

Secret Ballot Elections with Unconditional Integrity

David Chaum, Jeroen van de Graaf, Peter Y. A. Ryan, Poorvi L. Vora

Abstract This paper presents a voting scheme that allows voters to verify that their vote is accurately included in the count, whilst maintaining ballot secrecy and coercion resistance. It also presents a rigorously defined set of requirements for secret ballot voting, and proofs that the scheme satisfies these requirements.

Keywords: voting protocol, election protocol, voter-verifiability, unconditional integrity

1 Introduction

Election automation, initially introduced in part to improve the integrity of election results, unfortunately also allows new vulnerabilities. While the integrity of typical commercial transactions – such as bank deposits and withdrawals – may be ensured through the use of comprehensive and unforgeable receipts, the need for ballot secrecy makes such receipts inapplicable to elections. Even by itself, ensuring ballot secrecy is a challenge, as it is directly related to the difficult, and more general, problem of providing secrecy of the endpoints of a communication channel.

In 1979, Chaum [1] proposed a cryptographic technique to allow those trusting their own computer to participate in hiding who communicates with whom, in effect hiding communication channel endpoints. An initial example use of these techniques allowed voters, each using their own computer, to conduct a secret-ballot election in a distributed manner over a computer network. Even if everyone were to have their own computer, however, malware makes it unreasonable for voters today to trust personal computers with their votes.

The present work shows how elections can be conducted in a robust and efficient manner using computers in booths at polling places. A voter can easily check that the vote seen in the booth is in fact correctly included among the encrypted votes posted online. These votes and other encrypted data posted online also allow anyone with a computer to effectively audit that all the encrypted votes posted have been correctly included in the final tally. The resulting level of integrity is said to be unconditional, in that even if those running the election themselves had unlimited computing power and were to have programmed the computers in the booths in an arbitrary manner, still any attempt to manipulate the outcome would be detected with a probability that grows exponentially with the number of ballots tampered with.

This paper presents what we believe is currently the most rigorous and understandable model of requirements for election systems, a valuable contribution in its own right. Then we introduce the proposed election system by a new simplified mechanism that we find rather intuitive. Next this construction is extended in a single step to yield a construction for the most general solution to voting automation known, which allows voters to be sure that whatever they see in the booth will be included correctly in the outcome. Finally we prove that the full extended system satisfies the requirements of the model.

Consider, by way of introduction, how it would be to vote with the full system. You enter a voting booth and make various choices there using a computer interface. A summary detailing your vote is printed and shown you behind glass in the booth. If you believe a mistake has been made, or you wish to change your vote for whatever reason, you can pull down on a lever which sends the printed summary, still in view behind glass, to a shredder below. When you are satisfied with the vote displayed on such a printed summary, however, you simply move the lever either left or right, your free choice. If you move it to the left, a visible

mechanical mechanism feeds the top layer out to you for you to keep and the bottom layer dropped into a shredder below; if you move the lever right, the bottom layer instead is fed out and the top layer shredded. The special printing is clearly readable when the layers are laminated together behind the glass, but once they are separated each is rendered unreadable by unique coding.

Outside the booth, a portable device such as a mobile phone with built-in camera, which might be supplied by you or if not by an election observer, can even check the digital signature revealed on the receipt you keep. This ensures that the signature correctly signs all that is printed on the receipt and can be used to hold those running the election accountable. You can of course provide the receipt digitally or physically to a friend, political party, other organization or the media. If you have a computer at home, though, you can enter the receipt's serial number into a web browser and check that the information displayed matches the information in coded form on the receipt. If you wish, you can also use your computer to verify that the election is being conducted properly by simply running (or even writing yourself) software that checks what is displayed on the official election website. Such software would catch any significant cheating apart from the already-checked posting of the receipts. If any such software discovers malfeasance, the previously published digital signatures would prove malfeasance.

2 Related Work

Chaum [1] is credited with the first description of a cryptographic technique for electronic voting. Its main contribution to the voting literature has been the concept of a mix: an entity that receives several messages encrypted with its public key, decrypts the messages in a single batch, and permutes them, thus destroying links between input and output messages. Another major contribution to the voting literature was the use of homomorphic encryption schemes by Cohen and Fisher [2, 3]. Votes are never decrypted before counting, and properties of the encryption scheme allow the votes to be counted while in encrypted form. Benaloh and Tuinstra [4] describe a scheme that allows a voter to verify that her vote was correctly cast, without enabling her to provide reliable information on her vote to a third party.

In a recent important development, Neff [5] describes an efficient practical protocol for verifiable mixing, and its application to universally verifiable voting. Jakobsson, Juels and Rivest [6] provide a randomized partial audit of mixes which ensures that the probability that cheating mixes (those that remove, replace or misprocess votes) are caught is high. Several other voting methods based on mixes or homomorphic encryption have been proposed [7, 8, 9, 10, 11]. There are also several surveys on voting [12, 13, 14, 15, 16, 17, 18].

Until very recently, however, there has been no voting method, electronic or otherwise, that reassured the voter that her vote was correctly added to the tally, without requiring her to trust an entity while voting: whether a voting machine, or a polling official. Given that polling machines can be particularly untrustworthy [19], this has been very limiting. This paper describes a method, sketched by Chaum in [20], in which a voter can convince herself that efforts to change the tally are detected with very high probability, without having to trust any entity at the polling booth. Additionally, this is an information-theoretic property of the method, and does not depend on any problems being computationally intractable. Other voting protocols with similar properties have recently been proposed [21, 22, 23, 24]. However, unlike these, this paper addresses write-in ballots, and either provides more voter privacy, or provides higher efficiency audits. A technical description of [20] appears in [25].

3 Security requirements for an election

In this section we formulate a simple and concise list of requirements, which define what it means for an election to be fair. We are not aware of a similar list in the literature. Although many papers briefly mention requirements (see [26]), most do not discuss them in detail.

We create a concise list of well-formulated requirements for honest elections as follows. We first describe very

briefly a conventional election using paper ballots, and from this process we deduce a list of requirements that needs to be fulfilled. We do this following the chronological order of a conventional election, which simplifies the process of verifying that no requirement has been forgotten.

3.1 An election with paper ballots

A conventional election goes through the following steps:

1. Establish a list with the names of all legitimate voters.
2. Before starting the election, all present observe that the ballot box is empty.
3. A legitimate voter (who has not already voted) receives a ballot, enters the voting booth, and fills in her preference.
4. The voter verifies the preference filled on the ballot.
5. The voter deposits her ballot in the ballot box. (From that moment on, the vote is cast, and she cannot undo or modify her vote.)
6. When the time for voting has expired, all present observe the opening of the ballot box and tally of the contained ballots.
7. Those who disagree with the count can request a recount. The votes are recounted (maybe more than once), under the observation of all present, until there is consensus.

We believe this election is typical of elections and voting procedures in many countries, be it for confidential elections in the chambers of legislative bodies, or for public elections at various levels of government. Differences occur in such things as graphic ballot lay-out and the way the voters mark choices (a cross in a square, connecting an arrow, filling a circle, writing a number, writing a name, etc). These differences for the most part do not essentially alter the security requirements. Other seemingly innocuous differences, however, can impact upon the properties and may be called "security relevant". For instance, writing the name can have an impact on the number of rejected votes, and can facilitate establishing a link (on purpose or not) between a voter and her ballot. Another example is whether multiple contests are marked on the same ballot or on different ballots, which can affect the ease with which vote buying or coercion may occur.

Apart from the "security relevant" differences just listed, we believe that most elections have the same security requirements. In the next section we discuss these informally with the intention of defining a small set of such requirements which we believe are universal, and which we will use as a yardstick for our proposed protocol. The reader may safely skip to section 3.3, which provides a brief formal list of requirements and return to the detailed development of the ideas later if desired.

