Voting with unconditionally privacy: CFSY for booth voting

Jeroen van de Graaf
Departamento de Computação
Universidade Federal de Ouro Preto
jvdg@iceb.ufop.br

November 26, 2009

Abstract

In this note we simplify the Cramer, Franklin, Schoenmaker and Yung internet voting protocol to the booth setting. In it, objects of the form $g_0^r g_1^{x_1} \cdots g_l^{x_l}$ (mod 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, we show that almost all existing protocols that provide unconditional privacy use or could benefit from these commitments. Even though we present no novelty from a cryptographic perpective, the protocol presented is interesting from a voting perspective because it is simple enough to be understood by non-cryptographers, yet very powerful.

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 [BT08]).

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 [Cha85] 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

Before discussing unconditionally private voting protocols, we present a mathematical object with many interesting cryptographic properties, and which play an important role in many of the protocols presented in the next Subsection.

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 [CDvdG] (see page 98). These commitments are computationally binding, provided that the party who commits cannot break the discrete log in the group G.

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 too, but this possibility is not explored here. See [Bra00].)

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 *UHDLC*s 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, see Appendix A.

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 CFSY, 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 [Nef04][Adi06]
- 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 [MN06], SplitBallot [MN07], MarkPledge [AN06] and Scratch & Vote[AR06]

1.4 A short history of unconditionally private voting protocols

To the best of our knowledge, the first voting protocol that provides unconditional privacy was presented by Bos [BP], [Bos92]. 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[CFSY96]. Their basic version also uses $l = 1$ but it encodes a masked vote of two options as $\{-1, 1\}$. It uses Pedersen secret sharing[Ped91] 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 [vdG07a], 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 the paper for discussion).

Simultaneously, Chaum invented PunchScan (www.punchscan.com; for a technical description see [PH06, HP06]) An important theoretical contribution of PS is that it showed how to build an election protocol based on bit commitment, though the original papers used conventional symmetric encryption as a commitment scheme, yielding computational privacy only.

In [vdG07b] (see also [vdG09]), 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 [MN06] 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[MN07] is strongly based on PunchScan, but 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.

2 Preliminaries

2.1 Voting terminology

We distinguish the following entities and, for this draft version, we assume that the reader is familiar with their role,

- voters
- TA=Tallying Authority
- BIA=Ballot Issuing Authority
- bulletin board
- auditors/scrutineers
- VM=Voting Machine
- privacy, correctness, ...

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 completely control the hardware and software which the vote is generated, 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 v*th* 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 hs 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 on election day

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 by the TA
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:

$$\begin{array}{ll} h^* := \prod h(i) \pmod q & \text{product of the UHDLC} \\ r^* := \sum r(i) \pmod {q-1} & \text{sum of the random values} \\ x_1^* := \sum x_1(i) \pmod {q-1} & \text{sum of the votes for candidate 1} \\ \ldots & \ldots \\ x_l^* := \sum x_l(i) \pmod {q-1} & \text{sum of the votes for candidate } l \end{array}$$

   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 $h' \overset{?}{=} h^*$

2. Check whether r^* is indeed the decommitment value of h^* with respect to the tally published by the TA, i.e. whether $h^* \overset{?}{=} u(r^*; x_1^*, \cdots, x_l^*) \pmod q$.

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 [], For instance, represent 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 UHDLCs 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.* 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.* 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.

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 [Nef04][Adi06] See Appendix B for discussion.

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 .

5 How to communicate the decommitment values and add them

As is clear from the Figure 1, the voting machine must send the values $\langle h(i),\ r(i); x_1(i), \dots, x_l(i)\rangle$ to the TA. As already mentioned, it is paramount that the individual values for $r(i), x_1(i), \dots, x_l(i)$ 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.

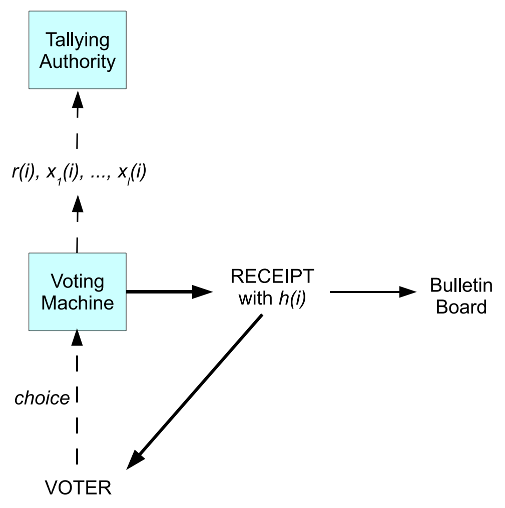

Figure 1: Private channels between the VM and the TAs. The thin dashed arrows denote a private communication channel, the thin arrow denotes a public communication channel, the thick arrows denotes physical, public channel.

