for our purpose, is a collection of broadcast channels where the input of one participant cannot depend on the input of any other participant. This could be achieved if all participants *simultaneously* performed a broadcast. In the context of our second and third protocols, a simultaneous broadcast among the authorities is sufficient.

It is not uncommon in multiparty computation to allow additional resources, even if these resources cannot be implemented with the threshold on the honest participants (the results of [RBO89] which combine a broadcast channel with $n/2$ honest participants being the most obvious example). Our work suggests that a simultaneous broadcast channel is an interesting primitive to study in this context. Furthermore, given a resource to implement bit commitment, we can implement a simultaneous broadcast: all participants commit to their values, and then all participants open these values. Since bit commitment can be implemented based on the laws of relativity [Ken99] (or more precisely, based on the postulate that information cannot travel faster then the speed of light), we conclude that simultaneous broadcast can also be achieved in this model. It may also be possible to directly implement a simultaneous broadcast using the laws of relativity.

Since a simultaneous broadcast channel can be achieved using bit commitment, which itself can be implemented with computational assumptions, we can replace in all our protocols the use of a simultaneous broadcast channel with temporary computational assumptions. Our protocols then provide *everlasting* security: as long as the computational assumptions are not broken *during* the execution of the protocol (more precisely, during the simulation of the simultaneous broadcast), the security of the protocols is perfect. Note that the privacy of individual votes remains perfect even if these computational assumptions are broken during the protocol.

1.1 Common Features to All Protocols

Our voting protocols involve n voters, each casting a ballot for a single choice among m candidates. The goal of the protocols is to faithfully count the number of ballots in favour of each candidate in such a way that voter's ballots remain private, honest ballots are counted and dishonest voters cannot influence the vote any more than by honestly voting. The protocols we present are based on a technique presented in [BT07]. The first protocol involves only the voters but the last two involve r voting authorities. In all protocols, dishonest participants can make the protocol fail (in our last protocol, only dishonest authorities can achieve this). All three protocols use probabilistic techniques to correctly evaluate the tally for each candidate. For this reason, the protocols are only correct with probability $1 - 2^{-\Omega(s)}$, with s being a chosen security parameter.

We present our protocols in the regular setup where each voter casts a ballot with a choice for a single candidate. Our protocols can easily be adapted to allow any number of voices per ballot (allowing, for instance, each voter to either choose two candidates, or to vote twice for the same candidate). We can also add a dummy candidate to allow voters to honestly cancel their ballots.

1.2 Summary of Results

All three protocols are exclusively based on private authentic channels and a simultaneous broadcast channel. In the first protocol, no assumption is made on the number of honest voters and in the last two, the only assumption is that at least one authority is honest. Under these assumption, our protocols provide perfect privacy and correctness. This was believed to be impossible [Gra08]. The major drawback is that any dishonest participant can make any protocol fail (except in our third protocol, where only dishonest authorities can make the protocol fail).

Protocols 2 and **3** make use of voting authorities. If we group the authorities together, they act as a trusted third party, which means that collectively they can violate privacy and correctness of the protocol. However, taken individually, both privacy and correctness are guaranteed as long as a *single* authority is honest. This suggests that in practice, authorities could be chosen to represent different interest groups, with each voter needing to trust only a single authority (note that it is not necessary for the voters to trust the *same* authority!).

It is common in multiparty computation to compare an implementation of a functionality with its *ideal* functionality. This ideal functionality is represented as a black box, accepting private inputs from each participant and privately communicating the function evaluation on these private inputs back to each participant. We now review the main features of each protocol.

Basic Voting (section 2)

- Only voters are involved in the protocol.
- A coalition of dishonest voters can only learn through the protocol what they would learn in the ideal functionality, and this even (and also) if the protocol fails.
- A single dishonest voter can make the protocol fail.
- If the protocol does not fail, then it is consistent with all ballots of the honest voters and some assignment of ballots for the dishonest voters.
- Dishonest voters cannot vote adaptively.

Voting with Authorities (section 3)

- Voters and a small number of authorities are involved in the protocol.
- Voters only interact with authorities.
- If at least one authority is honest, a coalition of dishonest voters and authorities can only learn what they would learn in the ideal functionality, and this even (and also) if the protocol fails.
- A single dishonest voter or authority can make the protocol fail.
- If at least one authority is honest and if the protocol does not fail, then it is consistent with all ballots of the honest voters and some assignment of ballots for the dishonest voters.
- If at least one authority is honest, a coalition of dishonest voters and authorities cannot vote adaptively.

