Bare-Handed Electronic Voting with Pre-processing

Ben Riva[1], Amnon Ta-Shma[2]

School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel

[1]benriva@post.tau.ac.il, [2]amnon@post.tau.ac.il

Abstract

Many electronic voting schemes assume the user votes with some computing device. This raises the question whether a voter can trust the device he is using. Three years ago, Chaum, and independently Neff, proposed what we call *bare-handed* electronic voting, where voters do not need any computational power. Their scheme has a very strong unforgeability guarantee. The price for that, however, is that they require the voter to tell his vote to the voting booth.

In this paper we propose a scheme where the voter votes bare-handedly, and still maintains his privacy even with respect to the voting booth. We do this by allowing the voter the use of a computer device but only at a pre-processing stage - the voting itself is done bare-handedly. This has many advantages. A voter who has to verify calculations at the booth has to trust the software he is using, while a voter who verifies pre-processed calculations can do that at his own time, getting help from whatever parties he trusts.

Achieving private, coercion-resistance, bare-handed voting with pre-processing is a non-trivial task and we achieve that only for elections with a *bounded* number of candidates. Our solution works by proposing an extension to known voting schemes. We show that such extended schemes enjoy the same unforgeability guarantee as that of Chaum and Neff. In addition, our extended scheme is private, and the voter does not reveal his vote to the booth.

KEYWORDS: electronic voting, receipt freeness, coercion resistance, universal verifiability.

1 Introduction

There are several basic properties required from an electronic voting protocol. A voting scheme has to be *unforgeable*, i.e., even a coalition of (computationally unbounded) adversaries can not forge the voting results. Also, it has to be *private*, meaning that an adversary can not learn how a specific voter voted. Another more subtle property is that of *coercion-resistance* which basically means that a voter can deny his vote. Finally, we would like the system to be *auditable* (also called *verifiable*), meaning that all actions taken during the election are written down on a public board open for inspection and verification by *everyone*.

There are many proposals for electronic voting schemes. Many of these schemes require that the voter uses computational power in the booth. The underlying assumption is that honest voters can control the algorithm they run. However, we can (and should) question this assumption. Viruses and spy-ware are a common reality today. How can one be sure that the algorithm one runs is indeed the intended one?

This led David Chaum [Cha04], and independently Andrew Neff [Nef04], to suggest the notion of what we term as *bare-handed voting*. The idea is that the voter comes to the voting booth without any computational power and manually verifies that his vote is properly processed (e.g., using his eyes and visual cryptography in Chaum's scheme). A very appealing aspect of this approach is that the auditors (and anyone can be an auditor) can verify the validity of the

elections in real time. As a result the system is truly unforgeable.

One way to view Chaum's and Neff's algorithms is that the voter delegates his computations to the voting booth, and his only role is to check his vote is correctly registered. The price of this approach is that the voting booth knows what each voter voted. Thus, in terms of privacy, the system is less satisfactory. For example, a government can easily find out what each citizen voted.

Recent schemes (e.g., [CRS05, LTR+06, AR06, Cha07]) use paper based voting where the paper ballots can be prepared in advance by one or more authorities. For example, in [Cha07] one authority prepares the ballots, in [CRS05] one authority prepares the ballots, but the ballots are encrypted with a cascade mixing using the public keys of several parties, and in [LTR+06] the encryption is distributed.

It is important to understand that there is no privacy towards the party (parties) that prepare the ballots. The above schemes transfer the point of failure. In [Cha04] we trust the booth and in the above schemes we have to trust the party that prepares the ballots. For example, if the government prepares the ballot then there is no privacy towards the government.

Moreover, even if we trust the parties who prepare (and encrypt) the ballots, there is still a severe privacy problem with existing schemes. Suppose some party A can watch the encrypted ballots before they are being used. Then, that party knows the matching between candidates and encryptions (that appears on the ballots). After a ballot is used, the published information on the public board contains the voter name and an encrypted value, and therefore party A knows exactly what the voter voted. This problem is the reason why a distributed encryption does not guarantee privacy against the ballot creators.

Thus, current schemes we aware of, either require some computational power from the voter at the booth, and then in return give the voter full privacy, or do not require computational power from the voter at the booth, but as a result the voter loses his privacy against the party that prepared the ballot. In this paper we show how to maintain privacy (even against the government) without requiring the voter to have computational power at the booth. We do that by letting the voter prepare the ballot himself. This raises several problems which we discuss now.