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 [vdG07a] 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. 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 [AN06], 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. For more details, see Appendix B.

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 [vdG09], in PaV 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 it is in Merge: 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 [MN07].

6.3 A protocol based on generic commitments

As explained in [vdG09], 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 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 [Est04], 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 [MN06], but is simpler to understand.

One difference with MN 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 MN also use ballot casting assurance.

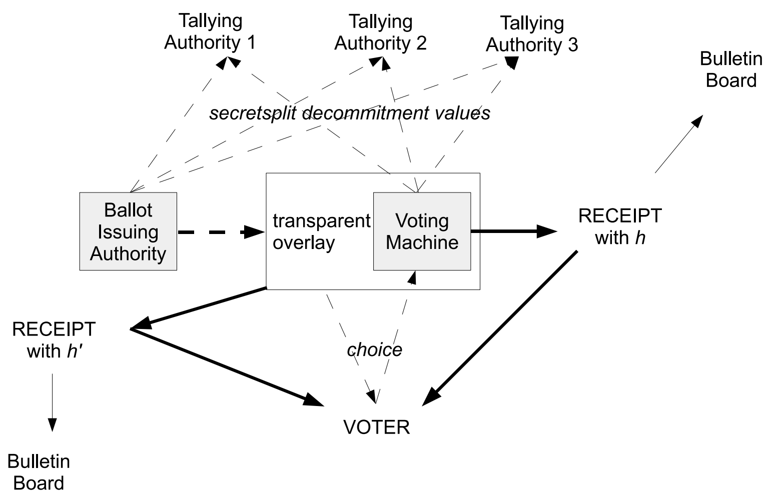

Figure 2: Private channels between the Voter and the BIA and VM. The thin dashed arrows denote a private communication channel, the thin arrow denotes a public communication channel, the thick dashed arrow denotes a physical, private channel. the thick arrows denotes physical, public channel.

6.4 Scratch & Vote and MarkPledge

Observe that encoding of the vote is the equivalent of homomorphic counters (see for instance Adida [Adi06]) but offering unconditional privacy, not computational. It therefore seems possible that the Scratch & Vote protocol of Adida and Rivest[AR06] can be modified to this setting.

MarkPledge 1 [AN06] 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 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 explain and to implement; in addition, it provides unconditional (or, if you like, eternal) privacy.

Acknowledgments

Stefan Brands helped in confirming that his digital credentials can be converted to unconditionally hiding commitments. This was before I realized that this idea was, in fact, already mentioned in CFSY.

References

[Adi06] Ben Adida. *Advances in Cryptographic Voting Systems*. PhD thesis, Massachusetts Institute of Technology, 2006.

[AN06] Ben Adida and C. Andrew Neff. Ballot casting assurance., 2006. Available online: `http://www.usenix.org/events/evt06/tech/full_papers/adida/adida.pdf`.

[AR06] Ben Adida and Ronald 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.

[Bos92] J.N.E. Bos. *Practical Privacy*. PhD thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 1992. Available online: http://alexandria.tue.nl/extra3/proefschrift/PRF8A/9201032.pdf.

[BP] J.N.E. Bos and G. Purdy. A voting scheme. Rump session of CRYPTO 88. Does not appear in proceedings.

[Bra99] S. Brands. A technical overview of digital credentials, 1999.

[Bra00] Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates*. MIT Press, 2000.

[BT08] Anne Broadbent and Alain Tapp. Information-Theoretically Secure Voting Without an Honest Majority. WOTE 2008 — IAVoSS Workshop On Trustworthy Elections, Leuven, Belgium, 2008.

[CDvdG] David Chaum, Ivan Damgård, and Jeroen 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.

[CFSY96] Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-autority secret-ballot elections with linear work. In *EUROCRYPT '96*, LNCS 1070, pages 72–83. Springer, 1996.

[Cha85] David Chaum. Showing credentials without identification: Signatures transferred between unconditionally unlinkable pseudonyms. In *EUROCRYPT*, pages 241–244, 1985.

[Est04] Gregory 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.

[HP06] B. Hosp and S. Popovenuic. Punchscan Voting Summary. Version dated Feb 13, 2006, obtained from first author, 2006.

