Voting with unconditional privacy: CFSY for booth voting

Jeroen van de Graaf

Departamento da Computação
Universidade Federal de Ouro Preto
jvdg@iceb.ufop.br

Abstract. We adapt the Cramer, Franklin, Schoenmaker and Yung internet voting protocol[12] to the booth setting. In this protocol, expressions of the form $g_0^r g_1^{x_1} \cdots g_l^{x_l} \pmod q$ are used to define an unconditionally hiding commitment scheme. Because of their homomorphic properties, they are particularly suited for voting protocols with unconditional privacy. In fact, a survey shows that almost all these protocols use, or could benefit from, these commitments. Though not novel cryptographically speaking, the protocol presented is interesting from a voting perspective, because it is simple enough to be understood by non-cryptographers, yet has many desirable properties, such as unconditional privacy, correctness under the discrete log assumption, individual and universal verifiability, and (optionally) ballot casting assurance. In addition, we discuss interesting relations to and/or simplifications, of several other protocols, such as the booth voting protocol of Moran and Naor[15], Split-Ballot[16], MarkPledge[2] and Scratch & Vote[3].

1 Introduction

1.1 Computacional versus unconditional privacy in voting

Voting protocols seem to come in two settings: either the protocol achieves computational privacy of the ballot and unconditional correctness of the vote count, or the reverse: unconditional privacy and computational correctness. Just as with bit commitment, achieving both unconditional privacy and unconditional correctness seems to be impossible, but this question is not fully settled yet (see [7]).

The overwhelming majority of voting protocols is based on homomorphic encryption and/or on encryption mixes, so all these protocolos have in common that they provide only computational privacy, This is somewhat surprising. Already a decade ago it has been argued in the context of credential mechanism [8] that privacy should be unconditional, since individuals cannot be expected to assess the strength of cryptographic mechanisms.

In addition, since storage is becoming cheaper every day, we must assume that the data on the bulletin board will be stored forever. This means that the moment the cryptographic assumption on which the privacy of the ballots was based is broken, it will be possible to derive who voted for whom. In other words, with computational privacy we can almost be sure that 30 or 300 years from now we can know who voter for who. This could raise the possibility for some nasty scenarios, for instance a dictator who has come to power goes after people who have voted against him (or his father) several decades ago.

In other words, these proposed voting protocols suddenly have a new, potentially dangerous, property that classical protocols never had. Though voting protocols based on homomorphic encryption and/or on encryption mixes have many interesting properties, including some advantages, we believe that the quest for finding suitable protocols with unconditional privacy is justified.

1.2 Unconditionally hiding discrete log commitments

Expressions of the form $u(r; x_1) := g_0^r g_1^{x_1}$ can be used as a bit commitment scheme which provides unconditional privacy combined with computational bindingness. Though sometimes called Pedersen commitments, expressions of this form were first presented in [10] (see page 98). These commitments are computationally binding, provided that the party who commits cannot break the discrete log in the group G. They play an important role in many of the protocols presented in this paper.

Note that if the order of G is q, then these expressionas are not just *bit* commitments, but values up to q can be committed too. (If the order q were unknown to the committer, as is the case with RSA moduli, there is no limit to the values that can be committed to, but this possibility is not explored here. See [6].)

Unconditionally hiding discrete log commitments, or *UHDLC*s, as we will call them, can be generalized to contain an *arbitrary* number of *integer* valued commitments *without* increasing its size. This can be done by extending the number of generators, resulting in $h = g_0^r g_1^{x_1} \cdots g_l^{x_l} \pmod q$. It is trivial to see that UHDLCs are homomorphic: $(g_0^{r(1)} g_1^{x_1(1)} \cdots g_l^{x_l(1)}) \cdot (g_0^{r(2)} g_1^{x_1(2)} \cdots g_l^{x_l(2)}) = g_0^{r(1)+r(2)} g_1^{x_1(1)+x_1(2)} \cdots g_l^{x_l(1)+x_l(2)}$. This property is what makes them important for voting. They have also many other useful properties.

