Anonymous one-time broadcast using non-interactive dining cryptographer nets with applications to voting

Jeroen van de Graaf

[1]Laboratório de Computação Científica – Universidade Federal de Minas Gerais (UFMG)
Av. Antônio Carlos 6627 – 31270-901 – Belo Horizonte (MG) – Brasil

jvdg@lcc.ufmg.br

Abstract. *All voting protocols proposed so far, with the exception of a few, have the property that the privacy of the ballot is only computational. In this paper we outline a new and conceptually simple approach allowing us to construct a protocol in which the privacy of the ballot is unconditional. Our basic idea is to modify the protocol of Fujioka, Okamoto and Ohta[10], which uses blind signatures so that the voter can obtain a valid ballot. However, instead of using a MIX net, we use a new broadcast protocol for anonymously publishing the vote, a non-interactive variation of the Dining Cryptographer Net.*

1. Introduction

1.1. Motivation

Voting protocols are often divided in three categories: based on MIX networks, based on blind signatures, and based on homomorphic encryption. See for instance [11] for a some what simplified but nevertheless interesting overview. A major flaw of most voting protocols proposed so far, is that the privacy of the ballot is only computational:

MIX For protocols based on MIX nets[2] this is so since, besides the usual assumptions about the authorities not revealing the permutations, their security is also based on RSA or ElGamal: if one knows the private key used for mixing, one can trace back a vote through the cascade of mixes and find out who submitted the vote.

blind signatures The blind signature[4] allows a voter to obtain a signed ballot from a voting authority, who will cross out the voter from the list of voters. After unblinding his vote, the voter needs to publish his ballot anonymously. All protocols published so far use a MIX net for this purpose, which reduces it to the previous case.

homomorphic encryption A homomorphic encryption scheme (see for instance [5]) uses a clever way to encrypt each individual vote, such that by manipulating (in most cases: multiplying) the encryptions the votes can be tallied, so that the problem reduces to decrypting collectively some specific value. It is obvious that someone with infinite computational power could decrypt all the ballots and therefore discover who voted for whom.

This flaw is really worrisome for the following reason: with storage becoming cheaper and cheaper every year, we *must* assume that all data that has been made public through an election protocol, will never be erased, i.e. that some copy of it will be stored forever. We also must assume that at some point in the future it will be possible to break the underlying computational assumption, and then it will become public who voted for whom. Though one can argue that this information might have become irrelevant after many decades, this point is more important than it seems. For instance, people might like to know who the President of the United States voted for when he was young. He might have had a flirt with the communist party, who knows. Even today historians will find it interesting to know Churchill's voting behavior in 1900, when he was about 25 years old. A more dramatic example (due to an anonymous referee) is a scenario in which a dictator gets elected democratically after decades of trying. Once in power, he systematically goes after the voters who voted against him in earlier elections, or after their descendents.

Real world voting systems always have had the property that the vote (the information containing the voter's choice) is irretrievably destroyed. Newly proposed protocols should have this property too. **Computational privacy is not enough for voting**, since one day or another the computational assumption will be broken. Of course, in less important elections it might be acceptable that the privacy of the vote is only computational. But it should be pointed out that estimating for how long the privacy of the ballot will be preserved is notoriously difficult, as various examples exist of computational assumptions being broken much earlier than expected.

1.2. This paper

In this paper we describe a conceptually simple voting protocol which has unconditional voter privacy. The main ingredient is that we propose a non-interactive (i.e. one round) version of the well-known Dining Cryptographers protocol[3], which, as far as we know, has not been published before, and is of separate interest. Though the basic idea is simple, some technical subtleties need to be resolved since, unlike the original protocol, there is only one round. In the next section we will give a high level sketch of the proposed voting protocol, whereas Section 3 contains a detailed description of the non-interactive variation of the Dining Cryptographers protocol.

1.3. Relation to other work

We are not aware of any paper that proposes the non-interactive use of the Dining Cryptographers protocols. Bos [1] presents a voting protocol based on DC nets, but this protocol makes the (in our opinion unrealistic) assumption that all voters are simultaneously online.