Voting with Authorities and Verification (section 4)

- Voters and a small number of authorities are involved in the protocol.
- Voters only interact with authorities.
- If at least one authority is honest, a coalition of dishonest voters and authorities can only learn what they would learn in the ideal functionality, and this even (and also) if the protocol fails.
- No coalition of voters alone can make the protocol fail.
- A single dishonest authority can make the protocol fail.
- If at least one authority is honest and if the protocol does not fail, then it is consistent with all ballots of the honest voters and some assignment of ballots for the dishonest voters.
- If at least one authority is honest, a coalition of dishonest voters and authorities cannot vote adaptively.
- Dishonest voters voting inappropriately will have their ballot revoked.
- A dishonest authority can choose to revoke the ballot of an honest voter.
- When a ballot is revoked, all voters and authorities know about it.

2 Basic Voting Protocol

We present a protocol that allows n voters to conduct an m-candidate vote. First, some notation: we say that participants share a *distributed bit with value b* if each participant holds a bit and the parity (binary XOR) of all bits is b. Within a group of n participants, we say that a voter *constructs* a distributed bit with value b if he chooses $b_i \in_R \{0,1\}$ such that $\bigoplus_{i=1}^{n} b_i = b$ and sends privately b_i to participant i. The values $\{b_i\}$ $(i = 1, \ldots n)$ are called *shares*. For now, voters create distributed bits among themselves. In sections 3 and 4, voters will create distributed bits among authorities. Our basic protocol is given as **Protocol 1**.

Protocol 1 Basic voting protocol

Input: $x_i \in \{1, \ldots, m\}$ and security parameter s
Output: for $k = 1$ to m, $y[k] = |\{x_j \mid x_j = k\}|$

Phase A (cast)
For each candidate $k = 1$ to m,

1. Each voter i sets the value of $n^2 s$ bits p_{ijk} $(j = 1, \ldots, n^2 s)$ in the following way: if $x_i \neq k$, then all bits are 0; otherwise, exactly ns bits (a fraction $1/n$ of the total) are randomly chosen such that $p_{ijk} = 1$ and the rest such that $p_{ijk} = 0$.
2. For each $j = 1, \ldots, n^2 s$, each voter i constructs a distributed bit with value p_{ijk}. Let the shares of each distributed bit be denoted $\{p_{ijk\ell}\}$ $(\ell = 1, \ldots n)$

Phase B (broadcast)
For every j and k, each voter ℓ, computes the parity of all received bits, $q_{jk\ell} = \bigoplus_{i=1}^{n} p_{ijk\ell}$. All bits are then simultaneously broadcast.
Phase C (tally)
To compute the tally, $y[k]$, for each value $k = 1, \ldots, m$, each voter sets: $v[k]_j = \bigoplus_{\ell=1}^{n} q_{jk\ell}$, $\sigma[k] = \sum_{j=1}^{n^2 s} \frac{v[k]_j}{n^2 s}$ and if there exists an integer v such that $|\sigma[k] - p_v| < \frac{1}{2e^2 n}$, where $p_v = \frac{1}{2}\left(\frac{n-2}{n}\right)^v \left(\left(\frac{n}{n-2}\right)^v - 1\right)$, then $y[k] = v$.
If for any m, no such value v exists, or if $\sum_{k=1}^{m} y[k] \neq n$, the protocol fails.

The complexity of **Protocol 1** is as follows: n voters each create $mn^2 s$ distributed bits, for a total of n messages of size $mn^2 s$. **Phase B** requires a single simultaneous broadcast among n participants, each sending a message of size $mn^2 s$.

Lemma 1. *(Correctness) If **Protocol 1** does not fail, the result of the vote is consistent with the vote of the honest voters and some non-adaptive choice for the dishonest voters, except with probability exponentially small in s.*

Proof. Our protocol is presented in a way that minimizes the number of messages sent by each voter; it is perhaps best understood intuitively in its sequential version. From this point of view, the following is repeated $n^2 s$ times. For each candidate, voters create a distributed bit. The value of the distributed bit is 1 with probability $1/n$ if this is the candidate the voter chooses and always 0

otherwise. All voters compute the XOR of all their shares and the result will eventually be simultaneously broadcast. The probability that the parity of the broadcast value is 1 is directly proportional to the number of voters voting for the candidate. By repeating this process with each candidate $n^2 s$ times, we can gather enough statistics to compute the vote exactly with very high probability.