1.1 Bare-handed voting with pre-processing

In this paper we consider bare-handed voting *with pre-processing*. In our model, voters need computational power but only at a pre-processing stage. They later on come to the voting booth (with the pre-processed papers) and vote bare-handedly. The pre-processing stage resembles preparing paper ballots in current manual elections. Any user can prepare any number of pre-processed ballots in the pre-processing stage. He can also choose to test the ballots or any (random) subset of them. Subsequently, the voter comes equipped with the pre-prepared ballots to the voting booth and manually votes. In the booth we require only simple human abilities such as: reading and the ability to compare strings.

In our scheme the voter prepares the ballots at home. This has a privacy advantage, but potentially makes the scheme coercible. Never the less, our protocol supplies a strong guarantee against coercion. We assume a powerful coercer that can give coerced ballots to the voter, and make sure the voter has no other ballots with him. We show that if a coercer can coerce a voter, then the coercion is detected with a good probability. This, in particular, implies that a coercer can not coerce many people to vote without being detected. We describe how this is done in Section 4.1.

One might ask why the use of a computer outside the booth is safer then the use of a computer device inside the booth (e.g., [BFP+01]). However, note that the voter has no way to check how his device functions inside the booth. Moreover, he can be coerced to use a malicious device. In contrast, a voter has a choice how to prepare his pre-processed ballots: he can download an open-source software, program such a software himself or use a public web-site (or his favorite candidate web site) for that. Furthermore, he can create as

many ballots as he wishes, and therefore he can choose a subset of the created ballots and check its validity.

We give an intuitive discussion of the problem and our solution in Section 4.1.

2 The participants, required properties and the attack model

We have voters, voting booths, trustees and auditors. As with many other schemes we have a *public board* which is a reliable database accessible by everyone. The auditors have access only to this public board and constantly check its integrity (data is only added to the database, old data does not change, everyone gets to see the same picture) and its contents (proofs are correct etc.). Everyone can be an auditor. One may think of this public board as an Internet site where all data is accumulated, and where its reliability stems from the fact that it is under constant public inspection. The assumption that such a public board can be maintained is made in many previous works (e.g., [HS00, Cha04]).

Some very basic requirements from an electronic voting scheme (stated in a very informal way) are:

Unforgeability - No one can falsify the result of the voting.

Eligibility, Unreusability - Respectively requires that only eligible voters vote and no voter can vote twice.

Auditability, Universal auditability - The first describes the ability of any individual voter to determine whether or not his vote has been correctly placed. The second corresponds to the ability of any auditor to determine that the whole protocol was followed correctly, given that votes had been correctly placed.

Robustness - Dishonest participants can not disrupt the voting. In particular cheating players should be detected and it should be possible to prove their malicious behavior and finish the voting process without their help.

Privacy - No one can link a voter with his vote.

Receipt-freeness, Coercion resistance - The notion of receipt freeness was introduced by Benaloh and Tuinstra [BT94], and it means that the voter can not prove to which candidate he voted. This notion can be generalized in several ways. The strongest one, usually called *coercion resistance*, avoids even scenarios where the voter cooperates with the coercer, and they both try to find a strategy where the voter can prove that he followed the coercer instructions (e.g., they can choose specific private keys and a strategy such that the voter can prove that he voted a specific value or a random value). A formal definition was given in Juels, Catalano and Jakobsson [JCJ05].

For unforgeability, auditability and universal auditability, we assume the malicious party includes any subset of malicious voters, the voting booth and all of the trustees. We assume the malicious party is computationally unbounded. The requirement is that if the malicious party changed the vote of t honest voters then it is caught cheating with probability at least $1 - 2^{-\Omega(t)}$.

There are many ways to define privacy, the most appropriate one is probably saying that the information the adversary holds is computationally close to a distribution that has very low mutual information with the actual mapping between voters and votes, and this should hold even if there is some a-priori knowledge on voting patterns. Such a definition protects not only individuals but also groups of persons (e.g., it will not leak information on the way a certain minority group voted). In any case, we inherit the privacy guarantee that we get from the underlying scheme that we use. For privacy, we restrict ourselves to computationally bounded adversaries. We allow the adversary to consist of a coalition of the voters, the booth and some of the trustees (the exact number of trustees depends on the underlying scheme).

Finally, for coercion resistance, the adversary

is computationally bounded. We allow the coalition of malicious players to include the voters, the coercer and some trustees (again, depending on the underlying scheme). We assume the booth does not cooperate with this attack.[1] We also need to use what we call a *recordable private channel* between the voter and the booth. A recordable private channel between two parties A and B is an untappable channel between A and B that has the following two properties: First, at the request of one of the players, the channel can be examined by an auditor (this is the reason we call the channel *recordable*), and, second, at the end of the conversation, if the two parties agree, the recording is erased and lost.