1.3 Contributions of this paper

Given these UHDLCs, we obtain a straightforward protocol for booth voting. Note that in this setting we may suppose that the authorities control the hardware and software of the Voting Machine. This leads to a considerable simplification compared to the protocol proposed by Cramer, Franklin, Schoenmaker and Yung[12], because in an internet setting one must verify for each incoming vote whether it has the proper format. A significant part of the CFSY paper is dedicated to solving this problem.

We can state the contributions as follows:

- The protocol is uncondionally private. As in CFSY, this is a consequence of commitment scheme used.
- The correctness of the vote count is guaranteed, unless the authorities can break the discrete log problem in G before the election ends. As in CFSY, this is a consequence of commitment scheme used.
- The protocol offers Individual Voter Verifiability, meaning that each voter can verify that his vote is included in the tally.
- Universal Verifiability: Any observer can verify that the tally was calculated correctly.
- Ballot Casting Assurance: Our protocols can be easily extended with the techniques of MarkPledge [17][1]
- The protocol is easy to explain to non-experts, and may suffice for very small elections.
- Some existing election system, such as the Brazilian one, could be enhanced using this protocol, *without a need to modify the existing system.*
- The protocol shows interesting relations to and/or simplifications, of several other protocols, such as the booth voting protocol of Moran and Naor [15], SplitBallot [16], MarkPledge [2] and Scratch & Vote[3].

1.4 Comparison to other work

To the best of our knowledge, the first voting protocol that provides unconditional privacy was presented by Bos [5], [4]. His voting protocol only allows Yes/No (encoded as 1 and 0, respectively) and votes are encoded as simple UHDLCs, i.e. $l = 1$. The votes and decommitment values are added using Dining Cryptographer nets modulo suitably chosen moduli, see §5.4. The DC nets used assume that all voters are online simultaneously, which for a large-scale election is not realistic.

Bos' work has often gone unnoticed, and a few years later Cramer, Franklin, Schoenmakers and Yung (CFSY) presented another protocol with unconditional privacy[12]. Their basic version also uses $l = 1$ but it encodes a masked vote of two options as $\{-1, 1\}$. It uses Pedersen secret sharing[18] to split the masked vote among the authorities, who add the votes and decrypt the result in a distributed fashion. CFSY briefly mention Multiway Elections: an extension of their protocol for elections with more than two options, but provide little detail. Also, CSFY consider internet voting, which is subtly different from booth voting in several security aspects; see 2.2.

For almost a decade, no progress was made on voting protocols with unconditional privacy. Then suddenly three different approaches appeared almost simultaneously and independently.

In [20], a non-interactive version of the Dining Cryptographers protocol is presented, in an attempt to resolve the main disadvantage of the Bos protocol. Unfortunately, to catch a disrupter large amounts of bit commitments with linear properties are needed. (Using UHDLCs these can be optimized enormously; in fact this was the original motivation of the search for efficient, unconditionally hiding commitment schemes with homomorphic properties.) But though the NIDC protocol presented there appears sound, its application to voting is still fraught with seemingly unsurmountable problems (see the final section of that paper for discussion).

At about the same time, Chaum invented PunchScan (www.punchscan.com; for a technical description see [19, 14]) An important theoretical contribution of PunchScan is that it showed how to build an election protocol from bit commitment only, though the original papers used conventional symmetric encryption as a commitment scheme, yielding computational privacy only. Its successor, Scantegrity[11] is also based on bit commitment.

In [21], the author proposes a merge between Prêt-à-Voter and PunchScan, using the former's ballot layout, and the latter's bit commitment scheme plus auditing process. If used in combination with an uncondionally hiding bit commitment scheme, an extremely simple voting protocol with unconditional privacy is obtained.

Moran and Naor have published two different protocols on voting with unconditional privacy. The protocol presented in [15] uses a voting machine, and is based on a generic commitment scheme. An optimized version uses UHDLCs, but for a slightly different purpose. The protocol we discuss in this paper can be thought of as a simplification of theirs, but it should be stressed that the authors had several other goals, whereas here we strive for simplicity. See Section 6.3 for more discussion.

The protocol presented in the second paper[16] is strongly based on PunchScan, but with a very interesting extra twist. The voter must vote by splitting his choice over two ballot halves, which are sent to two different authorities. These two authorities can compute the tally, but they cannot reconstruct an individual vote without conspiring. So there is no single point of failure with respect to privacy. See Section 5.5.