The only place a voter can deviate from the protocol is by creating distributed bits with an inappropriate ratio of 0 and 1 values. We first note that if the corrupted voters actually transmit the correct number of private bits in **phase A** and broadcast the correct number of bits in **phase B**, then whatever they actually send is consistent with some global ratio of even and odd distributed bits.

The ratio of even and odd distributed bits, when XORed, will give rise to some probability of an even or an odd bit in the simultaneous broadcast. It is possible to randomize the parity but not to derandomize it: the corrupt participants altogether can increase the probability of an odd broadcast but not make it smaller. Because votes for each candidate are added up for a consistency check, either the corrupted voters make a consistent number of votes or otherwise the protocol will fail. The use of a simultaneous broadcast channel ensures that the voter's inputs are independent of each other.

In the rest of the proof, we give a detailed analysis, using a Chernoff-type argument that the result of the vote will be correct with overwhelming probability.

We fix a value k and suppose that v voters have input $x_i = k$. Thus we need to show that in **Protocol 1**, $y[k] = v$, except with probability exponentially small in s.

Let us look at **phase C** of the protocol. Let p_v be the probability that $v[k]_j = 1$. For $v \leq n$, we have $p_0 = 0$, $p_1 = \frac{1}{n}$ and $p_{v+1} = p_v\left(1 - \frac{1}{n}\right) + (1 - p_v)\frac{1}{n}$. Solving this recurrence, we get

$$p_v = \frac{1}{2}\left(\frac{n-2}{n}\right)^v \left(\left(\frac{n}{n-2}\right)^v - 1\right). \tag{1}$$

Thus, the idea of **phase C** is for the participants to approximate p_v by computing $\sigma[k] = \sum_{i=1}^{n^2 s} v[k]_j / n^2 s$. If the approximation is within $\frac{1}{2e^2 n}$ of p_v, then the outcome is $y[k] = v$. We first show that if such a v exists, it is unique.

Clearly, for $v < n$, we have that $p_{v+1} > p_v$. We also have $\lim_{n\to\infty} p_n = \frac{1}{2} - \frac{1}{2e^2}$. Thus the difference between p_{v+1} and p_v is:

$$p_{v+1} - p_v = p_v\left(1 - \frac{1}{n}\right) + (1-p)\frac{1}{n} - p_v \tag{2}$$

$$= \frac{1 - 2p_v}{n} > \frac{1 - 2p_n}{n} > \frac{1}{e^2 n}. \tag{3}$$

Hence if such a v exists, it is unique. We now show that except with probability exponentially small in s, the correct v will be chosen. Let $X = \sum_{j=1}^{n^2 s} v[k]_j$ with $\mu = n^2 s p_v$ the expected value of X. The participants have computed $\sigma[k] = \frac{X}{n^2 s}$.

By the Chernoff bound, for any $0 < \delta \leq 1$,

$$\Pr[X \leq (1-\delta)\mu] < \exp(-\mu\delta^2/2)\,. \tag{4}$$

Let $\delta = \frac{1}{2e^2 np_v}$. We have

$$\Pr[X \leq \mu - \frac{n^2 s}{2e^2 n}] < \exp(-\frac{n^2 s}{8e^4 n^2 p_v}) \tag{5}$$

and so

$$\Pr[\sigma[k]_i - p_v \leq \frac{-1}{2e^2 n}] < \exp(-\frac{s}{8e^4 p_v}) \tag{6}$$

Similarly, still by the Chernoff bound, for any $\delta < 2e - 1$,

$$\Pr[X > (1+\delta)\mu] < \exp(-\mu\delta^2/4) \tag{7}$$

Let $\delta = \frac{1}{2e^2 np_v}$ and we get

$$\Pr[X > \mu + \frac{n^2 s}{2e^2 n}] < \exp(\frac{-n^2 s}{16e^4 n^2 p_v}) \tag{8}$$

and so

$$\Pr[\sigma[k]_i - p_v > \frac{1}{2e^2 n}] < \exp(\frac{-s}{16e^4 p_v})\,. \tag{9}$$

Hence the protocol produces the correct value for $y[k]$, except with probability exponentially small in s. □

Lemma 2. *(Privacy) In **Protocol 1**, no group of corrupted voters can learn more than what they would have learned in the ideal functionality, and this even if the protocol fails.*

Proof. No assumption is made about the number of dishonest voters. The case where all voters are corrupted is trivially private and in the case where only one voter is honest, his vote can be deduced even in the ideal functionality.