The first property is important for robustness, and the second for coercion resistance. This assumption requires some physical implementation, e.g., a printer printing the transcript between the two parties, where later on the printout is shredded. Similar definitions appear in previous works in the area. In Sako and Kilian [SK95] the channel is defined to have the second property only (and indeed no robustness is supplied), in Chaum's visual scheme proposal [Cha04] he assumes parts of the transcript can be shredded. We discuss this in more detail in Section 5.

3 Previous work

Unforgeability is usually easy to achieve. Privacy is also easy, but only against passive adversaries, e.g., in a scenario where dishonest votes are independent of honest votes. If we allow active adversaries, e.g., if dishonest players can vote based on what they see so far on the public board, then privacy is sometimes not guaranteed [Pfi94].

Coercion resistance and even receipt freeness are usually more difficult to achieve. Benaloh and Tuinstra proposed a receipt free scheme which was later broken [Hir01]. Sako and Kilian [SK95] proposed a receipt free scheme using mix-networks and Chameleon blobs but their scheme requires the voter to know at least one mix which is honest (rather than just knowing that one such mix exists). [HS00] proposed a similar but more efficient solution using threshold encryption, but it has the same drawback. Moreover, both schemes can be coerced.[2] [MBC01] proposed a solution which uses a tamper resistant smart-card that produces a random value hidden from the voter, and [BFP$^+$01] proposed a solution which requires an authority used for randomness (similar to the role of the booth in our solution).

Bare-handed protocols started with the ground-breaking work of Chaum and Neff [Cha04, Nef04]. Many other schemes followed (e.g., [CRS05, Rey05, LTR$^+$06], and the more recent [Cha07, AR06, MN06]). We mention that in many of these schemes there is no privacy towards the booth (and the voter simply tells his vote to the booth), and in many of these schemes privacy towards a malicious ballot creator is lost, see the discussion in the introduction.

We use the [CGS97] scheme (using threshold encryption for tallying) or the [SK95] scheme (using mix networks for tallying) as our underlying schemes. One nice feature of these schemes is that they have two separate phases: one for casting votes and one for tallying, casting a vote ends with a published encrypted vote that can not be opened by unauthorized parties. Also, both schemes use ElGamal encryption (described in Appendix A). The immediate benefit of using ElGamal is its homomorphic property, meaning $E(m_0, r_0) \cdot E(m_1, r_1) = E(m_0 \cdot m_1, r_0 + r_1)$ where $E(m, r)$ is an encryption of m using a randomness r.

4 A Bare-Handed Extension

4.1 An intuitive discussion

Let us summarize the situation so far. Someone has to prepare the encrypted vote. If the

[1] In manual elections there are voting booths that physically isolate the voter for privacy and coercion resistance. The same is true for electronic elections as well. All schemes that we are aware of guarantee privacy and coercion resistance assuming some trust in the system.

[2] A coercer can force the voter to vote randomly and verify his behavior.

voter prepares it at home, then we are susceptible to attacks on receipt freeness (because the voter can open his vote) and coercion-resistance (because the voter can be given the vote by the coercer). If, on the other hand, we ask the booth to encrypt the vote (as in Chaum's and Neff's schemes) we lose privacy.

We could also go a middle way: ask the voter to prepare the encrypted ballot, and then let the booth re-encrypt it. However, in such a case, the voter has to check the booth properly re-encrypts his vote (e.g., to see that the booth is not multiplying his vote with an encryption of a value other than one) and we do not want the voter to do computations at the booth. A simple solution might be to ask the booth to put the re-encryption and the original vote at the public board, and let the auditors check the calculations, but then we are back to revealing the original vote, and the coercion problems.

The key idea behind our solution is very simple. We borrowed it from the way paper-ballot elections are currently carried out. In paper-ballot elections, privacy and coercion resistance are obtained by making sure that the voting booth has paper ballots for each of the candidates. In a similar way, we ask the voter to prepare ballots with valid votes for *all* existing candidates. For reasons we explain shortly, we ask the voter to prepare *two* ballots. We also ask him to give a proof that:

- All the votes he prepared are legal and encode an existing candidate.
- He prepared two ballots for each of the candidates, and he *knows* the correspondence between the votes and the candidates.[3]

These proofs can be prepared in advance.