1.5 Paper outline

After a short section with some preliminary notions, we describe the protocol in Section 3. In the next Section, 4, we list the assumptions and then the properties of the protocol. In Section 5 we discuss several mechanism communication mechanisms between the Voting Machine and the Tallying Authority, exhibiting relationships to existing voting protocols that are uncondionally private. In Section 6 we list several other interesting observation; such how to add Ballot Casting Assurance, and how to modify protocols such as Scratch&Vote and MarkPledge.

2 Preliminaries

2.1 Voting terminology

We distinguish the following entities:

- Voters: cast a vote.
- VM=Voting Machine (VM): equipment responsible for recording the voter's vote, and pass it to the TA.
- Tallying Authority (TA): responsible for ensuring that votes are counted and anonymity maintained beyond the Voting Machine.
- Bulletin Board: mechanism used by the TA to publish data related to the election. An web site to which information is only appended (never erased) would be sufficient.
- Auditor: certifies that the election results are correct and have been determined by following the correct procedures.
- Scrutineers: interested third parties that verify whether the election results are correct. The main difference between them and the Auditor is that the latter chooses the random challenges to the TA.

2.2 Booth voting versus internet voting

Booth voting is subtly different from internet voting. The difference is that in booth voting the election authorities have complete control over the hardware and software creates the vote, while the voter has no a-priori reason to trust it. In internet voting this is the opposite. There we assume that the voter controls the hardware and software. This means that special checks are needed in which the authorities verify the format of each incoming vote, This is not an issue in booth voting: we will simply assume that the election authorities control the hardware and software, and it therefore trust blindly that the format of the incoming votes is correct. If a machine were defective, it would simply be replaced.

3 Description of the protocol

In this section we discuss in detail the Multiway Election variation of the protocol presented by Cramer, Franklin, Schoenmaker and Yung. For simplicity of exposition we first consider only one Tallying Authority who has total control over the Voting Machine; this will be discussed further in Section 5. Recall that we consider booth voting, not internet voting.

3.1 Encoding of the vote

We represent the vote for candidate v as a vector $(x_1, ..., x_l)$ which is 0 everywhere, except in the vth position, where it equals 1. We apply this vector representation to UHDLCs, obtaining the following encoding of a vote (cast ballot): $h = u(r; x_1, \cdots, x_l) = g_0^r g_1^{x_1} \cdots g_l^{x_l}$. Because of the homomorphic property of UHDLCs, the multiplication of the h's corresponds to addition of the votes.

3.2 Protocol 1

This suggests the following protocol. Note that parentheses are used to refer to values pertaining to a particular voter.

Phase 1: System initialization

1. In some public cerimony the system's parameters $G, g_0, \cdots g_l$ are computed. ◀

Phase 2: The voter's perspective

1. Voter i presses the candidate of his choice. After a confirmation, the Voting Machine computes $h(i) = u(r(i); x_1(i), \cdots, x_l(i))$, which it prints on a receipt, which is given to the voter. We will discuss its exact format below.
2. The voter steps out of the booth and lends his receipt to a poll worker, who scans it. The receipt's image is then processed using Optical Character Recognition or similar techniques, the result is printed and shown to the voter. The voter confirms the OCR interpretation of the system, and at this point the voter name and/or id number is associated to the receipt image. This confirmation corresponds to the casting of the vote. The voter receives an undeniable proof of this transaction. The poll worker then returns the receipt to the voter, who leaves the precinct.
3. (Optional) The voter shows his receipt to other voters or to helper organizations, who also scan and OCR it.
4. After the election is over, the voter goes to the election web site, types his name or voter id, and verifies that the image of the scanned receipt corresponds with the printed version he received. If they don't match, the voter has a proof that the TA is cheating. ◀

Phase 3: Tallying and publishing the votes Let there be V voters, and let i below be an index ranging over $\{1, \ldots, V\}$.