The protocol of Cramer et al. [6] uses techniques whose mathematics is similar to the math of DC nets. This protocol is exclusively devoted to voting and uses only one slot (as is the case with [1]), while we propose the use of many slots to allow each participant to broadcast his vote. As a consequence, their protocol has a much smaller message size than ours but is also less general. It only deals with Yes/No votes, but if the number of candidates or the way preferences are expressed changes, the protocol has to re-engineered. This is not the case for our protocol; we basically propose an anonymous broadcast channel which is insensitive to the exact lay-out of the message sent through it.

Another interesting paper is [9], which argues for “everlasting privacy” in voting, like we do. It uses a non-interactive bit commitment scheme as the underlying assumption, a primitive that we also need, but in most other aspects the techniques used in their paper are completely different.

2. A high-level sketch of the new voting protocol

The basic idea for this new voting protocol consists of two main ingredients, which are presented separately. Note that the first ingredient is not new.

2.1. The first ingredient: Blind signatures

The idea is that by using a Chaum-like blind signature, the voter gets a valid ballot from the voting authority. In particular there is the Fujioka, Okamoto e Ohta protocol[10] that could be used here. This protocol allows the voter Alice to contact a Ballot Issuing Authority in order to submit a blinded ballot. The Authority responds by (blindly) signing Alice’s ballot and sending it back to her. Because the blinding process is perfect (since all blinding factors are equiprobable, all possible votes are), the authority obtains no information whatsoever about Alice’s vote.

It is important that the Authority marks Alice’s name on the list of eligible voters, thus avoiding that Alice tries to vote twice. Equally important is that both parties sign their messages and maintain records of them, in order to resolve later disputes.

When receiving the message from the Authority, Alice unblinds it and verifies that it contains a valid, signed ballot.

2.2. Mixnets and their disadvantages

The next step is that Alice must cast her ballot through some anonymous broadcast mechanism. One possible way to accomplish this is by using MIX-nets. This allows the voter to submit his ballot in encrypted form, usually with several layers of encryption. All the ballots related to one election could be placed on a web site, for instance. Now Mixing Authorities are involved in decrypting the batch of ballots layer by layer, while shuffling the intermediate results. If the voter followed the protocol correctly, the last decryption will show the vote, signed by the Ballot Issuing Authority, in the clear. Later some audit protocol is run to verify (with high probability) that none of the mixing authorities cheated.

When submitting their vote, there is no reason for voters to withhold their identity. On the contrary, one can imagine that the identity of the voter is placed next to his encrypted vote, since the privacy is supposedly guaranteed by the mix net. However, when (and not: if) at some moment in the future the private (RSA or ElGamal) keys are broken, the permutation used by each mix can be reconstructed and all the links can be established between the encrypted vote as submitted by the user, and the fully decrypted result of the mix. So the privacy of the ballot will be violated.

2.3. The second ingredient: Non-interactive Dining Cryptographers

We therefore introduce a second and new ingredient of our protocol: instead of using a mix-net, the voter uses a non-interactive variation of the Dining Cryptographers protocol[3] to post her vote. This protocol can be informally described as follows:

Three cryptographers are having dinner. When they have finished their meal, the waiter informs them that their meal has been paid already. The cryptographers decide they want to find out whether the meal was paid by an outsider (the NSA), or by one of the three present. However, if the payer is one of them, the identity of this payer should remain secret, i.e. the payer should be anonymous. In order to accomplish this, they decide to run the following protocol:

1. *Each one flips a coin, and shares the result with his neighbour on the right.*
2. *Each one looks at his own coin and the one of his left neighbor. The two coins can have the same face up (heads-heads, tails-tails) or different faces up (heads-tails, tails-heads). Each person announces publicly "SAME" or "DIFFERENT", but with the additional rule that the person who paid reverses his statement (he "lies").*
3. *When an odd number of persons announces "DIFFERENT" they know one of them has paid; when an even number of persons announces "DIFFERENT" they know an outsider paid.*

It is not difficult to see how this protocol extends to many bits and many parties. See [3] and [1], Chapter 2. However, the model used in these papers is completely based on a network setting: people connected to each other through a LAN (Ethernet), broadcasting messages to each other. An implication of this model is that if problems occur, new transmission rounds are available to resolve those problems.

Here we propose a non-interactive variation of the DC net, an idea which, as far as we know, has not been explored in the literature. The idea is that each voter submits **once** a bit sequence, properly identified, which gets published on the Bulletin Board. This sequence might be long but should still be reasonable for transmission, in the order of ten of megabytes, say. The non-interactive variation works very much like the original DC proposal:

1. in a preliminary phase, each pair of parties exchanges random bits.
2. based on these random bits and on the party's input, it broadcasts a message.
3. all message are combined in such a way that all the random bits cancel, and only the inputs of all parties remain.