The booth role is to re-encrypt the ballot's votes (we call this *ballot's re-encryption*), which is necessary for coercion resistance. This, in turn, forces us to check the booth. The voter does this by randomly choosing *for every candidate* one of the two ballot's re-encryptions for testing the booth. The testing itself is done by the auditors using the data that appears in the public board. The voter then uses the other re-encryption of his candidate for the actual voting.

Thus, in the first stage a poll-worker checks the voter can associate votes with candidates, and in the second phase the voter checks the booth properly re-encrypts messages. A coercer may potentially use both stages for coercion. The way we bypass these problems is by forcing both tests to apply to *all* candidates. If you prove you can associate a vote to a candidate you reveal information. But if you do that for all candidates you reveal nothing.

The implementation details are important as (not surprisingly) there are some subtle points hiding. We mention two issues here:

- (Active attacks) The booth may cooperate with another voter A' to reveal A's vote by using the active attack of Pfitzmann [Pfi94].
- (A coercion attack) In the above protocol we assumed the voter is free to choose his random coin. However, a coercer might force the voter, e.g., to use a random coin which is a hash of B's re-encryptions. This, sometimes, enables a coercion attack. Such an attack also applies to [SK95] and [HS00].

Indeed, finding a working scheme requires delicate balancing. We begin with a formal statement of the protocol, followed by an (informal) proof of correctness.

4.2 A formal description of the voting process

Pre-voting : Here is what a voter V does at home. V prepares two ballots. Each ballot is printed on both sides (back and front) and contains records for each of the candidates. We now describe how V prepares such a ballot.

Say there are D candidates. For every $i = 1, \ldots, D$, V picks a random string r_i and prepares an encrypted vote $y_i = E(m_i; r_i)$ for the i'th candidate m_i (where the specifics of this encoding function E depends on the underlying scheme) along with

[3] This is necessary because the coercer might give the voter a set of valid ballots but without telling him which encrypted ballot corresponds to which candidate. We therefore ask the voter to show a poll-worker he can match ballots with candidates.

a NIZKP that y_i encrypts a legal candidate.[4]

On the front side, V prints D rows containing the D values y_i in a random order. On the back side, V prints D rows containing the D tuples (m_i, r_i) using the same random order. Also, on both sides, the voter's name (and a serial number if needed) appears in plain-text. See figure 1.[5]

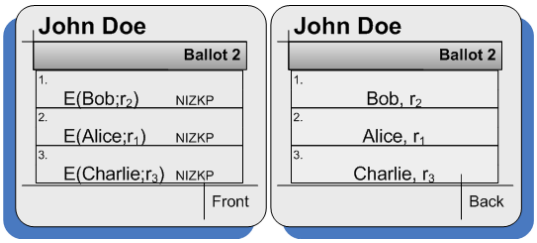

Figure 1: The ballot is printed on both sides. The back side contains (in plain-text) the candidates' names along with the random strings used for encrypting their corresponding votes. The front side contains the encrypted votes $E(m_i; r_i)$ along with a NIZKP that those encrypted values are valid.

Voting (and verification) : V identifies himself with an ID. In front of a poll-worker he shows (using a scanner for instance) the front sides of his two ballots, and this is published together with the voter name on the public board for universal verification. The booth B and the auditors check that the non-interactive, zero knowledge proofs are correct and all the votes on the front sides of the two ballots are legal.

A poll-worker picks a random number $i \in \{1, 2\}$ and publishes i on the public board. The poll-worker asks the voter to scan the back side of the i'th ballot, and it is sent to the public board. The booth and the auditors check that the back side matches the front side (this guarantees that the voter knows how to open his ballots). We denote by P the remaining ballot. The voter now enters the booth.

Casting a vote : Say the voter V wants to vote for the candidate that appears on the c'th row of P, $c \in \{1, .., D\}$. Then, V sends B the number c over the *recordable private channel*. The value c is *not* posted on the public board.

Re-encryption (and verification) : Say the front side of P has the D values $\{e_1, \ldots, e_D\}$. B computes two re-encryptions of the front side of P, i.e., two sets $P^{(0)} = \left\{e_1^{(0)}, \ldots, e_D^{(0)}\right\}$ and $P^{(1)} = \left\{e_1^{(1)}, \ldots, e_D^{(1)}\right\}$, where $e_i^{(0)}$ and $e_i^{(1)}$ are obtained by multiplying e_i with a random encryption $E(1; U)$ of 1. Then the booth picks two random permutations $\pi_0, \pi_1 \in S_D$ and publishes $\pi_0(P^{(0)})$ and $\pi_1(P^{(1)})$ *on the public board*, where $\pi(P)$ is the set P with the D rows of P permuted according to π. The booth also publishes on the public board a NIZKP that all rows in $\pi_0(P^{(0)}) \cup \pi_1(P^{(1)})$ are re-encryptions of some vote given in P. Finally, the booth also tells the voter, over the *recordable private channel*, the values $c_0 = \pi_0(c)$ and $c_1 = \pi_1(c)$.