1. For each i the voting machine communicates the $l+2$ values $\langle h(i),\ r(i)$ and $x_1(i), \ldots .x_l(i) \rangle$ privately to the tallying authority. ***Observation:*** *there exist several possible implementations of the private communication channel between the voting machine and the tallying authority, which are discussed in Section 5. For now we leave this unspecified.*
2. The TA computes the following values:

$$\begin{aligned}
&h^* := \prod h(i) \pmod q \\
&r^* := \sum r(i) \pmod{q-1} \\
&x_1^* := \sum x_1(i) \pmod{q-1} \\
&\ldots \\
&x_l^* := \sum x_l(i) \pmod{q-1}
\end{aligned}$$

These values are published on the bulletin board. ◀

Obviously, now the $x_1^*, \ldots, x_l^*$ contain the final tally, while r^* is the decommitment value of h^*.

Since each vote casting record, containing the name of the voter, the receipt image and the (OCRed) $h(i)$, is published on the bulletin board after the election, any person or entity with sufficient resources can check the correctness of the tally, as follows:

Phase 4: Verification of the tally

1. Compute $h' := \prod h(i)$ and check whether $h' \stackrel{?}{=} h^*$
2. Check whether r^* is indeed the decommitment value of h^* with respect to the tally published by the TA, i.e. whether $h^* \stackrel{?}{=} u(r^*; x_1^*, \cdots, x_l^*) \pmod q$. ◀

3.3 Suggested layouts for the receipt

In Step 2.2, the voter must confirm the OCR interpretation of the system, meaning she must compare a bit string on her receipt with the one produced by the system. Clever ballot layouts are necessary to make this process efficient.

Depending on the group G used for the UHDLC scheme, the value h will be about 200 bits for elliptic curves, or 1024 bits for the multiplicative group Z_p^*. Having a voter verify the correctness of many bits in a fast way is possible with some creativity.

One possibility is to use something like base 64 encoding, meaning that the sequence is cut in 6-bit chunks, each of which mapped to the caracters `[0-9a-zA-Z+/]`. In case of a 210 bit string, these leads to exactly 35 symbols. Suppose that these symbols are printed on the receipt using a fairly large point size, such that each symbol is at least one centimeter in height. The procedure is now as follows: the system scans and OCRs the receipt, and prints the recognized symbols using exactly the same layout and dimensions as used by the VM. If we assume that either the receipt or the printout is printed on transparent paper, the voter can easily check equality by putting one on top of the other; any difference will show clearly.

Even better results can be obtained using the techniques from [9], by representing the bits in a matrix in which a 0 is represented as and 1 as . Then if the two sheets are put on top of each other, two different bits will clearly show a square (or as a hole, if one sheet choses the bit complement representation). And by adding redundancy through the use of error-correcting codes, the authorities are either forced to cheat on many bits, or to create bit patterns that are inconsistent with the code.

Note that having the voter confirm the OCR result has the advantage that scrutineers do not have to OCR from the scanned image, and that no posterior dispute can arise about possible interpretation errors.

4 Properties of the protocol

We make the following assumptions:

(A) *The election authorities cannot break the discrete log problem for the parameters chosen before the elections ends.*
(B) *There exists a private channel between the voting machine and the Tallying Authority.* We will return to this point in Section 5
(C) *No information about the vote, other than what is sent through the private channel in previous item, leaves the voting booth.*
(D) *The voting machine always produces a correctly formatted* UHDLC. This assumption was discussed in Section 2.2.
(E) *The decommitment values of all unopened* UHDLC*s are properly and permanently destroyed after Phase 4 of the protocol is completed.*

Then the protocol has the following properties:

Unconditional privacy *For each voter i, the public view, including all receipts and all other data published on the bulletin board, reveals no information about the voter's choice.* As argued in CFSY, this is a straightforward forward consequence of the fact that the commitment scheme used is unconditionally hiding.

Correctness vote count *The voting machine and TA cannot change the tally.* As argued in CFSY, the only way the authorities can change the tally is if they can come up with different decommitment values $\langle r'(i)$ and $x'_1(i), ...x'_l(i)\rangle$ for h^*, but this is equivalent to breaking the discrete log problem in G, contradicting Assumption A.

Individual Voter Verifiability *Each voter can verify that his vote is included in the tally.* This follows from the fact that the value $h(i)$ printed on the receipt is also published on the bulletin board and is verifiably used in the tally process. CFSY does not have this property since in their setting there is no booth or receipt.