[MN06] Tal Moran and Moni Naor. Receipt-Free Universally-Verifiable Voting with Everlasting Privacy. CRYPTO 2006, 2006.

[MN07] Tal Moran and Moni Naor. Split-ballot voting: everlasting privacy with distributed trust. In *Proceedings CCS 2007*, pages 246–255. ACM, 2007.

[Nef04] C. Andrew Neff. Practical high certainty intent verification for encrypted votes., October 2004. Available online: `http://votehere.net/vhti/documentation/vsv-2.0.3638.pdf`.

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology - CRYPTO '91*, LNCS 576, pages 129–140. Springer, 1991.

[PH06] S. Popovenuic and B. Hosp. An Introduction to Punchscan. Version dated Oct 15, 2006. http://punchscan.org/papers/popoveniuc_hosp_punchscan_introduction.pdf, 2006.

[vdG07a] Jeroen 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.

[vdG07b] Jeroen van de Graaf. Merging Prêt-à-Voter and PunchScan. In *Anais do VII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 207–210. Sociedade Brasileira de Computação, 2007.

[vdG09] Jeroen van de Graaf. Voting with unconditional privacy by merging Prêt-à-Voter and PunchScan. In *TIFS*. IEEE, 2009.

A Properties of discrete log commitments

Brands defines expressions of the form $h = g_0^r g_1^{x_1} \cdots g_l^{x_l}$, but instead of using h as a commitment, it sent to a (certification) authority who issues a signature on h. This constitutes a credential. Then Brands shows how an individual can open only some of the x_i of the credential, or show linear relations in $\mathcal{Z}$ between the x_i, etc. These techniques are very powerful, we will only use a subset of them.

Definition 1 *Let an uncondionally hiding commitment be an expression of the form*

$$h = g_0^r g_1^{x_1} \cdots g_l^{x_l} \in G.$$

Here G is a group in which computing the Discrete Log is hard, like the multiplicative group $\pmod p$ for p prime and sufficiently large, or a group induced by an elliptic curve.

The x_i, of which there are l in number, are the contents, *whose values are defined $\pmod q$, with $q := \#G$. The number r is chosen uniformly random from $\{1, \ldots, q\}$, and is called the* key *to open the commitment.*

Straightforward implications of Brands' work are the following useful properties:

Theorem 2

(a) UHDLCs are constant size independent of the values of l and of the x_j.
(b) UHDLCs are uncondionally hiding.
(c) UHDLCs are computationally binding: the committer can change a value x_j only if he can break the (single) Discrete Log Problem.
(d) The committer can convince any party that two UHDLCs are commitments to the same value in an efficient witness indistiguishable proof, and these proofs can be repeated an arbitrary number of times.
(e) It is possible to open an arbitrary subset of x_j, without revealing any information about the unopened ones.
(f) It is possible to modify UHDLCs such that they have a homomorphic property modulo m, where m divides q.

Proof: (a) This is trivial. Note that if we use elliptic curves h can be only 190≈200 bits, roughly 14 × 14 pixels, or 40 symbols using an alphabet with $2^5 = 32$ symbols.

(b) Let $h := g_0^r g_1^{x_1} \cdots g_l^{x_l}$. Then for any $x'_1, \ldots x'_l$ there exists one r' such that $h := g_0^{r'} g_1^{x'_1} \cdots g_l^{x'_l}$, and since r is chosen randomly, any $x'_1, \ldots x'_l$ is equaly likely. Therefore h reveals no information about $x_1, \ldots x_l$.

(c) This follows from Proposition 2.3.3 on page 60 Brands' thesis[Bra00] (see also Property 1 in [Bra99]).

(d) First let us sketch the protocol discussed in §2.4.3 of Brands' thesis, in which A shows that she knows the decommitment values r and $x_1, \ldots x_l$ of h:

	Alice		**Bob**
0	$h := g_0^r g_1^{x_1} \cdots g_l^{x_l}$	$\xrightarrow{\quad h \quad}$	
1	$a := g_0^s g_1^{w_1} \cdots g_l^{w_l}$	$\xrightarrow{\quad a \quad}$	
2		$\xleftarrow{\quad c \quad}$	c is a random challenge
3	$b := g_0^{cr+s} g_1^{cx_1+w_1} \cdots g_l^{cx_l+w_l}$	$\xrightarrow{\quad b \quad}$	$b \overset{?}{=} h^c a$

Note that if $c \in \{0, 1\}$ we get a perfect zero-knowledge protocol, but for $c \in \{1, \ldots, q\}$ we get the Schnorr variation. In Proposition 2.4.8 of [Bra00] it is shown that this protocol is perfectly witness indistiguishable, and that the case of arbitrary l is as secure as the case $l = 1$, which is the Schnorr case.