When more than one voter is honest, privacy requires that, even if the tally of the honest voters is known, the individual ballots remain private.

In **phase A**, as long as at least one voter is honest, the value of each distributed bit is perfectly hidden. In **phase C**, no information is sent. We thus have to concentrate on **phase B** where the voters broadcast their information regarding each parity. Let H be the set of honest voters. The dishonest voters learn $\bigoplus_{\ell \in H} q_{jk\ell}$ but no information on these individual values is revealed. The dishonest voters can thus only evaluate the probability that this value is 1 but this information could be deduced from the output of the ideal functionality, for instance by fixing the corrupt participants' inputs to 1. □

It is important to note that the above results do not exclude the possibility of corrupted voters causing the protocol to fail while still learning some information as stipulated in Lemma 2. This information could unfortunately be used to adapt the behaviour of the corrupted voters in a future execution of **Protocol 1**.

3 Voting with Authorities

In this section, we introduce a variation of the previous voting protocol. Our motivation is to reduce the message complexity for the voters and reduce the need of private channels by introducing a relatively small number of voting authorities and by only requiring voters to communicate with these authorities. Additionally, the simultaneous broadcast is only required among the authorities. In this section and the following, we say that a voter constructs a distributed bit *among the authorities* if the voter creates a distributed bit as in section 2, except that the shares are distributed only among the authorities. Our protocol is given as **Protocol 2**.

Protocol 2 Voting with authorities

Input: $x_i \in \{1, \ldots, m\}$ and security parameter s
Output: for $k = 1$ to m, $y[k] = |\{x_j \mid x_j = k\}|$

Phase A (cast)
For each candidate $k = 1$ to m,

1. Each voter i sets the value of $n^2 s$ bits p_{ijk} $(j = 1, \ldots, n^2 s)$ in the following way: if $x_i \neq k$, then all bits are 0; otherwise, exactly ns bits (a fraction $1/n$ of the total) are randomly chosen such that $p_{ijk} = 1$ and the rest such that $p_{ijk} = 0$.
2. For each $j = 1, \ldots, n^2 s$, each voter i constructs a distributed bit *among the authorities* with value p_{ijk}. Let the shares of each distributed bit be denoted $\{p_{ijk\ell}\}$ $(\ell = 1, \ldots r)$

Phase B (broadcast)
All authorities ℓ, for every j and k simultaneously broadcast $q_{jk\ell} = \bigoplus_i p_{ijk\ell}$
Phase C (tally)
To compute the tally, $y[k]$, for each value $k = 1, \ldots, m$, each participant sets: $v[k]_j = \bigoplus_{\ell=1}^{n} q_{jk\ell}$, $\sigma[k] = \sum_{j=1}^{n^2 s} \frac{v[k]_j}{n^2 s}$ and if there exists an integer v such that $|\sigma[k] - p_v| < \frac{1}{2e^2 n}$, where $p_v = \frac{1}{2}\left(\frac{n-2}{n}\right)^v \left(\left(\frac{n}{n-2}\right)^v - 1\right)$, then $y[k] = v$.
If for any m, no such value v exists, or if $\sum_{k=1}^{m} y[k] \neq n$, the protocol fails.
Each authority broadcasts the outcome of the tally, if there is any disagreement, the protocol fails.

The complexity of **Protocol 2** is as follows: n voters each create $mn^2 s$ distributed bits, which are distributed among r authorities, for a total of nr messages of size $mn^2 s$. **Phase B** requires a single simultaneous broadcast among r authorities, each sending a message of size $mn^2 s$. **Phase C** requires r broadcasts of size as most $m \log n$.