Universal Verifiability *Any observer can verify that the tally was calculated correctly.* This follows from Phase 3 and the homomorphic property of UHDLCs.

Ballot Casting Assurance As described, the voter has *no* guarantee that the UHDLC given to him by the VM is indeed a faithful encoding of his choice v. However, our protocol can be easily extended with the techniques of MarkPledge [17][1]

The TA knows the ballot Any entity that knows the decommitment values can figure out the vote. This is inherent to the use of an unconditionally hiding commitment scheme. We will mitigate this property in Section 5.5 considering various TAs.

The Voting Machine knows the ballot A general problem of using a voting machine is that the hardware knows which button was pressed, so it knows the voter's choice. We discuss this in Section 6.2.

5 How to communicate the decommitment values and add them

As is clear from the Figure 1, the voting machine must send the values $\langle h(i), r(i); x_1(i), \ldots, x_l(i)\rangle$ to the TA. As already mentioned, it is paramount that the individual values for $r(i), x_1(i), \ldots, x_l(i)$ remain private. Subsequently, the TA must count the votes as explained in Step 3.2. There exist various cryptographic techniques to implement this, each with its own characteristics.

5.1 Voting machine and tallying authority located on the same server

Observe that even though we described the TA and VM as two different entities, one can envision an implementation in which they reside on the same machine. For very small elections this might suffice.

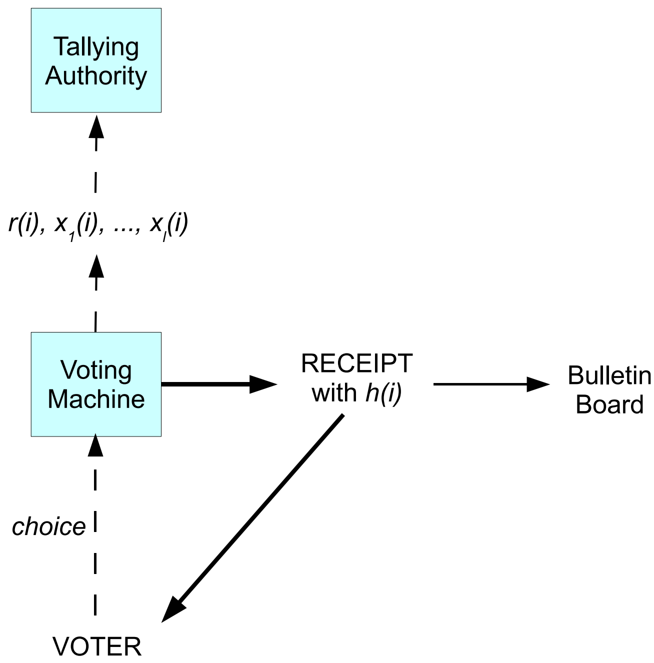

Fig. 1. Private channels between the VM and the TAs. The thin dashed arrows denote a private communication channel, the thin arrow denotes a public communication channel, the thick arrows denotes physical, public channel.

5.2 One-Time Pad

If the VM and the TA are located at different locations, they need a private channel to communicate. It is very tempting to use conventional techniques, such as symmetric encryption or public key encryption to encipher the communication between VM and TA. But these techniques do not provide unconditional privacy and therefore cannot be used.

Instead, the simplest solution is the **One-Time Pad**. This is easy: we may certainly suppose that the TA puts a sufficiently long random string in the VM which can be used as an encryption key.

5.3 Homomorphic encryption

Observe that there is no need for the TA to know the values $\langle r(i); x_1(i), ...x_l(i)\rangle$ of individual voters. It is sufficient if the TA can add them in order to find the tally. Homomorphic encryption, like the Paillier system, provides exactly this property: if $c_1 := E(m_1)$ and $c_2 := E(m_2)$ are the encryptions of two messages, then $c_1 * c_2 = E(m_1 + m_2 \pmod{M})$.

So the VM can send the values encrypted homomorphically to the TA through a private channel. The TA can add the values while they are encrypted, and only the results will be decrypted using a private key which is secret-shared among various election officials or authorities.