It is well known that DC-nets provide unconditional anonymity; this follows from the fact that the probability distribution on the random bits is uniform. See [3] for details.

In the non-interactive case, to avoid collisions, most bits of the sequence published will contain the XORs of the bits exchanged with other parties, but no message bits. Only at a very short interval a message, containg the vote with its signature, will be added to the XORs as well. Very short signatures that allow blinding can be obtained using elliptic curves, resulting in signature lengths of 160-200 bits.

Note that the signature scheme only need to be resistant to attacks from the voters for the duration of the election to avoid any insertion of false votes. If the signature scheme gets broken after the election, the result is not compromised; indeed, revealing the private key after the election has completed (more precisely: when no more ballots can be issued by the Ballot Issuing Authority) would not affect security.

After each participant has submitted (published) his sequence, the bitwise XOR for each bit position over all the sequences is computed, yielding anonymously the net messages published by the participants, i.e. the votes. A tallying authority calculates the final result, which can be verified by any scrutineer who cares to.

2.4. Some subtle issues

Ideally this would be the end of it, but there are three problems to be resolved:

Collisions A collision happens when more than one message is published in the same slot. The collision probability must be kept very low since it implies that at least two (random) votes get lost. However, there exist various straightforward techniques to keep the collision probability low. They basically consist of increasing the length of the bit sequence, or of sending the same message various times, through a different channel or through the same one. See Section 4.1 for discussion.

Disrupters A disrupter is a party that deviates from the protocol by broadcasting a garbage string (arbitrary random bits), instead of a string based on shared random bits and the party's input. his is truly an annoying problem because in the traditional (interactive) DC setting catching disrupters is already problematic: all participants have to engage in an on-line protocol to drive out the disrupter(s). In our case we cannot catch disrupters afterwards, so we must catch them when they submit. This can be done by having the parties commit on their random bits exchanged. To preserve privacy we will need a Bit Commitment Scheme that is computationally binding and unconditionally hiding. Then we will use a cut-and-choose protocol in which the sender shows that he is following the protocol. In particular he needs to show that for the whole sequence, except the part that contains the message (vote), the bits are truly the result of XORs of bits already committed to. Whether he follows the protocol for the bits dedicated to the message we could check but don't have to; he could sent in garbage, but at least he won't disrupt the channel.

No-showers A no-shower is a party that went through the initial phase of the protocol, has shared its random bits with other parties, but did not submit any sequence. His identity will become known, and the best solution is to have people who have interacted with him recalculate their submissions. Alternatively, in a setting of a small set of authorities who exchange random bits with each voter, the authorities can simply disregard the random bits of the no-shower(s).

Note that in a situation where there is a relatively large amount of trust between the participants, disrupters and no-showers are of no concern.

3. A detailed description of the Non-interactive Anonymous Broadcast protocol

In this section we describe a Non-Interactive version of the Dining Cryptographers protocol. Though the motivation for this protocol is voting, we describe the protocol in a general setting, i.e. we do not use voting-specific terminology. But we do assume the same message size for all participants.

Some parts of the protocols are of a challenge-response nature, in which the responses are always either random bits or random permutations (which, of course, can be obtained from random bits). In the protocol description we will write that a party commits, and then receives a challenge from a trusted random source. However, it will be understood that this is implemented using the Feige-Shamir heuristic: after having fixed the commitments, the party applies a hash function to them and uses the results for the challenge. This technique generally believed to be secure, and reduces a three-step protocol to one with only one step.

3.1. Notation

We suppose there are P participants $\mathcal{P}_i$, with $i \in \{1 \dots P\}$. The purpose of each participant $\mathcal{P}_i$ is to publish a message v_i anonymously.

To this end a DC channel is available of total size N bits, which is divided into S slots, each of size L. We define the net input of participant $\mathcal{P}_i$ to the DC channel as $M_i = M_i[1] \dots M_i[S]$; here each $M_i[s]$ denotes a slot.

Now $\mathcal{P}_i$ may occupy at most 1 out of those S slots to publish v_i, so there is one $s \in \{1 \dots S\}$. The other $S-1$ slots must remain empty, i.e. for $s' \neq s$ we have that $M_i[s'] = 0^L$, a string consisting of L zeroes.