The voter publishes a bit $b \in \{0, 1\}$ and the booth reveals Π_b on the public board (and if the booth is honest then $\Pi_b = \pi_b$), along with the randomness used to create the re-encryptions in $P^{(b)}$. The *auditors* check correctness and the *voter* checks that $\Pi_b(c) = c_b$, i.e., that the booth's permutation is consistent with the ordering the booth declared to the voter.

Publishing a vote : The booth publishes $c_{\bar{b}}$ over the public board and the vote is taken to be $P_{c_{\bar{b}}}^{(\bar{b})}$. The voter V checks that the published value matches $c_{\bar{b}}$ that was sent to him over the recordable private channel. If everything so far is correct, V and B shred the channel's record (and in particular they shred c) and V leaves the booth.

This completes the voting stage. The tallying stage is done as in the original, underlying

[4] Such non-interactive, zero knowledge proofs are described in Appendix B.

[5] Another subtle point is the following. A coercer might supply the voter with legal ballots whose back side is covered with a scratch area, and tell the voter to vote with a non-scratched ballot. The voter is able to show the back side of the test ballot (by first scratching it) but must keep the other ballot covered, effectively enforcing a random vote [Ano07]. We solve this problem by testing the voter in front of a poll-worker.

scheme.[6] Notice that the voter can pre-compute the votes (and the non-interactive proofs) in the ballots, and can come to vote at the booth bare-handed, carrying only his two ballots of votes.

4.3 Informal proof of correctness

4.3.1 Coercion resistance

We assume a coercer prepared the voter's two ballots and directed him to act in a specific way. We first notice that -

Claim 1 *If one of the paper ballots the voter prepares is not legal the voter is caught with probability close to one. Also, if one of the two ballots the voter prepares does not contain a vote for each candidate, or, if the voter can not match the corresponding back and front parts of a ballot, then the voter is caught with probability close to half.*

One can argue that probability one half is not small enough. However, notice that this means that if a coercer tries to coerce t people, then with high probability (except for probability $2^{-\Omega(t)}$), about $t/2$ of them will be caught, and so with high probability the coercer himself will be detected.[7]

If the voter holds a valid vote for each candidate, and he can associate encryptions with candidates, he can, in particular, vote to any candidate he likes. In the rest of this subsection we show he can not prove to the coercer what choice he had made. After the voter leaves the booth, the private channel transcripts are shredded. An outsider only sees the published information on the public board which contains the voter's selected bit b, the published re-encrypted vote and one set of re-encryptions which is opened in full (and so is independent of the value c). In fact, the third item can be efficiently simulated, and so does not add any information. The first item, the selected bit, can be chosen in any way the coercer directed. The second item, the re-encryption of the actual vote, is an ElGamal re-encryption of one out of k votes and using randomness and keys that the coercer (and the voter) does not have. Thus, this re-encryption is computationally indistinguishable from re-encryption of any of the other votes. In particular, the voter can claim he sent any c and the coercer will accept with the same probability.

4.3.2 The other requirements

The proof of the other requirements is similar to preceding schemes and we omit it, and we only consider the bare-handedness property. We look at the voter's actions in the booth. The voter gives the front sides of the two ballots and the back side of the selected ballot. The voter then picks his vote c by looking at the back side of his remaining ballot and choosing the row number of the candidate he supports. Then the voter selects a random bit. Finally, the voter has to compare two integers (each between 1 and D) for checking the booth. We believe all of this can be done by humans without the help of a computing device.

One comment is in place. We (and most of the other works in the area) assume the existence of NIZKP. Indeed, if OWF exist, and if the parties have access to a source of shared randomness, every language in NP has a NIZKP [FLS90]. However, the NIZKP obtained using such techniques has some large polynomial complexity, and is impractical. Also, one should question this common source of randomness. Another way to go is to use the Fiat-Shamir heuristics [FS86], but then we can not claim anything formal about unforgeability against unbounded adversaries.

[6] We mention that if we take the underlying scheme to be [CGS97] (using threshold encryption) then some of the NIZKP we use already appear in the original scheme and so they should be combined.