5.4 Using a Dining Cryptographer's net

This is the solution proposed by Bos in his thesis. He assumes the existence of a DC net which operates modulo $q - 1$. In the context of his thesis this DC net is implemented in an interactive way, so assuming that all voters are simultaneously online.

As already mentioned in Section 1.4 in [20] another implementation is suggested in which, after some initialization phase, each voter simply submits a long string, together with a proof that the format of this string satisfies all he restrictions. As the original DC protocol, this protocol comes in two flavors: one without any authority and in which each pair of voters exchanges random bits between them, and another, in which each voter exchanges random bits only with a small number of authorities.

5.5 Using a Verifiable Secret Sharing Scheme

This is the solution proposed by CSFY. They suggest to use Pedersen's VSS[18]. This scheme has the advantage that the authorities can sum the shares locally. I.e. the authorities can each separately do the necessary steps to tally the votes locally. Consequently, if a sufficient number of parties cooperate in reconstructing these values, they can reconstruct the tally and publish it.

6 Other observations

6.1 Ballot Casting Assurance

The protocol presented in Section 3.2 has the disadvantage that the voter must trust that the voting machine faithfully encodes the voter´s choice in the commitment. In [2], Adida and Neff present a very nice technique to allow a voter to verify that the voting machine created a correct ballot with the correct choice. In this technique, the machine commits itself towards the voter by printing a very cleverly constructed commitment of the voter´s choice. The voter now issues a random challenge, and the machine responds by printing a proof which has the property that it convinces the voter who has extra side-information, but does not reveal anything to an outsider.

6.2 Secret channels between the voter and the authorities

A second disadvantage of the protocol is that the machine gets to know the choice made by the voter, so the voter must trust that the machine does not leak this information. It would therefore be desirable to have protocols in which a voter can cast her vote *without* the machine knowing the choice.

Protocols like Prêt-à-Voter and PunchScan are fully paper-based in their original version and therefore have this property already. However, as is explained in [21], in Prêt-à-Voter the bottom layer of the ballot is identical for each voter, implying that the vote capture process can be automated. But instead of printing the image of screen containing the row marked by the voter, we encode the row marked as a UHDLC. A ballot, printed on transparent paper, is put on the display of a voting machine. The voter then puts his mark which is encoded using a UHDLC. in which BCA techniques can be used to verify that the encoding is faithful. So the voter's choice is split in two, as explained in [21]: the offset x, which is encoded in the UHDLC of the Ballot Issuing Authority, and the mark chosen by the voter, enoded in the UHDLC of the voting machine. As depicted in the diagram below, besides having a voting machine, we now also need a Ballot Issuing Authority.

However, with this modification it is no longer possible to use the vector encoding of votes and add them homomorphically, since a vote v is split between an offset x, and a mark y. In order to count the votes, the authorities must first add the commitments to x and y modulo m. Only after this step, they can proceed tallying the votes. There does not seem a straightforward way to recode a value $v \pmod{m}$ to the vector encoding of Section 3.3 using unconditionally hiding commitments without an authority getting to know v. The resulting protocol is in spirit equivalent to SplitBallot [16].

6.3 A protocol based on generic commitments

As explained in [21], by generalizing the techniques of Bennett and Rudich it is possible to obtain a commitment scheme in which two commitments can be added modulo an arbitrary integer N, and which is based on any bit commitment. It is therefore possible to emulate the homomorphic