As in the original DC protocol, any pair of participants $\mathcal{P}_i, \mathcal{P}_j$ can choose to engage in an exchange of random bits. If this is the case they share an edge in the so-called privacy graph and we call them neighbors. The privacy properties of this protocol are identical to those described in the original paper[3].

We introduce the following notation:

- $M[s]$ is the sth slot of M
- $M[[u]]$ is the uth bit of M
- We denote the random string of size $N = SL$ shared between $\mathcal{P}_i$ and $\mathcal{P}_j$ with R_{ij}. If no sharing takes place, we define $R_{ij} = 0^N$.
- The overall random string used by $\mathcal{P}_i$ for encryption is $R_i = \bigoplus R_{ij}$, where j ranges over the neighbors of $\mathcal{P}_i$.
- $\mathcal{P}_i$'s overall contribution to the channel is called C_i, and we have therefore that $C_i = M_i \oplus R_i$.
- We denote bit commitments with a bar: $\overline{x}$ is a commitment to the bit x.
- In the next section we will use bit commitments with a special property, which can be implemented as a vector of pairs of ordinary bit commitments. They will be denote by $\overrightarrow{\overline{x}}$.

3.2. Bit commitments with XOR

An essential property of our protocol is that each participant must commit to his contribution, and show that it has the proper format, though without showing the value.

Since we want unconditional privacy, we obviously need a bit commitment scheme that is unconditionally hiding and computationally binding. There are various options here, but for concreteness we use a bit (string) commitment scheme based on hash functions, as presented in [8].

Actually, ordinary bit commitments are not good enough for our protocol, since we will need a way to prove that linear relations between bit commitments hold without opening the values. For instance, we would like to show that $\overline{x_1} \oplus \overline{x_2} \oplus \ldots \overline{x_k} = 0$. That is, we want to show that the equality $x_1 \oplus x_2 \oplus \ldots x_k = 0$ holds without revealing any other information about the x_i. We will show a general construction to accomplish this property for *any* kind of bit commitment, at the expense of a factor $2K$, where K is a security parameter.

The solution presented here, attributed to Bennett and Rudich, is described Section 2.2 of [7], where it is called ”Bit Commitment with XOR”, abbreviated BCX. The idea is to represent each BCX bit commitment as a vector of pairs of simple bit commitments, such that each pair XORs to the committed bit value. This allows for challenges on one half of the bit commitment, without revealing its value. We describe the scheme here informally, using a simple example. A more formal description is given in [7], which also shows that the scheme generalizes easily to proving linear relations between many bit commitments.

Using this approach, a bit commitment to $x = 1$ has the following representation: $\overrightarrow{x} = \langle(\overline{0},\overline{1}),(\overline{1},\overline{0}),(\overline{0},\overline{1}),(\overline{0},\overline{1}),(\overline{1},\overline{0})\rangle$, where $K = 5$, an artificially low value. Suppose $\mathcal{A}$ is committed to $x = 1$ and also to $y = 0$ as follows: $\overrightarrow{y} = \langle(\overline{1},\overline{1}),(\overline{1},\overline{1}),(\overline{0},\overline{0}),(\overline{1},\overline{1}),(\overline{0},\overline{0})\rangle$, and that she needs to prove that the two bit commitments are different, i.e. that $x \oplus y = 1$.

1. The first step of the protocol is that $\mathcal{A}$ tells for each pair of $\overrightarrow{x}$ and $\overrightarrow{y}$ whether the left component is equal or different (the two components (columns) are labeled 0 and 1, also named *left* and *right*; the rows are labeled from 1 to K). Or, equivalently, she opens the values $z_k = x_{k0} \oplus y_{k0}$ for $k = 1, \ldots, K$.
2. In the second step she receives K challenge bits $b_1, \ldots, b_K$ from the trusted random source.
3. Thirdly, for each $b_k = 0$, she is required to open the left component, x_{k0} and y_{k0}, of the kth pair of $\overrightarrow{x}$ and $\overrightarrow{y}$, and $\mathcal{B}$ must check that they are equal. If $b_1 = 1$ then $\mathcal{A}$ opens x_{k1} and y_{k1} and $\mathcal{B}$ must check that they are different (their mod 2 sum adds to 1). This must be executed for each challenge bit $b_1, \ldots, b_K$.