3.2 Informal discussion of security requirements

The following subsections discuss the requirements that an election must satisfy.

3.2.1 About who may vote

(ONLY VALID VOTERS) *Only legitimate voters, called voters for short, can create a ballot and deposit it in the ballot box.*

Explanation: Any election has a finite set of persons who have the right to participate in that election and only these persons are allowed access to the voting process.

(ONE MAN ONE VOTE) *A voter can cast at most one vote.*

Explanation: The same person voting twice is not allowed.

3.2.2 About the act of voting — the creation and casting of ballots

(BALLOT SECURITY) *Filling in the ballot and putting it in the ballot box is a confidential act, and under no circumstance, not even with the connivance of the voter, should it be possible to deduce for whom or for what the voter cast or did not cast votes.*

Explanation: It is important to realize that this requirement has two sides. First, the voter should have the freedom to express her will without the risk of repercussion. To guarantee this, nobody should be able to discover for whom or what she voted or did not vote.

Second, it is necessary to prevent so-called “improper influence” of voters, which includes the buying and selling of votes. Consequently it should not be possible, *even with the cooperation or connivance of the voter*, to deduce the vote. For this reason it is of the utmost importance that, during the marking and casting of the ballot, no proof or receipt is created which could be linked to a vote inside the ballot box, since this would permit coercion and the buying and selling of votes.

In order to guarantee the secrecy of the ballot, there exists a private space, the voting booth, where the voter can fill the ballot. This requirement could be reformulated by stating that the only information that may leave the voting booth is the vote as filled in on the ballot, but nothing else.

(VERIFICATION OF BALLOTS) *The voter can verify her vote, that she created a valid vote, and can revise her vote before committing.*

Explanation: After having created her vote but before actually casting (relinquishing) it, the voter should have the right to check her vote is recorded as intended and that it is valid. She should have the opportunity to correct or revise her ballot.

(BALLOT WILL BE TALLIED) *The voter can convince herself that her vote is included in the set of votes tallied.*

Explanation: This requirement is the hardest to realize. We would like to be able to hand a proof/receipt to the voter, allowing her to verify that her vote is among the set of votes tallied. However, conventional wisdom has it that this requirement contradicts the more important requirement of ballot secrecy. In the case of paper ballots, this requirement is theoretically achieved in the following way: after the voter has cast her vote by depositing the ballot in the ballot box, she waits until the closing of the election, and when the ballot box is opened for the tallying of the votes, she is sure that her ballot is among the set, even if she does not know which particular ballot corresponds to the one she filled in. It is interesting to note that, essentially, the voter’s faith is based on the common sense notion that an object put in some place stays there and won’t disappear by itself.

There is a tension between the requirement of verifiability (and auditability, see below) on the one hand, and the secrecy of the ballot on the other, which makes the design of election systems satisfying both requirements without using ballots or other physical objects extremely tricky. Implicit in each election is a sub-procedure that shuffles votes, to protect the anonymity of the voter, but destroys the link between the ballot and the person. This procedure, trivial when dealing with physical objects such as ballots or playing cards, is difficult to simulate in the virtual world. The problem is made more difficult by the verifiability requirement: it should be possible to ensure that the virtual shuffle did not add, subtract or replace any of the shuffled items.

This paper provides a solution to the verifiability problem, while not requiring the voter to trust any other entity. In particular, it provides a solution that addresses the two contradictory requirements of ballot secrecy and ballot verifiability.

3.2.3 About the integrity of the vote — from casting through counting

(INTEGRITY OF BALLOT AND BALLOT BOX) *It should not be possible for anyone to modify a ballot, or remove it from the ballot box, nor should it be possible to add ballots not coming from legitimate voters.*

Explanation: The votes represent the (anonymous) will of the voters (at least at the moment of casting), and any modification would alter that will.

This requirement explains for instance why the ballot box is shown to be empty before starting the election, why the ballot box should remain in a publicly visible place, and also why transparent ballot boxes sometimes are used.

(SECRECY UNTIL THE END) *All votes remain secret until the end of the voting session.*

Explanation: First, revealing partial results early would violate the secrecy of the ballot for those who voted already. Second, knowing the partial result might influence the vote of someone who votes later. Moreover, exclusive access to this information during the voting period could provide advantage in terms of allocation of electioneering resources or even trigger disruption of the voting process.

3.2.4 About the tallying of the votes

(COUNTING IS PUBLIC) *The tallying of votes happens in a public and verifiable way.*

Explanation: For higher credibility of the result it is important that party representatives and neutral observers are present and able to verify the process.

(CORRECTNESS OF THE COUNT) *All valid ballots encountered in the ballot box, and only those, will be included in the count.*

Explanation: Votes not written on a proper ballot, for example, should not be counted as they could represent multiple votes from a single voter. Additionally, votes that are ambiguous ought not to be counted.

(RIGHT TO AUDIT) *It should be possible to audit the tally.*

Explanation: In conventional paper elections, any person can contest the result and request a recount of the votes, which also happens in a public session. In principle this process should converge to a result with which everybody agrees.

In practise things are more complicated: sore losers could prefer requesting recount after recount instead of admitting defeat. Some rules are usually implemented to limit this effect. Another problem is that manual counting is notoriously unreliable: it is not uncommon that a third recount gives a third value, instead of confirming one of the first two, so no convergence takes place.

In fact, recounting is not the crucial property. One could imagine that a counting session is completely video-taped in a way that any disputes can be resolved through the tape by viewing it again, not by another recount. So what is at stake is the requirement of auditability, not recounting, which is only one possible implementation, especially in a manual system.

It is interesting to observe that all these requirements can be stripped down to the following two requirements:

A Filling in the ballot shall be a confidential act (as described above).

B Everything that does not contradict A shall be transparent.

This may, however, be very succinct, probably too succinct to be useful in practice. But it is nevertheless an interesting observation since it points out what is essential from a security point of view.

It is appropriate also to note that full transparency often conflicts with privacy. For example, the list of those who voted could be made public, in order to ensure that the ballot box is not stuffed with votes cast on behalf of those who did not vote. However, this limits the privacy of the authorized voter, who might wish to keep private the fact of her having voted or not. Furthermore, in some countries, like Brazil, voting is obligatory, and if it is known that a particular voter did not vote, there could be repercussions.

3.3 Formal list of security requirements

We condense the previous list into 8 requirements, which are what will be used in the remainder of this paper.

REQUIREMENT A (ONLY VALID VOTERS) *Only persons on the valid voter list, called voters, can create a ballot and deposit it in the ballot box.*

REQUIREMENT B (ONE MAN ONE VOTE) *A voter can cast at most one vote.*

REQUIREMENT C (PRIVACY INDIVIDUAL BALLOT) *Filling in the ballot and putting it in the ballot box is a confidential act, and under no circumstance, not even with the connivance of the voter, should an outsider be able to deduce for whom or for what the voter casted her vote.*

REQUIREMENT D (INDIVIDUAL VERIFIABILITY) *The voter can verify that she created a valid vote, can revise her vote before casting it, and can convince herself that her vote is included in the set of votes tallied.*

REQUIREMENT E (BALLOT BOX SECURITY) *During the voting session it should be possible neither to see the vote on any ballot deposited in the ballot box, modify a ballot, remove a ballot from the ballot box, nor add ballots not coming from voters.*

REQUIREMENT F (COUNT INTEGRITY) *All valid votes encountered in the ballot box, and only those, will be included in the count.*

REQUIREMENT G (PUBLIC VERIFIABILITY) *The tallying of the votes happens in a manner such that anyone can verify that Requirements C, E and F hold.*

REQUIREMENT H (ROBUSTNESS) *The process has a very high probability to succeed.*

The full protocol presented in this paper, section 7 does not address requirements A and B. It satisfies, to an extent made more precise in section 8, requirements C-G.