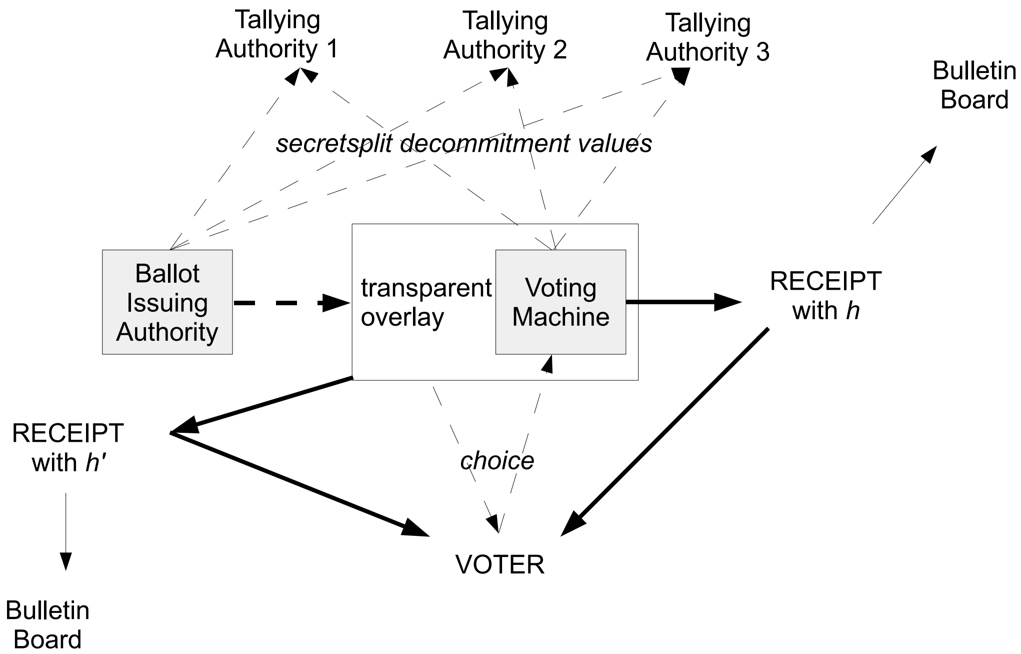

Fig. 2. Private channels between the Voter and the BIA and VM. The thin dashed arrows denote a private communication channel, the thin arrow denotes a public communication channel, the thick dashed arrow denotes a physical, private channel. the thick arrows denotes physical, public channel.

properties of UHDLCs, use the encoding of the vote presented in 3.1, and the protocol. By using an underlying commitment scheme that is universally composable and extending the proofs presented in [13], it is possible to prove that the complete voting protocol is UC; this is subject of current research. The resulting protocol would be similar to the one presented in [15], but is simpler to understand.

One difference with [15] is the encoding of the vote: they use expressions of the form $g_0^r g_1^{H(``Alice'')}$, where H is a collision-resistant hash function. With this encoding the commitments can not be used to tally the votes, but it is possible to mask (blind) the votes, put them in a table, and use a straightforward cut-and-choose protocol (at the expense of an expansion by a factor k) which allows public verification that the tally of the votes is correct. Note that [15] also use ballot casting assurance.

6.4 Scratch & Vote and MarkPledge

Observe that the encoding of the vote is the equivalent of homomorphic counters (see for instance Adida [1]) but offering unconditional privacy, not computational. It therefore seems possible that the Scratch & Vote protocol of Adida and Rivest[3] can be modified to this setting.

MarkPledge 1 [2] uses the same vector encoding of the vote as we do, but then uses homomorphic encryption based on ElGamal, which offers computational privacy only. Applying UHDLCs instead, we get Protocol 1 with ballot casting assurance, unconditional privacy, and a mixing/auditing process which is much simpler. MarkPledge 2 was motivated by reducing the ballot size, but UHDLCs are already very compact anyway.

6.5 Applying the protocol to internet voting

A security concern is now that the TA cannot longer trust that the $h(i)$ received over some HTTP connection (behind which supposedly sits a legitimate voter) are of the right format. In the one-authority scenario this is not a problem, however, since the TA sees the values $h(i)$, $r(i)$ and $x_1(i), \ldots .x_l(i)$ anyway, and he simply rejects these votes and published them on the bulletin board. Another possibility is to use homomorphic encryption; see below.

The upshot is this: in situations where voters are much more concerned with having their vote included in the tally and either trust the election organizers sufficiently with the privacy of their vote or do not really care, this protocol is attractive, especially since the math is extremely simple and no complicated mixing is involved.

Observe that the length of the proof is proportional to the number of candidates, l. When this number is large, the receipt might become quite long. Therefore, an alternative strategy is to only print the pledge strings on the receipt, and a cryptographic hash of the other information. The VM sends this information to the election website where all proofs are verified. The voter needs to verify that the hash computed corresponds with the hash printed on his receipt.