[7] Another direction one might be tempted to take, is to ask the voter to come with $J+1$ ballots and to use J of them for verification. The error probability then becomes $\frac{1}{J+1}$, and so goes down only linearly in the number of ballots.

5 A practical version using shredding

In our protocol we require two physical devices: a public board and a recordable private communication channel. These assumptions are not easy to implement. We further modify the protocol, simplifying the interaction between the voter and the booth with the goal of demanding less from the recordable private channel. Our modification is similar to ideas used in Prêt *à* voter scheme [CRS05] and in the recent Scratch and Vote scheme [AR06].

The modification is as follows: The protocol begins as before. The voter prepares two ballots, one is tested, and the remaining ballot P is used for the actual voting. The booth then prepares, as before, two re-encryptions $P^{(0)}$ and $P^{(1)}$ of P. Here we deviate from the previous protocol. The booth prints a ballot with two columns. The j'th row of the ballot consists of $P^{(0)}_j$ in the left column and $P^{(1)}_j$ in the right column (along with a NIZKP that the re-encryption is legal). The order of the rows in the re-encryptions $P^{(0)}$ and $P^{(1)}$ is the same as the order of P. Also, the values in each column are signed by the booth and covered with a scratch surface (see Figure 2). Next, we do the following:

Figure 2: 1. The two column are covered. 2. The voter selects a column b, tears his vote and scratches it. 3. The test column is scratched in front of the poll worker. The vote and the test column are sent to the public board. All the rest is shredded.

- The voter picks $b \in \{0,1\}$ at random, scratches off the row of his candidate from this column (recall, that this is determined by the row of the candidate in the back side of P) and publishes it as his vote.[8]
- The voter surrenders the other unscratched pieces to the poll-worker. He shows the poll-worker that only one piece (that of his candidate) is scratched.[9]
- The remaining pieces of column b are shredded. Also, the voter (or the poll-worker) scratches off the other column. He publishes it and takes it home as a receipt. The booth reveals the randomness used to create the re-encryptions in this column, and the *auditors* check correctness.

One problem that exists with this protocol is that we can not settle disputes. Consider for example the scenario where the auditors discover the information in the scanned column is inconsistent with the data the booth publishes, and the booth claims the voter did not scan the information the booth sent him. The protocol supplies no way of determining whether the voter is honest and the booth is dishonest or vice versa. This problem implicitly appears in all previous protocols (and in particular in [CRS05] and [AR06]) and is a reflection of the fact that the channel that we use is not private, recordable channel. There are several pragmatic suggestions how to solve this problem (e.g., the booth prints its data on a special paper).

The protocol is similar to the one described in Section 4 but what we gain here is simpler interaction. Other than the problem discussed above (which is common to other protocols in the area) the protocol enjoys the same properties as the one in Section 4. We omit the proof for lack of space. Thus, our protocol is as practical as the other protocols in the field, while enjoying true privacy even with respect to the booth.

We remark that many of our computations require long random numbers. As in [AR06], we can reduce the ballot's size by replacing the long random numbers with much shorter random numbers, using these shorter numbers as seeds of a pseudo random generator. Also, bar-codes can be used to encode long strings.

[8] If we assume the communication with the poll-worker is public, then the voter also separates all the rows of his chosen column. He does that in order to hide his chosen row from outsiders.

[9] The reason the voter has to show a poll-worker that all other rows are still covered is to avoid vote-buying. Otherwise, a voter can be paid for voting with an encrypted value that starts, say, with a specific sequence, effectively forcing a random vote.

References

[Ano07] Anonymous reviewers, 2007.

[AR06] Ben Adida and Ronald L. Rivest. Scratch and vote: Self-contained paper-based cryptographic voting. In *Workshop on Privacy in the Electronic Society*, 2006.

[BFP+01] Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard. Practical multi-candidate election system. In *PODC*, pages 274–283, 2001.

[BT94] Josh Cohen Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *STOC*, pages 544–553, 1994.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, pages 174–187, 1994.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *EUROCRYPT*, pages 103–118, 1997.

[Cha04] David Chaum. E-voting: Secret-ballot receipts: True voter-verifiable elections. *j-IEEE-SEC-PRIV*, 2(1):38–47, January/ February 2004.

[Cha07] David Chaum. Punchscan, http://punchscan.org, April, 2007.

[CP92] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *CRYPTO*, pages 89–105, 1992.