Lemma 3. *(Correctness) If at least one authority is honest, and if **Protocol 2** does not fail, the result of the vote is consistent with the vote of the honest voters and some non-adaptive choice for the dishonest voters, except with probability exponentially small in s.*

Proof. The proof is obtained by replacing voters by authorities at the appropriate place in proof of Lemma 1. It is important here that the correctness

probability only depends on s and not on the number of voters or authorities. □

Lemma 4. *(Privacy) In **Protocol 2**, if at least one authority is honest, no collusion of dishonest voters and authorities can learn more than what they would have learned in the ideal functionality, and this even if the protocol fails.*

Proof. The proof is very similar to the proof of Lemma 2. In **Protocol 2**, part of the work performed by the voters in **Protocol 1** is done by the authorities. If at least one authority is honest, there is no way dishonest participants (voters or authorities) can learn any information about the value of the distributed bit created by an honest voter. The rest of the argument is the same as in Lemma 2. □

Note that in **Protocol 2**, any participant can make the protocol fail. Voters can do this, for instance, by setting an abnormally high number of distributed bits to 1, and authorities can do this by changing their inputs into the simultaneous broadcast. Furthermore, note that in **Phase B**, although the simultaneous broadcast happens among the authorities, it is not a problem if the voters are passive listeners. At the end of **Phase C**, the authorities broadcast the result of the tally. We required unanimity of these messages in order to declare that the protocol has succeeded.

4 Voting with Authorities and Verification

One of the issues with the previous two protocols is that any voter can cause them to fail by introducing noise. In this section, we use the cut-and-choose technique, augmented with an equality test, to allow authorities to revoke a noisy ballot. This is done by having each voter distribute many encrypted but identical *votes*, where a *vote* is k lists of n^2s bits (as created, for instance, in step 1 of **Phase A** of **Protocol 2**). A vote is *correct* if its contents correspond to the construction of step 1 of **Phase A** of **Protocol 2**, i.e. all bits are even except one candidate which has exactly ns bits sets to 1. The authorities then open half of the votes and verify the correctness; a subsequent step will ensure that the unopened votes are equal, thus providing exponential security.

Our protocol is presented as **Protocol 3**, in which the authorities use the following two simple routines.

Random choices: authorities can generate common random bits in the following way. Each authority locally generates a random bit, after which all authorities simultaneously broadcast these bits. The common random bit is set to be the parity of the broadcast bits. Obviously, this value is truly random if at least one authority is honest. This process can be done in parallel, requiring only one simultaneous broadcast.

Distributed bit equality: suppose the authorities share two distributed bits. They can verify if these two distributed bits have the same value without revealing this value. Let $a = \bigoplus_{i=1}^{r} a_i$ and $b = \bigoplus_{i=1}^{r} b_i$ be the two distributed bits.

Each authority i simultaneously broadcasts $c_i = a_i \oplus b_i$. If $\bigoplus_{i=1}^{r} c_i = 0$ then the distributed bits are equal (unless an authority is cheating). A dishonest authority can make the protocol output the wrong answer, but under no circumstance will this process reveal any information about the values of a or b.

Protocol 3 Voting with authorities and verification

Input: $x_i \in \{1, \ldots, m\}$ and security parameter s
Output: for $k = 1$ to m, $y[k] = |\{x_j \mid x_j = k\}|$ as well as a list of voters with revoked ballots

Phase A (randomness)
The authorities generate enough common random bits.
Phase B (verification and vote casting)
For each voter:

1. Each voter executes step 1 of **Phase A** of **Protocol 2**, thus creating one *vote*.
2. $2s$ copies of the vote are made, and for each vote, the shares of the distributed bits are computed as in step 2 of **Phase A** of **Protocol 2** (the shares are independently randomly chosen).
3. Each vote is encrypted with two random permutations: the first permutation changes the order of the k candidates, and the second permutation changes the order of the $n^2 s$ distributed bits (the same permutation is applied for each candidate within a vote).
4. The shares of the encrypted votes are distributed among the authorities.
5. The authorities randomly choose s votes and simultaneously broadcast all bits involved in these votes.
6. If any of the opened votes is not correct, the voter's ballot is revoked.
7. Each authority reveals to the voter which votes were opened. If the voter receives inconsistent messages, his ballot is revoked.
8. For the s remaining votes, the voter reveals to the authorities both the permutation that was applied on the distributed bits and the permutation that was applied on the candidates. The authorities permute their shares of the remaining votes so that all votes are equal.
9. The authorities perform *distributed bit equality* tests between each distributed bit of the first remaining vote and all corresponding distributed bits for all other remaining votes. If any of these tests fail, then the voter's ballot is revoked. If all tests succeed, all but the first remaining vote are discarded.