4 One dimensional protocol

In this section we present a simplified version of the full protocol based on work presented in [22]. In place of the visual cryptography we encode the vote as a pair of aligned strips. This encoding means that the voter can only choose amongst a predefined set of options and so write-ins are not possible. However, in other respects it is equivalent to the full protocol and is useful pedagogically as it avoids much of the complexity of the visual cryptography. We present the full protocol in section 7.

We consider a simple voting process in which each voter is invited to vote for one of n options (one of which might be a “spoilt ballot” option). Before we do so, we first establish some notation.

- We denote a public key by a capital letter, e.g. A , and its corresponding private key by \hat{A} , and the pair by $\langle A, \hat{A} \rangle$.
- Encrypting or verifying an arbitrary message M using a public key A is denoted as $\{M\}_A$, whereas decrypting or signing with the private key is denoted as $\{M\}_{\hat{A}}$.
- Suppose that there are k Tellers, each one running two mixes. Therefore Teller i has two public keys T_{2i-1} and T_{2i} with corresponding private keys \hat{T}_{2i-1} and \hat{T}_{2i} .
- Each booth has three public-key/private-key pairs: $\langle U, \hat{U} \rangle$ used for generating the pseudo-random sequence embedded in the upper layer, $\langle L, \hat{L} \rangle$ used for the lower layer, and $\langle Z, \hat{Z} \rangle$ used for signing the whole receipt. For simplicity of the presentation, we will omit the details of the application and checking of the digital signature. Full details will be given when we come to describe the full protocol later.
- The letters U and L are (also) used as superscripts, the variable a is used as an index on $\{U, L\}$, meaning a specific one of the two layers. The other, complement layer is denoted by \bar{a} .

4.1 Walkthrough of casting a vote

Our voter Anne first registers at the polling station and enters a booth to initiate the vote casting process. When she registers, she might, for example, be provided with a smart card or one-time-password to enable her to cast exactly one vote.

The booth generates a **unique serial number** q associated with this voting session. Suppose for simplicity that we are dealing with a single race with n candidates. The booth generates two n -bit long pseudo-random strings, which are related in a deterministic manner to q through the booth's private keys. We will describe their construction shortly. One is destined for the upper strip, the other for the lower strip.

For concreteness, let us assume that there are $n = 6$ options offered to the voters. In this case the booth device generates two 6-bit pseudo-random strings. Suppose that the upper string w^U is:

0, 0, 1, 0, 1, 1

and the lower, w^L :

0, 1, 1, 1, 0, 0

The booth now inserts these strings into a table as follows:

1	2	3	4	5	6
0	0	1	0	1	1
0	1	1	1	0	0

We will refer to these bits as the **encryption bits**.

The booth now inserts **information bits** into the spaces to form an initial, blank ballot: bits are inserted in the spaces in the upper strip that match the encryption bits immediately below on the lower strip. Similarly for the information bits on the lower strip:

1	2	3	4	5	6
0 0	0 1	1 1	0 1	1 0	1 0
0 0	0 1	1 1	0 1	1 0	1 0

The booth is now ready to invite Anne to cast her vote. Suppose that she chooses “4”. The booth inverts the information bits in the “4” cell, thus we obtain:

1	2	3	4	5	6
0 0	0 1	1 1	0 0	1 0	1 0
0 0	0 1	1 1	1 1	1 0	1 0

Note that in the “4” cell, the upper and lower symbols are each others complement, for all the other cells the upper symbols continue to match the lower.

The booth now prints a graphical representation of this onto the strips, along with the serial number q :

1	2	3	4	5	6	q
♥ ♥	♥ ♠	♠ ♠	♥ ♥	♠ ♥	♠ ♥	q
♥ ♥	♥ ♠	♠ ♠	♠ ♠	♠ ♥	♠ ♥	q
1	2	3	4	5	6	

Where 0 is replaced by ♥ and 1 by ♠.

Assuming that Anne is happy that her choice is accurately encoded and the q values match, she presses a continue button. The booth now prints some further encrypted commitments \mathcal{E}^U and \mathcal{E}^L to both strips.

These are the **envelopes** that contain the information that allows the tellers to reconstruct the appropriate encryption bits and hence extract the original vote. This will be described in detail shortly. Anne should check that the \mathcal{E}^U and \mathcal{E}^L given on the top strip exactly match those given on the lower strip, e.g.:

1 ♥ ♥	2 ♥ ♠	3 ♠ ♠	4 ♥ ♥	5 ♠ ♥	6 ♠ ♥	$q, \mathcal{E}^U, \mathcal{E}^L$
♥ ♥	♥ ♠	♠ ♠	♠ ♠	♠ ♥	♠ ♥	$q, \mathcal{E}^U, \mathcal{E}^L$
1	2	3	4	5	6	

Assuming that these do indeed match, Anne again presses the continue button, at which point the booth offers her the choice between the upper and lower strips. Suppose that she chooses the upper. The booth prints the **seed** for the upper strip, s^U , on both strips, as well as RETAIN on the upper strip and DESTROY on the lower strip:

1 ♥ ♥	2 ♥ ♠	3 ♠ ♠	4 ♥ ♥	5 ♠ ♥	6 ♠ ♥	$q, \mathcal{E}^U, \mathcal{E}^L, s^U, \text{RETAIN}$
♥ ♥	♥ ♠	♠ ♠	♠ ♠	♠ ♥	♠ ♥	$q, \mathcal{E}^U, \mathcal{E}^L, s^U, \text{DESTROY}$
1	2	3	4	5	6	

The seed s^U will be used later to check encrypted commitment \mathcal{E}^U , as described in section 4.3.

Anne now detaches the printout from the printer, and separates the two strips along the middle:

1 ♥ ♥	2 ♥ ♠	3 ♠ ♠	4 ♥ ♥	5 ♠ ♥	6 ♠ ♥	$q, \mathcal{E}^U, \mathcal{E}^L, s^U, \text{RETAIN}$
♥ ♥	♥ ♠	♠ ♠	♠ ♠	♠ ♥	♠ ♥	$q, \mathcal{E}^U, \mathcal{E}^L, s^U, \text{DESTROY}$
1	2	3	4	5	6	

On leaving the booth she should present the strip marked “Destroy” to an official who destroys it in front of her (this is to ensure she does not take it with her, as then she could use it to prove how she voted). She is now left with a ballot receipt of the form:

1 ♥ ♥	2 ♥ ♠	3 ♠ ♠	4 ♥ ♥	5 ♠ ♥	6 ♠ ♥	$q, \mathcal{E}^U, \mathcal{E}^L, s^U, \text{RETAIN}$
----------	----------	----------	----------	----------	----------	---

This strip will be used by the Tellers to reconstruct the vote. In particular, the Tellers will reconstruct the vote by constructing the encryption bits in the destroyed lower layer, using the information provided in \mathcal{E}^L . Note that knowing the encryption string on the retained strip reveals nothing about the vote in the absence of the information bits on the destroyed strip, provided that some reasonable cryptographic assumptions hold (see Section 6).

After leaving the booth, she can present her receipt to one or more “bar-code” readers to be checked for well-formedness. These readers are able to reconstruct the encryption string and the corresponding envelope (\mathcal{E}^U in this case) from the revealed seed, s^U , as described in section 4.3. These may be compared with the corresponding values printed on the receipt, allowing Anne to be confident that (a) the booth provided the correct envelope \mathcal{E}^U , and (b) it correctly generated the encryption string. That is, if the encryption bits had to be generated to decrypt the lower layer, the decryption would have been correct. As the device was required to commit to these values before it knew which strip would be checked, Anne should feel confident that the device did not attempt to cheat by incorrect construction of the receipt. This is the *cut-and-choose* element of the protocol.