It is easy to see that $\mathcal{A}$ does not reveal the actual values of the bit commitments through this protocol since only either the left or the right component of each pair is revealed, while the value is defined as the XOR of both. As far as the binding property is concerned: obviously, for each row in which she tries to cheat, $\mathcal{A}$ gets caught with a probability $1/2$.

It is important to observe that in the protocol of (in)equality between two BCXs, the unopened halves are not lost (useless), implying that the BCX can and must be preserved.

For instance, after a proof that $\overrightarrow{x} = \overrightarrow{y}$, the remaining halves constitute a new BCX $\overrightarrow{t}$ with the property that $t = x = y$. Note that the view of the protocol showing inequality should be stored and will be needed when opening, since if $z_k = 1$, this has the effect of flipping the semantics of the corresponding bit commitment for the kth pair: $t = x_{kb'_k} \oplus y_{kb'_k} \oplus z_k$, where $b'_k = 1 \oplus b_k$, the bitwise complement of b_k.

3.3. Preliminary phase

As in the original DC protocol, during the preliminary phase each pair of neighbors creates a random bit string R_{ij} of size N. But unlike the original protocol, we require that both $\mathcal{P}_i$ and $\mathcal{P}_j$ commit individually to each bit of R_{ij} using the BCX bit commitment scheme explained above. To avoid any type of collusion between them, it is essential that they show to the world (the other participants, and any other observer) that they are committed to the same value.

We do this as follows: we first consider $\mathcal{P}_i$ and $\mathcal{P}_j$ as one party, written $\mathcal{P}_{ij}$, who will jointly create a set of N BCXs, but of size $2K$ instead of K. (If they cannot agree on how to do this jointly, we assume that at least one party aborts and that this particular pair of parties will not contribute.) They will prove the well-formedness to the other participants by showing that all pairs of the same BCX $\overrightarrow{x}$ encode the same values, i.e. that $x_{k0} \oplus x_{k1} = x$ for $k \in \{1, \ldots, 2K\}$. This is done as follows:

1. P_{ij} creates an additional BCX $\overrightarrow{y}$ of the same value, i.e. $x = y$.
2. The trusted source of randomness supplies $2K$ challenge bits b_i, as well as a random permutation σ on $\{1, \ldots, 2K\}$.
3. P_{ij} proves equality between $\overrightarrow{x}$ and $\overrightarrow{y}$, applying the permutation σ to shuffle the pairs, i.e. by showing that either $x_{k0} = y_{\sigma(k)0}$ or $x_{k1} = y_{\sigma(k)1}$, depending on the value b_k.

If P_{ij} tries to cheat on a subset $\mathbf{A}$, this remains undetected only if the permutation σ maps $\mathbf{A}$ onto itself. If $a = \#\mathbf{A} > 1$ this happens with probability $\binom{2K}{a}/(2K)!$. By repeating the protocol this probability can be reduced to any desired level of security.

After this protocol has completed, $\mathcal{P}_i$ and $\mathcal{P}_j$ split their double BCX of size $2K$ in two BCXs of size K by dividing the pairs evenly between them, for instance $\mathcal{P}_i$ stays with the first K pairs $1, \ldots, K$ and $\mathcal{P}_j$ stays with the second K pairs $K + 1, \ldots, 2K$.

3.4. Publication phase

During the second phase of the protocol each participant decides which message v_i he wants to publish, for instance a signed vote. This part consists of the following substeps:

1. $\mathcal{P}_i$ commits to his input M_i, which contains v_i, and proves that it has the proper format;
2. $\mathcal{P}_i$ commits to the contribution C_i and proves that it has the proper format.