[CRS05] David Chaum, Peter Y. A. Ryan, and Steve Schneider. A practical voter-verifiable election scheme. In *ESORICS*, pages 118–139, 2005.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *FOCS*, pages 308–317, 1990.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

[Hir01] Martin Hirt. *Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting*. PhD thesis, ETH Zurich, September 2001. Reprint as vol. 3 of *ETH Series in Information Security and Cryptography*, ISBN 3-89649-747-2, Hartung-Gorre Verlag, Konstanz, 2001.

[HS00] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *EUROCRYPT*, pages 539–556, 2000.

[JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *WPES*, pages 61–70, 2005.

[LTR+06] D. Lundin, H. Treharne, P. Ryan, S. Schneider, J. Heather, and Z. Xia. Tear and destory: chain voting and destruction problems shared by pret a voter and punchscan and a solution using visual encryption. In *Workshop on Frontiers in Electronic Elections (FEE)*, 2006.

[MBC01] Emmanouil Magkos, Mike Burmester, and Vassilios Chrissikopoulos. Receipt-freeness in large-scale elections without untappable channels. In *I3E*, pages 683–694, 2001.

[MN06] Tal Moran and Moni Naor. Receipt-free universally-verifiable voting with everlasting privacy. In Cynthia Dwork (Editor), editor, *CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 373–392. Springer-Verlag, August 2006.

[Nef04] Andrew Neff. Practical high certainty intent verification for encrypted votes. 2004.

[Pfi94] Birgit Pfitzmann. Breaking efficient anonymous channel. In *EUROCRYPT*, pages 332–340, 1994.

[Rey05] D. J. Reynolds. A method for electronic voting with coercion-free receipt. In *Workshop on Frontiers in Electronic Elections (FEE)*, 2005.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - A practical solution to the implementation of a voting booth. In *EUROCRYPT*, pages 393–403, 1995.

A ElGamal encryption

We use ElGamal encryption over a multiplicative group of prime order, as suggested by [Pfi94, SK95]. Specifically, we first publicly choose two large primes q' and q such that $q|q'-1$. Let k be the integer such that $q' = qk + 1$. We also fix a generator g' of $F^*_{q'}$. The cyclic group G we work with is the one generated by $g = (g')^k$ and its order is $\frac{q'-1}{k} = q$.

The public key then includes (other than q', q, g) a value $y \in G$. The private key is $x \in G$ such that $y = g^x$. To encrypt a message $m \in G$ (given the public keys q', q, g, y), we choose uniformly at random $r \in_R [1..q-1]$ and output $E(q', q, g, m, y; r) = (g^r, y^r \cdot m)$. To decrypt a message (α, β) we compute $m = \beta \cdot \alpha^{-x}$.

As we work with global values q', q, g and y shared by all participants, we abbreviate $E(q', q, g, m, y; r)$ to $E(m; r)$. ElGamal is homomorphic, namely $E(m_1; r_1) \cdot E(m_2; r_2) = E(m_1 \cdot m_2; r_1 + r_2)$.

B Reviewing some zero-knowledge proofs

B.1 (Non-interactive) Zero knowledge proofs

Assuming OWF and a shared source of randomness, every problem in NP has a non-interactive proof system. However, these proofs have high (polynomial) complexity [FLS90], and even worse, we do not have a trusted shared source of randomness. In practice, we take a three round interactive proof and convert it to a non-interactive proof using the Fiat-Shamir heuristics [FS86] (changing the challenge to be the hash of the transcript preceding the challenge). We use the next three interactive proofs: equality of discrete logarithms from [CP92], one-out-of-ℓ re-encryption and one-out-of-ℓ message encryption, both from [CGS97, CDS94]. For completeness, we soon describe them. We mention that both the interactive and non-interactive protocols are coercible if the transcripts are public (we demonstrate this soon).

B.2 Zero-knowledge proof of equality of discrete logarithms

Let G be a multiplicative group of order q, and let g_1, g_2 be two (possibly different) generators of G. The input is $v, w \in G$. The prover knows discrete logarithms of v and w, i.e., x_1 and x_2 such that $v = {g_1}^{x_1}, w = {g_2}^{x_2}$, and claims they are the same, $\log_{g_1} v = \log_{g_2} w$.

The following protocol is from [CP92]:

- The prover chooses $z \in [2..q]$ at random and sends $a = g_1^z$, $b = g_2^z$ to Bob.
- The verifier chooses a challenge $c \in [2..q]$ at random and sends it to the prover.
- The prover sends $r = (z + cx)\ (mod\ q)$ to Bob.
- The verifier checks that $g_1^r = av^c$ and $g_2^r = bw^c$.