4.2 Decrypting a Vote

Once the election has closed, the booth passes the image of each receipt to a web site to be publicly posted. Once the phase of voters checking that their receipts are accurately posted is (successfully) completed, the **tellers** take over and perform a **Chaum anonymizing mix**: progressively transforming and shuffling the terms until finally the decrypted votes are output. The outputs of each mix stage are posted to the web site to enable auditing later. To understand this in detail we need to examine more carefully the construction of the encryption strings and the envelopes.

4.2.1 Construction of the encryption strings

The Booth has two independent public key-private key pairs, $\langle U, \hat{U} \rangle$ and $\langle L, \hat{L} \rangle$ used in the construction of the upper and lower strips respectively. The two private keys are used to sign the serial number q , generating the two seeds

$$\begin{aligned} s^U &:= \{q\}_{\hat{U}} \\ s^L &:= \{q\}_{\hat{L}} \end{aligned}$$

The encryption strings will each be formed by generating $2k$ strings of length n and *xoring* these together. For concreteness we will describe the construction of the upper string, the lower is entirely analogous.

From the upper seed s^U , the values e_1^U, \dots, e_{2k}^U are generated as follows:

$$e_i^U := g(s^U, i), \quad \text{for } i = 1, \dots, 2k.$$

where g is a cryptographic hash function.

Now the e_i^U are input into another hash function, h , to yield the terms that will form the final (upper) encryption string:

$$d_i^U := h(e_i^U) \quad \text{for } i = 1, 2, \dots, 2k.$$

This second hash function h should yield strings of length n (or we might simply truncate the output of an off-the-shelf hash function). Finally the upper encryption string is formed by computing the bitwise *xor* (denoted as \oplus) of these $2k$ strings:

$$w^U := d_1^U \oplus d_2^U \oplus d_3^U \oplus \dots \oplus d_{2k}^U$$

4.2.2 Construction of the envelopes

In order that the tellers may reconstruct the encryption bits of the discarded layer, the e_i values are encrypted, in nested form, with the public keys of the tellers. Since the d_i are all images under h of the e_i , collectively the tellers are able to reconstruct the encryption bits w on the discarded strip.

Let us describe the construction of the envelopes more precisely. For concreteness, we will describe the construction of just one of the envelopes. In our example this will be the one for the lower strip, \mathcal{E}^L , as this

one will be used by the tellers to reconstruct the lower encryption string and so interpret the information bits on the retained, upper strip.

The outer envelope, \mathcal{E}^L , is constructed in a nested fashion: each envelop is the encryption of one of the e bit strings concatenated with an inner envelope. The booth first forms the innermost envelope \mathcal{E}_1^L by encrypting e_1^L under the public key T_1 . For technical reasons, associated with the auditing of the final mix, we need to include a zeroth envelope \mathcal{E}_0^L , which might be constructed as

$$\mathcal{E}_0^L := g(s^U, 0)$$

The first, innermost envelope is now formed as:

$$\mathcal{E}_1^L := \{e_1^L, \mathcal{E}_0^L\}_{T_1}$$

The next envelope is formed by concatenating e_2^L with the innermost envelope \mathcal{E}_1^L and encrypting both with the second public key, T_2 . The i^{th} envelope is given by:

$$\mathcal{E}_i^L := \{e_i^L, \mathcal{E}_{i-1}^L\}_{T_i}$$

Thus:

$$\mathcal{E}^L := \mathcal{E}_{2k}^L := \{e_{2k}^L, \{e_{2k-1}^L, \{e_{2k-2}^L, \dots, \{e_1^L, \mathcal{E}_0^L\}_{T_1} \dots\}_{T_{2k-2}}\}_{T_{2k-1}}\}_{T_{2k}}$$

For technical reasons that are explained when we come to describe the teller auditing process in section 7, each teller performs two mixes. These will be arranged in such a way as to ensure that the public keys are applied in the reverse order to the sequence in which they are used to construct the envelopes. Observe that as a consequence mixes and tellers are numbered **backwards**: the mix that will start by taking away the outermost envelope uses private key \widehat{T}_{2k} , and the teller that operates this mix (and mix $2k - 1$) is numbered k . And the teller that completes the mixing process is the teller numbered 1.

Note that the construction of the receipts sketched above is completely deterministic given the value of q : the encryption strings w^U and w^L are fixed as a function of the signature algorithm and the private keys \widehat{U} and \widehat{L} , and of the two hash functions g and h . It is this fact that enables the bar code readers to verify the correct construction of the encryption strings on a retained strip. It also serves to thwart kleptographic channel attacks, [27, 28].

4.2.3 The role of the tellers

Once all the “raw” receipts have been posted to the web site, a simple transformation is performed on them before they are passed to the first teller. The receipts carry a lot of information that is extraneous to extracting the vote but was needed in order to audit the well-formedness. All of this extraneous information needs to be stripped off before the tellers start their processes. Retaining this information, besides being wasteful, would allow tracking of ballot terms through the mixes.

Firstly each receipt string will be $2n$ symbols and comprises an alternation of (a symbolic representation of) encryption and information bits, the slice depending on whether the receipt was derived from the upper or lower strip. Thus all the receipt strings are transformed into n long strings comprising only information bits.

The serial number, envelope for the retained strip, seed value and the RETAIN text should also be discarded at this stage. Finally, the cell numbers should also be discarded and the symbols should revert to the bit representation for ease of manipulation. In short, only the information bits and the envelope for the destroyed layer should be retained and entered into the mixes.

Thus, in our example, Anne’s ballot receipt would go from:



to:

$$0 \ 0 \ 1 \ 0 \ 1 \ 1, \ \mathcal{E}^L =: (\mathcal{I}^U, \mathcal{E}^L)$$

i.e. the information bits \mathcal{I}^U of the upper strip along with the envelope \mathcal{E}^L which will enable the tellers to reconstruct the corresponding encryption bits of the lower strip and so extract the vote. Henceforth we will drop the U and L superscripts, and refer to $(\mathcal{I}, \mathcal{E})$ as a **pair**.

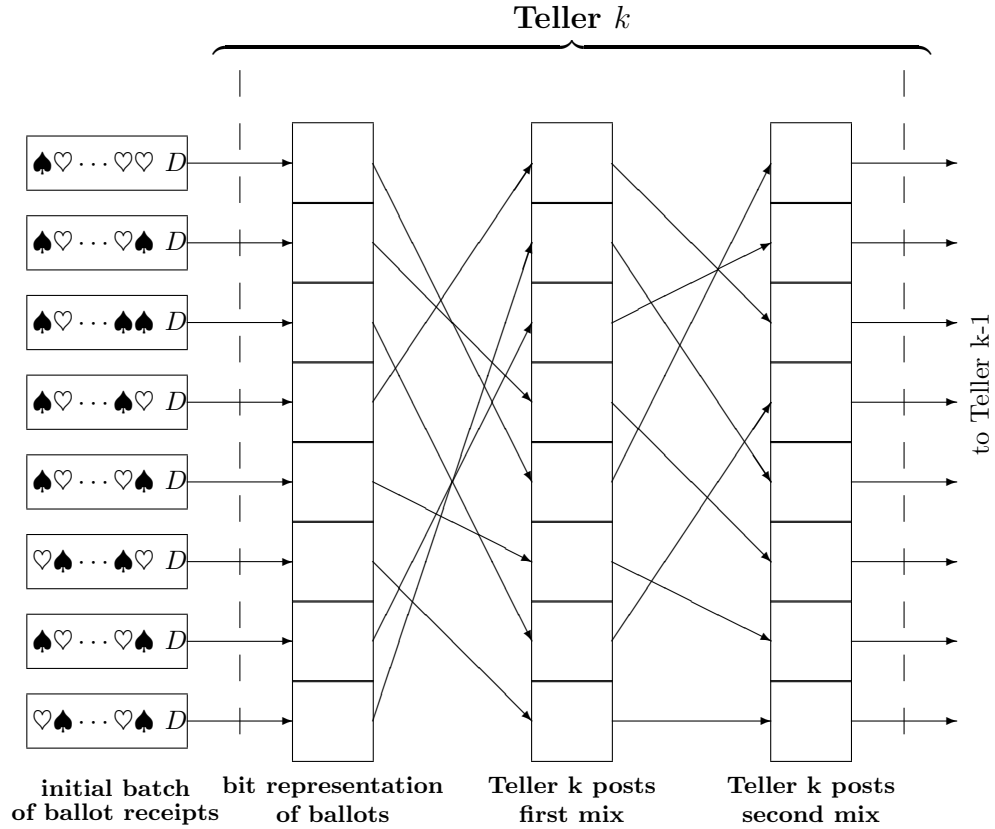


Figure 1: Teller k : permutations

Teller k now collects the resulting pairs $\langle \mathcal{I}, \mathcal{E} \rangle$ and performs its anonymizing mixes, as depicted in Fig 1. Note that the envelopes are unique to each receipt.