3.4.1. Commitment and proof of M_i

1. Let v_i be the message that $\mathcal{P}_i$ wants to publish. $\mathcal{P}_i$ now creates M_i by selecting a slot $s \in \{1, \ldots, S\}$ randomly. He sets $M_i[s] := v_i$, whereas for $s' \neq s$ he sets $M_i[s'] := 0^L$, a slot with only zeroes.
2. $\mathcal{P}_i$ commits to $M_i[[1..N]]$, the individual bits of M_i.
3. Through a proof, $\mathcal{P}_i$ must show that M_i has the proper format, i.e. that at least $S - 1$ slots are zero. To this end we use a straightforward subprotocol:
   - i $\mathcal{P}_i$ chooses a random permutation σ of size S, and uses it to permute the slots in M_i, thus creating M_i'. In other words, $M_i'[s] := M_i[\sigma(s)]$. Then he commits to the individual bits of M_i'.
   - ii A random challenge bit c is generated by the trusted source.
   - iii If $c = 0$ then $\mathcal{P}_i$ reveals the permutation σ and proves equality between $\overrightarrow{M_i'}$ and $\overrightarrow{M_i}$ under the permutation σ. If $c = 1$ then $\mathcal{P}_i$ opens the bit commitments of M_i' for those slots that contain zeroes only.

This protocol must be executed K times in parallel, where K is a security parameter. Cheating succeeds only if $\mathcal{P}_i$ can predict the challenge bits in each round, which happens with probability 2^{-K}.

3.4.2. Commitment and proof of C_i

$\mathcal{P}_i$ now adds the random bits R_i exchanged between his neighbors to the input M_i in order to compute his contribution C_i as follows: $C_i[[n]] = M_i[[n]] \oplus R_i[[n]] = M_i[[n]] \oplus R_{ij_1}[[n]] \oplus \cdots \oplus R_{ij_u}[[n]]$, where $j_1, \ldots, j_u$ are the indexes of $\mathcal{P}_i$'s neighbors, and where n ranges from 1 to N. Then $\mathcal{P}_i$ publishes C_i and signs.

Observe that during the preliminary phase $\mathcal{P}_i$ committed himself to R_{ij}, and in the first step of the publication phase he committed to M_i, in both cases using the special BCX commitment scheme presented in section 3.2. So using the protocol presented in that section, $\mathcal{P}_i$ can show (in the "committed world") that the assignment of C_i is correct, i.e. that indeed $C_i[[n]] = \overrightarrow{M_i[[n]]} \oplus \overrightarrow{R_{ij_1}[[n]]} \oplus \cdots \oplus \overrightarrow{R_{ij_u}[[n]]}$ for each n.

4. Technical considerations

4.1. Calculating the Collision Probability

Considered separately from the context of voting, the NI DC channel deserves a performance analysis. Since participants choose slots randomly, there always exists a chance that a collision occurs, i.e. two participants occupy the same slot, and consequently the corresponding slot contents (the v's) are lost. If $S = 365$ and $P = 23$, we are back to the birthday paradox: with probability approximately 1/2 we have a collision, so the message of two participants is lost. To reduce this probability we can increase S. A well-known formula that approximates the collision probability for this case is $1 - e^{-P(P-1)/2S}$.

Another solution is to run Q DC nets in parallel. The probability that in all of them a collision occurs is $(1/2)^Q$, and that the *same* participant is involved in all of them equals

$(\frac{2}{P})^Q$ (where we assume that the collisions in the DC nets are independent, and where we ignore collisions which involve more than two participants (which have a very low probability)).

But we can do even better. Instead of using $Q = 10$ (say) parallel nets, it is certainly more effective to use the same total number of slots, i.e. $S' = 3650$ but let the participant choose 10 slots randomly, instead of only one. Since the first version (Q parallel nets) is a special case of the second ($S' = QS$), the collision probability of the second is bounded by the first. Preliminary computers simulations suggest it is orders of magnitudes lower.

Approximating this probability accurately is not a simple exercise and a more careful analysis is appropriate. For instance, it would be interesting to see how the parameters interrelate and be able to answer questions such as: Given a total of S slots and P participants, how many message T should each participant send in order to maximize successful completion of the protocol? Or reversely, given P participants, how should we choose S and T if we want the failure probability to be really low, say 10^{-20}? These questions are still subject of ongoing research.

4.2. Optimizing the BCX

The current version of the protocol is rather crude, the main point of this paper showing the possibility of unconditional privacy in voting in a conceptually simple way. Very rough estimates indicate that the current version of this protocol will result in files in the order of giga- or terabytes, for $P = 500$ (the average size of a precinct in Brazil). However, it seems probable that by fine-tuning of the protocol and a careful analysis of the probabilities, substantial gains can be obtained.

For instance, a major cause is the expansion caused by the BCX, as every bit of the channel needs at least one BCX, which is very inefficient representation. In fact, the bit commitment scheme used by Bos in his voting protocol ([1], Chapter 3) have exactly the desired properties resulting in substantial gains. Another possibility for savings is that the current protocol is in some sense too robust, and that by trading off the probability of catching someone cheating on an individual vote some efficiency can be gained.