Now to show that $h(1)$ and $h(2)$ encode the same x_i, it is suficient to show that A knows the decommitment value of $h(1)/h(2) = g_0^{r(1)-r(2)} g_1^{x_1(1)-x_1(2)} \cdots g_l^{x_l(1)-x_l(2)} = g^{r(1)-r(2)}$ with respect to the basis consisting of g_0 only.

(e) This follows from Chapter 3 of [Bra00]. As a special case for opening only one content field x_i, see the protocol in Figure 4 of [Bra99].

(f) Let m divide q and define a commitment to a value $x \pmod m$ as $h = g_0^r g_1^{wm+x}$, with w random in $\{1, \ldots, (q)/m\}$. Obviously this commitment scheme is homomorphic $\pmod m$. Showing equality between two commitmentent $x(1)$ and $x(2)$ modulo m can accomplished by proving knowledge of

$$h(1)/h(2) = g_0^{r(1)-r(2)} g_1^{(w(1)-w(2))m+(x(1)-x(2))} = g_0^{r(1)-r(2)} g_1^{(w(1)-w(2))m}$$

with respect to $\langle g_0, g_1^m \rangle$. In this example we used $l = 1$, but it is obvious that the technique can be generalized to arbitrary l. ◀

B Ballot casting assurance

In [AN06] a very nice technique is presented to allow a voter to verify that the voting machine created a correct ballot with the correct choice. To this end the voter submits a random challenge of L bits; a cheating voting machine gets caught with a probability of 2^{-L}.

We will briefly explain this technique using a simple example. Let there be 4 candidates where our voter chooses number 3, so we must encode the vector $\overline{x} = (0, 0, 1, 0)$. Now if the voting machine wanted to prove that the ballot it printed really corresponds to a vote for 3, then it could simply print the decommitment value. This proof would give the voter absolute security. but that would reveal the whole vote.

We settle therefore for something almost as good: the voting machine proves it printed the right ballot with a very high probability, $1-2^{-L}$. In addition, the technique introduced allows us to *fake* similar proofs for options 1, 2 and 4 in such a way that the voter, who sees some additional information *not* printed on the receipt, is convinced that the commitment encodes his choice, 3. But anybody else, without this extra information, cannot distinguish between the 4 options; the real proof is embedded in 3 fake proofs which, to any other person, are equally convincing.

In the technique proposed by Neff, each entry of the zero/one vector $\overline{x}$ representing the vote, is not encoded as one simple bit commitment. Instead the BennettRudich (BR) encoding is used. The idea is to represent each bit commitment as a vector (again) of pairs of simple bit commitments, such that each pair $\langle x_L, x_R \rangle$ XORs to the committed bit: $x_L \oplus x_R = x$. This allows for challenges on one half of the bit commitment, without revealing its value. (In fact, Neff uses the complement encoding $x_L \oplus x_R = x \oplus 1$, but for the success of the technique this is irrelevant.)

For concreteness suppose a BR encoding of length $L = 5$, then we could have (ignore the underlining for a second):

$$\begin{array}{c} 0 = (0\underline{1}001, \underline{0}1\underline{001}) = BR(1) \\ 0 = (1\underline{0}101, \underline{1}0\underline{101}) = BR(2) \\ 1 = (1\underline{1}100, \underline{0}0\underline{011}) = BR(3) \\ 0 = (0\underline{0}110, \underline{0}0\underline{110}) = BR(4) \end{array}$$

One easily verifies that the bitwise xor of the bitstrings correspond to the values in the first column. Note that each bit is committed to individually, so in this example we have a total of $4 \times 5 \times 2 = 40$ bit commitments.