For each receipt, Teller k performs its first mix (shuffle) as follows:

- applies its first secret key \widehat{T}_{2k} to the outer envelope $\mathcal{E}_{2k} = \mathcal{E}$ to decrypt the values e_{2k} and the enclosed envelope \mathcal{E}_{2k-1} .
- computes $d_{2k} := h(e_{2k})$ and *xors* this with the receipt string \mathcal{I}_{2k} to form \mathcal{I}_{2k-1} .
- applies a secret permutation to the resulting batch of $\langle \mathcal{I}_{2k-1}, \mathcal{E}_{2k-1} \rangle$ pairs.
- posts the resulting, shuffled pairs to the web page.

Thus each pair is transformed as:

$$\langle \mathcal{I}_{2k}, \mathcal{E}_{2k} \rangle \rightarrow \langle \mathcal{I}_{2k} \oplus d_{2k}, \mathcal{E}_{2k-1} \rangle$$

Teller k now repeats essentially the same kind of process on the resulting batch of terms using its second secret key, i.e., it applies its second secret key to the envelopes, transforms the terms as previously using the e values extracted by the decryption and posts the resulting pairs in a secret, shuffled order. The shuffling of the second mix is entirely independent of the first. Note that the booth encrypted the outer two envelopes using the teller's two public keys, so that Teller k is able to strip both of these layers off.

Teller $k - 1$ now takes the $\langle \mathcal{I}_{2k-2}, \mathcal{E}_{2k-2} \rangle$ pairs posted by Teller k and performs its two Mixes in the same fashion.

Finally, when Teller 1 performs its mixes, this yields a set of receipt strings that show the votes in the clear. More precisely, a vote for candidate v will be revealed as a 1 in the v^{th} position and 0's in all the other positions. In the case of Anne's vote ("4") we get:

0 0 0 1 0 0

To see this, recall the construction of the encryption bits and information bits. The encryption bits are formed by *xoring* the strings d_1, \dots, d_{2k} together. The corresponding information bits were then chosen to complement in the position corresponding to the choice of candidate and to match in all other positions. The result of the tellers' operations on the receipts is to *xor* all the d_i into the information string. This is equivalent to *xoring* the encryption string into the information bit string. By construction of the receipt, the only position in which the information and encryption bits are complements is the chosen position. Hence we get a 1 in the chosen position and zeros elsewhere, so yielding the canonical representation of the vote.

The overall effect then is to have posted on the web site, in the left hand column say, the batch of initial receipts. The next column along shows the stripped down terms: just the information bits of the retained strip and the envelope for the destroyed strip. No shuffling is performed between these two initial columns so allowing easy verification of the correctness of this step. This column is then followed by a set of columns showing the outputs at each mix stage. Finally, in the far right hand column we will have the fully decrypted ballot images. Aside from the first two columns, each column will be a secret permutation of the previous one. Each of these permutations will be chosen at random and independently. Note that the transformation of the ballot terms in each mix prevents the permutation being reconstructed by simple matching of elements.

Assuming that all the tellers perform their transformations correctly, there will be a one-to-one correspondence between the elements of each column and the next. The exact correspondence – which receipt is the decryption of which receipt in the previous column – will be hidden and known only to the teller who performed the transformation between those columns. Thus, the receipts will have undergone multiple, secret shuffles between the first column as posted by the booth and the final decrypted column, showing the votes in clear. This ensures that no voter can be linked to her vote, so ensuring ballot secrecy.

The fact that several tellers are used gives several layers of defence with respect to ballot secrecy: even if several of the tellers, but not all, are compromised, the linkage of voters with their votes will remain secret. The extent of voter privacy provided is described more precisely and formally in section 8.

The decrypted votes will all be available in the final column output of the last teller and so the overall count can be verified by anyone.

The description so far has assumed that all the players, the booth and the tellers, have behaved correctly, in accordance with the rules of the scheme. If everyone obeys the rules we can be sure that the election will be both accurate and private. However, should the booth or any of the tellers cheat, then the accuracy and privacy could be undermined. The design philosophy behind the scheme is to avoid having to place such trust in the components of the scheme.

4.3 Checking on the booth

Anne should perform two checks that serve to detect attempts by the booth to cheat. She should run her receipt through a reader device that checks that the receipt has been correctly formed by the booth. Such devices should be readily available at the voting station for example and provided by independent organizations, such as the Electoral Reform Society in the United Kingdom or the League of Women Voters in the US. These devices, given the information on the receipt, can reconstruct the encryption string, w^U in our example, and the corresponding envelope, \mathcal{E}^U , from the seed value that will have been revealed for the retained strip. Thus, had the booth tried to corrupt Anne's vote by altering these values, it will be detected.

The computations performed by the checker are essentially the same as those performed by the booth device in computing the values on the retained strip:

First, apply the booth's upper public key U to the revealed seed and check that this yields the correct q :

$$q = \{s^U\}_U = \{\{q\}_{\hat{U}}\}_U$$

The e_i values can be computed from s^U , as previously using the g hash function. From the e_i values, the envelope for the retained strip, \mathcal{E}^U in our example, can be reconstructed using the public keys of the tellers, T_i . The reconstructed value for the envelope can be checked against the value printed on the retained ballot strip.

Now the d_i can be computed by applying $h(\cdot)$ to the e_i computed earlier. These can then be *xored* together to give the encryption string, w^U , also available on the retained strip. In practice, the protocol would also require the booth device to apply a digital signature to the receipt which would also be checked at this point. We have omitted the details here to keep the description simple.

These checks ensure that if a malicious booth shows the voter an encoding of her vote, then creates a false envelope which will produce a different decryption by the tellers, it has a one in two chance of being caught out.

Note that the algorithms for these checks are publicly known, so in principle, anyone could construct such a checker and make it freely available. Similarly anyone would be able to examine such a checker to establish that it was performing correctly. Indeed, if she is really enthusiastic, Anne may choose to run several such independent checks.

Once all the receipt batches have been posted to the web site, Anne should also check that her receipt is accurately recorded there. Note further that anyone can subsequently check the validity of the receipts posted to the web site.

4.4 Checking on the Tellers

Checks must also be performed to ensure that the tellers perform all their transformations accurately, i.e. they do not alter, remove, inject or corrupt votes as these move through the mixes.

Recall that each teller performs two decryptions on each pair, and therefore each teller performs two permutations on each batch. They post the outputs of each of these mixes to the web site.

However, if no further information about the process were revealed, tellers could alter votes without fear of detection. Most obviously, the last teller could alter some of the final decrypted votes in favor of a particular candidate. The other tellers could also alter votes but, unless they are able somehow to determine what vote is encoded, they will not be able to produce such a predictable result. In fact random alteration of bits is likely to result in detectable errors in the final decryption. In any event, we want to ensure that we can detect corruption of votes during the mixes.