4.3. Ballot marking

A major flaw of the current protocol is that there are various ways in which a voter can mark his ballot, which leaves the protocol very vulnerable to vote buying and selling.

After reading an earlier version, Madhu Sudan observed that a voter can mark his ballot by choosing a particular (set of) slot(s). A newer version tried to address this issue by taking away this freedom to choose the slot(s). However, Tal Moran observed that this does not solve the problem, since the voter is the only one who knows beforehand which slot(s) he is entitle to. In addition, an anonymous referee observed that (a hash of) the signature on the ballot from the Fujioka-Okamoto-Ohta scheme can also be used for such purposes.

These are serious concerns, and at the moment their does not seem a simple way to address them.

5. Conclusions

This paper shows a conceptually simple protocol for voting with unconditional privacy. The paper does so using a non-interactive version of the Dining-Cryptographers protocol, which is not as efficient (in terms of message size) as other voting protocols that offer unconditional privacy, but is of interest in itself since it may have other applications. A serious drawback of the protocol is that it permits ballot marking, that is, the voter can show which vote in the final output was her's.

The resulting protocol is certainly feasible for voting in small groups (up to 50 participants, say) where the chance of someone disrupting or not participating is low. Otherwise it might be wiser to define a small number of authorities, whose main role is to reduce the interactions necessary to eliminate no-showers. As in the mix networks, these authorities protect the privacy of the voters, but unlike the mix case, there is no additional computational assumption.

The author strongly believes that unconditional privacy for voting is a desirable property. The fact that at some unknown point in the future voter privacy is completely violated is not acceptable, and the public may actually reject electronic voting systems once this point becomes clear. Therefore, the search for practical voting protocols with this property is an important challenge. However, it seems that to get unconditional privacy each voter must exchange a sequence of random bits (dispose of a private channel) with other voters or with authorities. For large elections this might be a very difficult to accomplish.

Acknowledgements

I would like to thank Berry Schoenmakers, who gave important feedback on an earlier (and erroneous) version of this paper. Ron Rivest and Madhu Sudan generously shared information on the issue of approximating the collision probability in the protocol. I also would like to thank those anonymous referees whose contributions have helped to improve the quality of the paper.

References

[1] Bos, J.N.E. *Practical Privacy*, PhD Thesis, http://citeseer.ist.psu.edu/bos92practical.html.

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

[3] Chaum, D. *The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability.* Journal of Cryptology, 1, 1988, pages 65–75.

[4] Chaum, D. *Untraceable electronic mail, return addresses, and digital pseudonyms.* Communications of the ACM, 24(2):84-88, 1981.

[5] Cramer, R., Gennaro, R., Schoenmakers, B., *A Secure and Optimally Efficient Multi-Authority Election Scheme.* CRYPTO 97 http://citeseer.ist.psu.edu/cramer97secure.html.

[6] Cramer, R., Franklin, M., Schoenmakers, B., Yung, M.*Multi-Authority Secret Ballot Elections with Linear Work.* EUROCRYPTO 96 http://citeseer.ist.psu.edu/cramer96multiauthority.html.

[7] Crépeau, C., van de Graaf, J. and Tapp, A. *Committed Oblivious Transfer and Private Multy-Party Computation* CRYPTO'95, Springer, LNCS, vol. 963, 1995, pp.110-123.

[8] Jakobsson, M., Juels, A. and Rivest, R. *Making Mix Nets Robust For Electronic Voting By Randomized Partial Checking* (published where?).

[9] Moran, T. and Naor, M. *Receipt-Free Universally-Verifiable Voting With Everlasting Privacy*, CRYPTO 2006.

[10] Fujioka, Okamoto, Ohta *A Practical Secret Voting Scheme For Large Scale Elections*. (AUSCRYPT '92). See http://theory.lcs.mit.edu/∼rivest/voting/papers/-FujiokaOkamotoOhtaAPracticalSecretVotingScheme-ForLargeScaleElections.pdf

[11] Traoré, J., Arditti, D. e Girault, M. *Voting protocols — state of the art and e-poll project.* http://www.francetelecom.com/sirius/rd/fr/-memento/mento20/chap2.html.php.