7 Conclusion

The author strongly believes that the success of voting protocols depends in part on their simplicity. Protocols that cannot be explained to colleague during lunch with some scribles on a napkin are probably doomed. In this respect, voting research is different from cryptographic protocol research. The protocol presented in this paper is fairly simple to explain and to implement; in addition, it provides unconditional (that is, eternal) privacy.

Acknowledgments

Withheld.

References

1. B. Adida. *Advances in Cryptographic Voting Systems*. PhD thesis, Massachusetts Institute of Technology, 2006.
2. B. Adida and C. A. Neff. Ballot casting assurance., 2006. Available online: `http://d8ngm cuv6pmeemmv . olli eefood.rest events/evt06/tech/full_papers/adida/adida.pdf`.
3. B. Adida and R. L. Rivest. Scratch & Vote: self-contained paper-based cryptographic voting. In *WPES 2006 — ACM Workshop on Privacy in the Electronic Society*, pages 29–40. ACM, 2006.
4. J. Bos. *Practical Privacy*. PhD thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 1992. Available online: http://alexandria.tue.nl/extra3/proefschrift/PRF8A/9201032.pdf.
5. J. Bos and G. Purdy. A voting scheme. Rump session of CRYPTO 88. Does not appear in proceedings.
6. S. Brands. *Rethinking Public Key Infrastructures and Digital Certificates*. MIT Press, 2000.
7. A. Broadbent and A. Tapp. Information-Theoretically Secure Voting Without an Honest Majority. WOTE 2008 — IAVoSS Workshop On Trustworthy Elections, Leuven, Belgium, 2008.
8. D. Chaum. Showing credentials without identification: Signatures transferred between unconditionally unlinkable pseudonyms. In *EUROCRYPT*, pages 241–244, 1985.
9. D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security & Privacy*, 2(1):38–47, 2004.
10. D. Chaum, I. Damgård, and J. van de Graaf. Multiparty computations ensuring privacy of each party's input and correctness of the result. In *Advances in Cryptology - CRYPTO '87*, LNCS 293.
11. D. Chaum, A. Essex, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. Rivest, P. Ryan, E. Shen, A. T. Sherman, and P. L. Vora. Scantegrity ii: End-to-end voter-verifiability by voters of optical-scan elections through confirmation codes. *IEEE Transactions on Information Forensics and Security*, 4(4):611–627, 2009.
12. R. Cramer, M. K. Franklin, B. Schoenmakers, and M. Yung. Multi-autority secret-ballot elections with linear work. In *EUROCRYPT '96*, LNCS 1070, pages 72–83. Springer, 1996.
13. G. Estren. Universally composable committed oblivious transfer and multi-party computation assuming only basic black-box primitives. Master's thesis, School of Computer Science, McGill University, Montreal, Canada, 2004.

14. B. Hosp and S. Popovenuic. Punchscan Voting Summary. Version dated Feb 13, 2006, obtained from first author, 2006.
15. T. Moran and M. Naor. Receipt-Free Universally-Verifiable Voting with Everlasting Privacy. CRYPTO 2006, 2006.
16. T. Moran and M. Naor. Split-ballot voting: everlasting privacy with distributed trust. In *Proceedings CCS 2007*, pages 246–255. ACM, 2007.
17. C. A. Neff. Practical high certainty intent verification for encrypted votes., October 2004. Available online: `http://8uum wgg w. olli eefood.rest/vhti/documentation/vsv-2.0.3638.pdf`.
18. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology - CRYPTO '91*, LNCS 576, pages 129–140. Springer, 1991.
19. S. Popovenuic and B. Hosp. An Introduction to Punchscan. Version dated Oct 15, 2006. http://punchscan.org/papers/popoveniuc_hosp_punchscan_introduction.pdf, 2006.
20. J. van de Graaf. Anonymous one-time broadcast using non-interactive dining cryptographer nets with applications to voting. In *Anais do VII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 68–79. Sociedade Brasileira de Computação, 2007.
21. J. van de Graaf. Voting with unconditional privacy by merging Prêt-à-Voter and PunchScan. *IEEE Transactions on Information Forensics and Security*, 4(4):674–684, 2009.