The solution we adopt in our description of the simplified protocol is based on the approach proposed by Jakobsson et al [6]. In the full protocol described in Section 7, we use a variation of the approaches described

in [6]. Tellers are required to reveal a proportion of the links along with associated secret values to allow the checking of the correctness of the transformations on these selected links: enough to make the probability of significant corruption going undetected vanishingly small, but not enough to allow any of the final decrypted votes to be traced back to the original votes cast. In effect we perform random audits on the actions of the tellers.

The selection process is best described as follows. Recall that each teller performs two mixes and so, on the website, there will be three columns associated with each teller: an input column, an intermediate column (showing the output of the first mix), and a final column showing the output of the second mix. This last column of course forms the input column for the next teller, aside from the last teller, for whom the final column shows the decrypted votes. Each teller can be audited quite independently: for each teller, the auditing authority goes down the intermediate column for that teller and for each term makes a random binary, *left/right* choice. If the choice is *left*, the teller is required to reveal the incoming link from the left, i.e. to reveal the source term in the input column and the associated e value that was revealed for that link. If the choice is to the *right*, the teller is required to reveal the outgoing link, i.e., the target term in the output column and again, the associated e value.

The point of this procedure is to ensure that for each link the teller will have a 50/50 chance of being audited whilst at the same time ensuring that there will never be any full links across the two mixes associated with each teller. Thus, we can never trace a ballot pair from the teller's input column to a term in its final output column. Furthermore, we can now audit each teller independently using this procedure and so ensure that any attempt to trace a ballot receipt across the mixes to the final decrypted vote will encounter multiple breaks.

For each of the revealed links, if the pair has been transformed correctly we should have:

$$\langle \mathcal{I}_{j-1}, \mathcal{E}_{j-1} \rangle \longrightarrow \langle \mathcal{I}_j, \mathcal{E}_j \rangle$$

where

$$\begin{aligned} \mathcal{E}_{j-1} &:= \{e_j, \mathcal{E}_{2i}\}_{T_j} \\ \mathcal{I}_j &:= \mathcal{I}_{j-1} \oplus d_j \\ d_j &:= h(e_j) \end{aligned}$$

Note that, along with the links, the corresponding e_j is revealed. Auditors can therefore compute $h(e_j)$, and check that this equals

$$\mathcal{I}_{j-1} \oplus \mathcal{I}_j$$

The T_j key is public so the auditor can also calculate

$$\{e_j; \mathcal{E}_j\}_{T_j}$$

and check that this equals \mathcal{E}_{j-1} .

The fact that e_j is revealed along with the link is significant in that it foils a potential attack on the secrecy: knowledge of the e_j value is essential in performing the checks. As this is the pre-image of a one-way function it should be intractable to compute this from knowledge of the final output, the d_j value.

If the checks could be performed knowing only the putatively linked pairs, then the scheme would be vulnerable to a guessing attack: Given a putative link, compute the checks. If the checks work you have, with high probability, identified a valid link. In this fashion you could, in principle, reconstruct as much of the secret permutation as needed. Without knowledge of the e_j pre-images, such an attack is intractable.

5 Visual cryptography

The full protocol as described in Section 7 uses visual cryptography, as does the original version; see [20]. In this version we first show how the one-dimensional protocol could benefit from visual cryptography. After

that we show how this can be extended to two dimensions.

For the remainder of this paper we make a different assumption about the printer that will produce the voting receipts. We assume that the receipt is printed on a strip of paper, where

1. the strip actually consists of two different layers, which can be separated — typically the voter will choose one layer while the other will be destroyed;
2. the printer is capable of printing on both layers at the same time;
3. the two layers are sufficiently transparent, allowing the voter to see what has been printed on both sides by looking from above.



5.1 Using visual cryptography for the one-dimensional protocol







We will now present a modified version of the one-dimensional protocol that takes full advantage of these new features. Observe that the only thing that changes is the ballot lay-out. In all other respects the protocol is completely identical.

Recall the bit representation of Anne’s vote for option “4”, using a slightly different alignment:







1	2	3	4	5	6
00	01	11	00	10	10
00	01	11	11	10	10

Observe that only in column 4 are the upper and lower bits each other’s complements.

Suppose now that we represent a 0 by  and 1 by . In this case the upper layer will look like

1	2	3	4	5	6
					

whereas the lower, *when viewed from above*, will look like:

1	2	3	4	5	6
					

If these two layers were printed on the same transparent strip of paper, the image would look like:

1	2	3	4	5	6
					

clearly showing to the voter which option he chose.

Observe that the effect of superimposing two transparent layers on each mini-pixel is that of a logical OR: the resulting mini-pixel is transparent only if the pixels on both layers are transparent; if either is opaque, the resulting pixel (as observed by a human) is opaque. However, when applying the bit coding introduced above, the actual effect of superimposing two bits (of four pixels each) is that if the two bits are equal, the result is 50 % transparent, but if the two bits differ then the results is opaque. This can be interpreted as an *xor* operation.

Observe also that by a minor change in the coding we can obtain a different visual effect. In Anne’s bit representation we had agreement of the bits in all columns, except in the column that represented her vote.

However, we can also do exactly the opposite by complementing the bits in the bottom layer, thus obtaining agreement only in the column representing her vote:

1	2	3	4	5	6
00	01	11	00	10	10
11	10	00	00	01	01

The corresponding ballot image would therefore be:

1	2	3	4	5	6
■	■	■	■	■	■

This is the strategy we will adopt in the two-dimensional visual cryptography as we will see below, since it enhanced visibility. Observe though that this coding implies a somewhat contradictory terminology, namely that the image of a blank, empty ballot is opaque, i.e. black.

5.2 Visual cryptography in two dimensions

Visual cryptography was first described in a paper by Naor and Shamir [29]. It is needed in the full protocol to allow write-in ballots. In its initial, most simple form it is like this: let V be a bit map image, represented by an $r \times s$ binary matrix. We want to hide this image in two transparent sheets with images W and W' , such that the two sheets superimposed show the image of V , whereas each sheet alone reveals no information about V .

This can be done as follows: generate W as a random $r \times s$ binary matrix (i.e. each entry is 0 or 1, chosen uniformly at random). Define W' as follows:

$$W'_{ij} = \begin{cases} W_{ij} & \text{if } V_{ij} = 1; \\ \text{COMPLEMENT}(W_{ij}) & \text{if } V_{ij} = 0 \end{cases}$$

The result is that two sheets superimposed are opaque on every pixel where $V_{ij} = 0$. But if $V_{ij} = 1$, the result will be transparent with probability $\frac{1}{2}$, depending on the (randomly chosen) value W_{ij} , see Figure 5.2. Observe, in the figure, that the “e” is barely recognizable; the resolution of W must be bigger than the resolution of V . Alternatively we can improve readability adopting the coding already introduced in the previous section: ■■ for 0 and ■■ for 1. Also, in this case a higher resolution for W and W' would be desirable, but the figure illustrates the main point of visual cryptography: neither image W nor W' reveals anything about the original image V , but the superimposition of the two images does. This is similar to the One Time Pad: neither the key K nor the cryptogram C reveals anything about the message M , but knowing both K and C the value M can be retrieved. In both cases it can be said that the message does not reside in one of the two parts W or W' (K or C , for the OTP), but only in the correlation between the two.

5.3 Visual cryptography as used in the full protocol

For the full protocol we make two refinements to the visual cryptography scheme as presented above.

In the first place, none of the bits come from a natural source of randomness. Instead, all bits are generated pseudo-randomly coming from an initial seed, whose value is obtained by encrypting the serial number of the vote with the private key of the Booth. This pseudo-randomness is necessary to restrict to a minimum the freedom of the Booth. If we allowed the Booth too much freedom by having it choose true random bits, it can find ways to cheat.