For the sake of exposition, let us first show an incomplete protocol that will convince the voter that this set of 40 commitments indeed encodes a 3, but does not hide this from other persons. After the voter has pressed his choice, 3 in this example, the voting machine makes a commitment (pledge) by printing the left bit values of $BR(3)$. In our example this pledge is 11100. Recall that these values are random. Now the voter enters a challenge, for example c =RLRRR. When applied to $BR(3)$, this corresponds to the string 01011 (the bits that are undelined). Note that the voter can compute this from the pledge. The voting machine must respond to the challenge by opening half of $BR(3)$ according to the challenge string. In other words, it must open the bit commitments of the following bits and show their value:

$$\begin{array}{c} BR(3)_{1R} = 0 \\ BR(3)_{2L} = 1 \\ BR(3)_{3R} = 0 \\ BR(3)_{4R} = 1 \\ BR(3)_{5R} = 1 \end{array}$$

Note that if the voting machine were lying and $BR(3)$ encoded a 0, then the voting machine would be caught with prob 1/2 for each pair, yielding 2^{-5}. Also note that we do not actually expect the voter to do this verification in the booth. We only assume that that the Voter emits a random challenge, whereupon the voting machine finishes the printing of the receipt which the voter can take home or to a helper organization to check for correctness.

However, as it is, this protocol is *insecure* because it reveals the voter's choice. Therefore we will camouflage this text, by embedding the *true* proof that $BR(3) = 1$ with *fake* proofs of $BR(1) = 1, BR(2) = 1$ and $BR(4) = 1$. These fake proofs are obtained as follows: for each pair of $BR(1), BR(2)$ and $BR(4)$, the voting machine opens only one side, left or right, according to the challenge string RLRRR, thus obtaining $BR(1)|_{LRLLL}$ = 01001, $BR(2)|_{LRLLL}$ = 10101 and $BR(4)|_{LRLLL}$ = 00110 respectively (the underlined bits).

Obviously, this is not consistent with the pledge string p^* entered by the user. The trick, however, is to compute *fake* pledge strings p_1, p_2 and p_4 that are consistent with $BR(1) = BR(2) = BR(4) = 1$. In other words, to compute the fake pledge strings which correspond with 01001, 10101 and 00110, we must take these strings in combination with the challenge RLRRR but make it consistent with the values in the first component of each pair, i.e. consistent with LLLLL and the fact that the BC is (supposedly) 1. This implies that for all positions were the challenge string equals R, we must flip the bit. So writing RLRRR as 10111 it means a bitwise xor: $01001 \oplus 10111 = 11110 = p_1$, $10101 \oplus 10111 = 00010 = p_2$, $00110 \oplus 10111 = 10001 = p_4$. Observe that $01011 \oplus 10111 = 11100$, which is, of course, exactly the only true pledge string.

The complete protocol now is as follows: after entering the challenge, the voting machine prints the pledges p_1, p_2, p_3, p_4 and the decommitment values of $BR(1)|_{RLRRR}, BR(2)|_{RLRRR}, BR(3)|_{RLRRR}, BR(4)|_{RLRRR}$. To the voter, who verifies that the pledge string p^* equals p_3, this is convincing. But to an outsider, who has no access to p^*, all 4 possible scenarios are equally convincing, since each of them is consistent and there is no way to figure out which was the one actually followed.

Summarizing: the receipt is created as follows:
1. The voting machine prints the pledge $p^* = BR(3)|_{LLLLL}$. This part will be destroyed when the voter casts his vote.
2. The voting machine prints the bit commitments $BR(1); BR(2); BR(3); BR(4)$.
3. The voting machine prints the challenge c entered by the user.
4. The voting machine prints the 4 response strings p_1, p_2, p_3, p_4. The voter must verify that p^* equals p_3. If not she must walk out of the booth and complain.
5. The printer also prints the decommitment values of $BR(1), BR(2), BR(3)$ and $BR(4)$ in agreement with the challenge c, showing that $BR(i) = 1$ in each of these cases. The VM can open $BR(1), BR(2)$ and $BR(4)$ as a 1, because the "pledge" strings p_1, p_2, p_4 are fake, they were computed retroactively, to be consistent with p^* and $BR(i)|_{LLLLL}$.

So the protocol only makes sense because the voter, when in the booth, sees the pledge string p^* being printed **before** she gives a challenge c to the voting machine, and because she checks that p^* is equal to p_3. Once p^* has been torn off and destroyed, nobody can distinguish which of the 4 decommitment values correspond to the voter 's choice, since all of them show that the $BR(j)$ equals 1.

One way summarize this protocol is to say that each of the 4 original bit commitments 0, 0, 1 and 0 has been replaced by 5 copies, each consisting of a pair. This pair-property on the one hand allows for a cut-and-choose, and on the other hand allows for simulating (faking) the view of the protocol to open all zeroes as ones. This property is formalized by saying that the protocol is Honest-Verifier-Zero-Knowledge.