The protocol is a honest verifier, perfect, statistical zero knowledge, with perfect completeness and $1/q$ soundness error. It is not known to be zero knowledge against dishonest verifiers.

It also, three round public coin, so the proof can be turned non-interactive, by using the Fiat-Shamir heuristic, changing the challenge c with the hash of a, b, v, w.

B.3 A Zero knowledge proof for 1-out-of-ℓ re-encryption

We use the same notation as before - we let G be the multiplicative group as before and fix some $g, h \in G$. Now, the prover wants to prove that for a publicly known pair (x, y) there is an ElGamal re-encryption[10] in the ℓ encrypted pairs $(x_1, y_1), \ldots, (x_\ell, y_\ell)$. We assume the re-encrypted pair is $(x_t, y_t) = (x, y) \cdot E(1; r) = (xg^r, yh^r)$, where r is a random secret value known only to the prover. The protocol is described in Figure 3. It is taken from [CGS97].

Using the Fiat-Shamir heuristics, the protocol can be made non-interactive using the challenge $c = H(a, b, x, y, x_1 \cdots x_L, y_1, \cdots, y_L)$. The prover publishes c, d, r and the verifier is the same as before.

[10] We say a pair (a', b') is a re-encryption of (a, b) if $(a', b') = (a, b) \cdot E(1; r)$ for some value r.

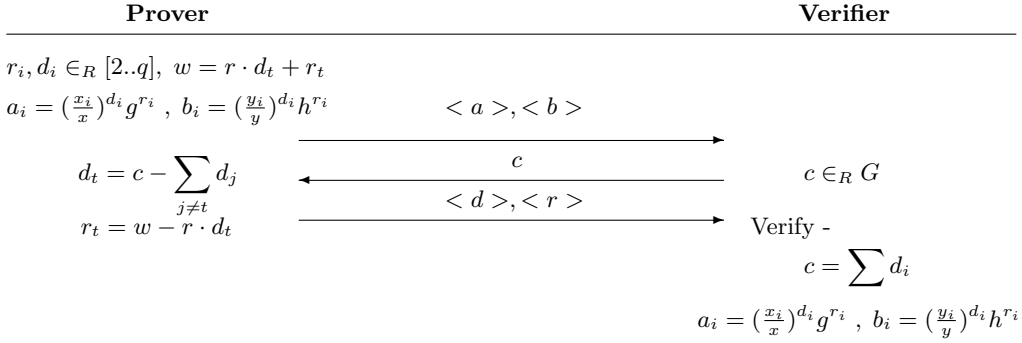

Figure 3: Re-encryption of 1-out-of-ℓ interactive proof. The input is: G, $g, h \in G$, $(x_1, y_1), \ldots, (x_\ell, y_\ell)$ and $(x, y) = (x_t, y_t) \cdot E(1; r)$. a denotes the vector $a = (a_1, \ldots, a_\ell)$ and similarly b, d, r

Re-encryption is a symmetric property (if (a', b') is a re-encryption of (a, b), then (a, b) is a re-encryption of (a', b') [11]). In particular, the above is also a ZKP for the case where we are given (x, y) and we want to prove it is a re-encryption of one out of the ℓ pairs (x_i, y_i).

B.4 A Zero knowledge proof for 1-out-of-ℓ message encryption

We now look at the following problem: we are given ℓ plain-text messages $m_1, \ldots, m_\ell$ and one encryption (x, y) and we want to prove it encrypts one of the ℓ plain-text messages. The protocol for that is given in [CGS97] and is based on the 1-out-of-ℓ re-encryption, and we give it here for completeness.

Given $m_1, \ldots, m_\ell$ and $(x, y) = E(m_t; r)$ (for some t and r known only to the prover), the prover publishes $(x_i, y_i) = (x, y m_i^{-1})$. It is easy to check that $(x_i, y_i) = E(m_t m_i^{-1}; r)$. The prover now proves that one of (x_i, y_i) is a re-encryption of $E(1; 1)$ using the ZKP from previous subsection.

B.5 Coercion in zero-knowledge protocols

We mention that both the interactive and non-interactive protocols are coercible if the transcripts are public. e.g., during the interactive protocol of *Zero-knowledge proof of equality of discrete logarithms* the prover commits to z (using g_1^z). If the transcripts are public, a coercer can coerce the prover to reveal z (which can be done in only one way) and using this he can calculate $x = (r - z)/c$. In the non-interactive protocol this coercion is done using the hash function and z.

[11] This is because $E(1; r)^{-1} = E(1; -r)$.