Furthermore, in the visual cryptography scheme present above, all true randomness came from W , and W' was just a function of V and W . In our case this not good enough. We need that the pseudo-randomness

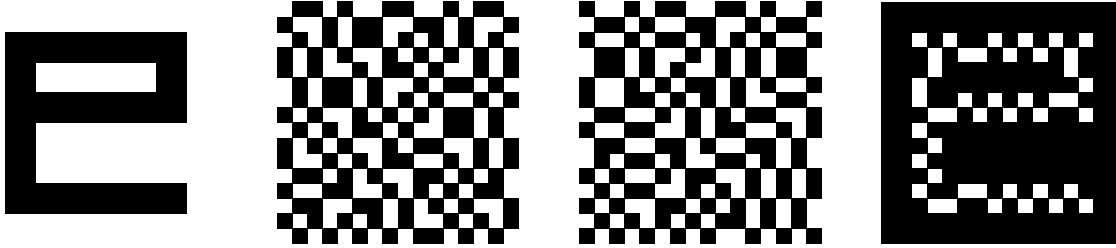


Figure 2: Visual Cryptography: The superposition of two random images produces a representation of a given image

be completely intertwined with both W and W' . Observe that this statement in the true random case is meaningless since there the role of W and W' are interchangeable.

We obtain the mentioned intertwining as follows. We will call a pixel (bit, matrix entry) W_{ij} *even* if $i + j \equiv 0 \pmod{2}$, and *odd* otherwise. Observe that even- and oddness create a checkerboard pattern. We now require that for even bits the pseudo-randomness comes from W_{ij} while W'_{ij} is computed from V_{ij} and W_{ij} . But for odd bits the opposite is true: the pseudo-randomness comes from W'_{ij} while W_{ij} is computed from V_{ij} and W'_{ij} .

Section 7 describes the use of visual cryptography in the full protocol.

6 Assumptions

6.1 Cryptographic assumptions

The protocol presented here does not use new cryptographic techniques, but combines existing methods in a novel way.

In this section we list the assumptions that we make, and outline several ways to implement them, thus showing that our assumptions are realistic. For this we use existing cryptographic methods, and sound practices for combining these methods into a robust protocol – a widely accepted approach. We are convinced that for the protocol to be realistic, speed is an important consideration, and that therefore a pragmatic approach to the cryptographic tools used is more important than mathematical rigor. In many cases we actually have a wide range of options, where faster usually means less rigorous, and vice versa.

Observe that our assumptions are used only to protect the privacy of the ballot. The correctness of the vote count of the protocol does not depend on any cryptographic or computational assumptions (see the Main Theorem 1).

ASSUMPTION A (EXISTENCE DETERMINISTIC SIGNATURE SCHEME) *We assume that a deterministic, public key signature algorithm exists.*

By *deterministic public key signature algorithm* is meant, as in standard texts such as [30, section 11.2], an efficient algorithm that associates with a message a deterministic *signature* generated using the private key of a public-private key pair. The signature can be efficiently checked by using the public key of the entity, but it is computationally infeasible, except with a probability close to that of a guess made uniformly at random, to construct, without access to the private key, another message with the same signature.

At various places we need a signature algorithm. When one is used by the Booth to construct a ballot, it is important that this algorithm be deterministic. Else, a booth could possibly use the freedom in determining the random bits to change the vote. The RSA cryptosystem certainly satisfies assumption A, but it is also possible that an ElGamal signature scheme could be made to do so.

ASSUMPTION B (EXISTENCE HASH FUNCTION) *We assume that one-way hash functions exist.*

By *one-way hash function* is meant, as in standard texts such as [30, section 9.4], a function that is efficient to compute, that maps a finite string of arbitrary length to one of fixed length, and for which it is computationally infeasible to determine the input, from the output, with probability sufficiently distinct from that of a guess, made uniformly at random.

ASSUMPTION C (EXISTENCE PSEUDO-RANDOM NUMBER GENERATOR) *We assume that a pseudo-random number generator exists.*

By *pseudo-random number generator* is meant, as in standard texts such as [30, chapter 5], an efficient deterministic algorithm that takes as input a fixed-length string, and outputs a larger string, such that it is computationally infeasible to predict, with non-negligible advantage, the $(l + 1)^{th}$ bit of the output from the first l bits.

The functions g_i are assumed to be one-way functions, while h is assumed to be a pseudo-random number generator. Such functions have been extensively studied and can be constructed from conventional encryption algorithms.

ASSUMPTION D (EXISTENCE EFFICIENT AND RELIABLE MIXING PROTOCOL) *We assume that a sound mixing protocol exists which has the following properties:*

- 1) *It allows several participants to submit encrypted strings to a public repository.*
- 2) *A group of authorities collectively decrypts the input strings while shuffling (permuting) the inputs.*

Mixing is a well-studied technique, introduced by Chaum [1]. Today many flavors of mixing exist (see [18]) but they all have in common that they use some public-key cryptosystem, like RSA or ElGamal. It is interesting to observe that the original paper [20] outlines a new variation of mixing, improving on the results of Jacobson, Juels and Rivest [6].

6.2 Other assumptions

ASSUMPTION E (BOOTH PRESERVES PRIVACY) *The booth destroys any information related to the key presses of the voter, as well as the ballot image of the layer not chosen by the voter.*

This assumption is not different from that for any other DRE. The security of cast ballots is not discussed in this paper, hence other methods need to be used to ensure that the Booth does not retain the entire vote and associate it with a serial number. The method described in this paper does not make the election more vulnerable to these problems than do other election schemes, and the problems may be addressed as they would be in other election schemes.

ASSUMPTION F (OTHER LAYER BALLOT DESTROYED) *The layer not chosen by the voter, and the information printed on it, will be completely destroyed.*

Since both layers together allow one to see the ballot, it is imperative that the layer not chosen be destroyed. It should also not be possible to take photos or make copies of its image.

ASSUMPTION G (TWO CONSECUTIVE HONEST TELLERS) *At least two consecutive tellers are honest.*

By *honest* is meant a teller that preserves ballot secrecy and does not provide information on the shuffle used by it; in particular, an honest teller performs a random shuffle. Note that honesty in ballot decryption is *not required* and tellers who do not decrypt correctly will be discovered with overwhelming probability during the audit. However, a teller who reveals the shuffle cannot be prevented from doing so by this protocol.

Alternatively, one may assume one honest Teller if each Teller operates four Mixes, instead of two.

7 The Complete Protocol

7.1 The Setting

7.1.1 Participants

The participants in the protocol are:

Voter: who may determine that her vote is counted

Booth: is responsible for recording the voter's vote. The system must catch attempts by the Booth to change votes.

Tellers: are responsible for ensuring that votes are counted and anonymity maintained beyond the Booth. This role is played out in physical elections by some combination of candidate representatives and government officials, depending on the country.

Interested Third Parties: may verify that the system is working as it should. This role is played out by organizations such as the League of Women Voters in physical elections in the US. The method described in this document requires the participation of a non-negligible fraction of voters, or, on their behalf, Interested Third Parties. This participation is the only way to detect attempts by the Booth to change votes.

Auditor: certifies that the election results are correct and have been determined by following the pre-specified, well-defined process. Who the Auditor is depends on who the results are being certified for. In physical elections in the US this role is played by a specified government/judicial official. In physical elections in some countries like India, this role is played by a citizen who is not directly answerable to the Parliament and hence is more independent of the current office bearers. In physical elections in new democracies, this role is played by organizations like the Carter Center, who may also function as Interested Third Parties. In the method described in this document, only a few independent audits are possible. More audits will compromise voter anonymity.

Public: Bulletin Board that holds all receipts, decrypted receipts, and audit results, and displays them to the public, thus enabling anyone to count the votes and follow the vote verification process.