Phase C (broadcast and tally)
Phases B and **C** of **Protocol 2** are performed with all remaining non-revoked votes.

Note that in **Protocol 3**, any dishonest authority can make the protocol fail and any authority can dishonestly revoke any voter's ballot.

The complexity of **Protocol 3** is as follows: each of the n voters sends r messages of size $2mn^2s^2$ for the votes (step 4) and r messages of size $n^2s^2 \log(n^2 s) + sm \log(m)$ for the permutations (step 8). In order to generate enough random bits, the authorities are involved in a single simultaneous broadcast of size $n \log(\binom{2s}{s}) \in O(ns)$. For the rest of the protocol, the r authorities are involved in step 5 in a simultaneous broadcasts of size mn^2s^2 for each voter; in step 7, they require a message of size s for each voter, and in step 9, they broadcast

$(s-1)mn^2s$ bits. **Phase C** requires one last simultaneous broadcast of size mn^2s as well as r broadcasts of size as most $m \log n$.

Lemma 5. *(Correctness) If at least one authority is honest, and if **Protocol 3** does not fail, then every ballot that is not revoked is correctly counted except with probability exponentially small in s.*

Proof. The proof is identical to the proof of Lemma 3. The verification of the vote only makes the protocol more robust. □

Lemma 6. *(Privacy) In **Protocol 3**, if at least one authority is honest, no collusion of dishonest voters and authorities can learn more than what they would have learned in the ideal functionality.*

Proof. To see that privacy of the vote is guaranteed if at least one authority is honest, we first observe that **phase B** of the protocol does not reveal information about the voters' choice; it only ensures correctness of the vote. Once this phase is done, the rest of the protocol is identical to **Protocol 2** and the same argument as in Lemma 4 can be used here. □

As mentioned at the beginning of this section, in **Protocols 1** and **2**, a voter can vote in an inconsistent way, causing the protocol to fail with very high probability. In **Protocol 3** the votes are verified: if a vote is not correct, there is only a probability exponentially small in s that the vote will not be revoked. Thus, dishonest voters can only make the protocol fail with exponentially small probability in s. We formalize this below.

Lemma 7. *(Robustness) No coalition of voters can alone make the protocol fail, except with exponentially small probability in s.*

Proof. The only way for a voter not to provide the correct information in **phase B** is to generate incorrect votes. Since half of the votes are opened, and the other half is checked for equality, the only way for a voter to successfully provide an incorrect ballot is for the s opened votes to be correct and the s remaining votes to be incorrect, yet identical. This happens with exponentially small probability in s. □

5 Conclusion

We presented three voting scheme with unconditional security and information-theoretic correctness, without assuming any bound on the number of corrupt voters or voting authorities. For this to succeed, we had to assume pairwise private channels and a simultaneous broadcast channel (as discussed, this assumption can be replaced by temporary computational assumptions, yielding everlasting security). We also had to allow any participant to cause the protocol to fail. Fortunately, we were able to relax some of the above assumptions in **Protocols 1** and **3** by introducing a set of voting authorities.

We are currently considering a tradeoff between the revoking power of authorities and the correctness of the protocol. This can be achieved as a modification of **Protocol 3** by randomly grouping the authorities and by performing the protocol in parallel within each group.

Although our initial motivation was of theoretical nature, we believe that this work might lead to interesting practical applications.

Acknowledgements

The authors wish to thank Sébastien Gambs for proofreading and Jeroen van de Graaf for suggesting that we write up and submit our ideas.

References

[BOGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the 20th annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 1988.

[BT07] A. Broadbent and A. Tapp. Information-theoretic security without an honest majority. In *Proceedings of the 13th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT '07)*, pages 410–426, 2007.

[CCD88] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proceedings of the 20th annual ACM Symposium on Theory of Computing (STOC)*, pages 11–19, 1988.

[CGMA85] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proceedings of the 26th annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 383–395, 1985.

[Cha88] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[Gra08] J. van de Graaf. Private Communication, 2008.

[HM05] A. Hevia and D. Micciancio. Simultaneous broadcast revisited. In *Proceedings of the 24th annual ACM symposium on Principles of distributed computing*, pages 324–333, 2005.

[Ken99] A. Kent. Unconditionally secure bit commitment. *Physical Review Letters*, 83:1447–1450, 1999.

[RBO89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the 21st annual ACM Symposium on Theory of Computing (STOC)*, pages 73–85, 1989.