7.1.2 The construction of the ballots and the receipts

In analogy with the simplified protocol presented in section 4, we explain in the first steps how the Booth generates an empty ballot. We do this for expository purposes only. In reality the booth has no need to generate an empty ballot but could generate the ballot image showing the voter's option once she has entered her choice. We then modify this empty ballot with the voter's choice, and create an image of it.

Step C1: Creation of q

First, the Booth generates a true random number q , the serial number of the ballot.

Step C2: Computation of s^U and s^L

where s^a is the Booth's digital signature on q using the private key of the corresponding layer.

Step C3: Computation of seals e_i^U and e_i^L

We suppose that g_i is a secure family of one-way functions; possibly $g_i(\cdot) := g(i; \cdot)$ for a suitable function g . It provides distinct values e_i^a for each Teller so that no Teller may be able to compute another's share: $e_i^a := g_i(s^a)$

Step C4: Computation of pseudo-random xor shares d_i^U and d_i^L

$d_i^U := h(e_i^U)$ and $d_i^L := h(e_i^L)$ respectively. Further, h and g_i are public functions. The function h is also one-way; its image should be n bits, half the size of the ballot image.

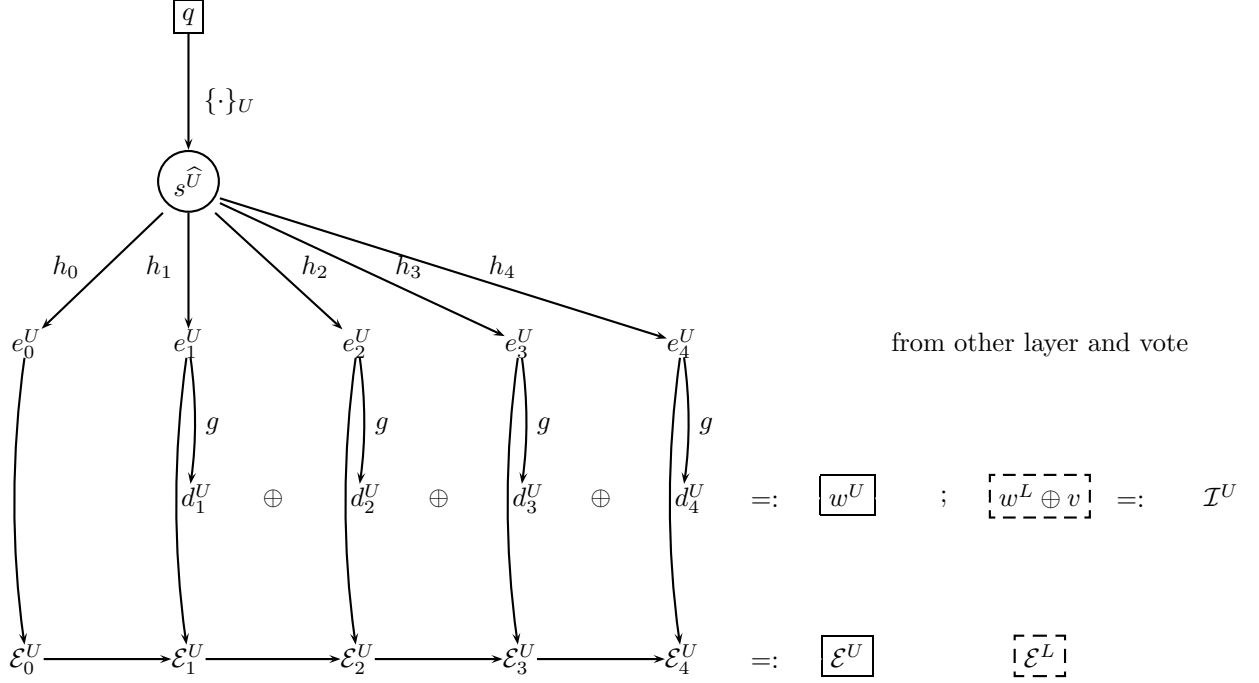


Figure 3: Construction of \mathcal{B}^U , the upper half of the ballot, from the serial number q . The boxed values appear on the ballot; the circled value will only be printed for the layer that the voter chooses to retain. The envelopes are defined as: $\mathcal{E}_{i+1}^U := \{e_{i+1}, \mathcal{E}_i^U\}_{\hat{T}_{i+1}}$. The construction of \mathcal{W}^L is similar, with U and L interchanged.

Step C5: Computation of pseudo-random sequences w^U and w^L

Two pseudo-random bit sequences w^U and w^L of size n are computed as *xor*-sums of shares: one corresponding to each Mix, the share of the i^{th} Mix being d_i^a .

Hence the two pseudo-random number sequences generated are:

$$w^a = \bigoplus_{i=1}^{2k} d_i^a = \bigoplus h(g_i(s^a)) \quad (1)$$

where w^a is the pseudo-random number sequence for layer $a \in \{U, L\}$. Each sequence has n bits.

Step C6a: Creation of two empty ballot images \mathcal{I}^U and \mathcal{I}^L

The pseudo-random sequences will result in two ballot images \mathcal{I}^U and \mathcal{I}^L as follows: The pixels in each image are divided into two disjoint subsets: one contains all the even pixels and the other all the odd pixels. Here a pixel is called even if its row number added to its column number is even, and odd otherwise. \mathcal{I}_{odd}^a denotes the odd pixels in image \mathcal{I}^a , and \mathcal{I}_{even}^a the even ones.

For each image, one of these subsets is pseudo-randomly generated; analogous to the one-dimensional protocol, these may be thought of as the encryption bits. For instance, \mathcal{I}_{even}^L will be defined by using sequence w^L ; likewise \mathcal{I}_{odd}^U is determined by w^U . The pixels of the subset with complement parity are computed by taking the bits of the complement layer, *xored* with the vote image \mathcal{V} . However, an empty ballot image \mathcal{V} with all zeroes in its pixel position will not induce a difference between any two pixels of the two layers, so $\mathcal{I}^U = \mathcal{I}^L$.

We use the convention that \blacksquare means 0 and \blacksquare means 1. When superimposing two pixels we say that the resulting pixel is transparent if the both pixels agree, if the pixels are different the result

will be opaque. It follows that an empty ballot image will be "transparent": none of the pixel positions will be opaque. But alternatively, if we choose to flip completely the bits of one (say the bottom) layer, it follows then that an empty ballot is completely opaque.

Step C6b: Creation of two ballot images \mathcal{I}^U and \mathcal{I}^L with vote V

As was stated above, half of the pixels of the ballot image are computed by taking the bits of the complement layer $xored$ with the vote image V . So we can write that $\mathcal{I}_{odd}^L = \mathcal{I}_{odd}^U \oplus V$ and $\mathcal{I}_{even}^L = \mathcal{I}_{even}^U \oplus V$. In other words, the Booth modifies the two binary images \mathcal{I}^U and \mathcal{I}^L (which constituted an empty ballot) such that superimposing the two images shows the vote to the voter. This essentially comes down to flipping the pixel values for those positions where $V_{ij} = 1$.

Recall that for the "regular" pixels we had that $\mathcal{I}_{even}^L := w^L$ and $\mathcal{I}_{odd}^U := w^U$, so that in terms of information content we can write $\mathcal{I}^L = \langle w^L, w^U \oplus v \rangle$ and $\mathcal{I}^U = \langle w^U, w^L \oplus v \rangle$. However, the information is presented in a very special way, allowing the voter to see his vote easily.

Observe the close analogy between this protocol and the one-dimensional protocol presented earlier—even the intricate intertwining of the two layers by using alternate pixel positions is essentially the same.

We provide a more formal description of the creation of the ballot images \mathcal{I}_U and \mathcal{I}_L from w^U, w^L and V . Let $f(i, j)$ (and $f'(i, j)$) be one-to-one functions that associate each even (or odd) matrix entry to an integer between 1 and n , inclusive.

For even pixels we define:

$$\mathcal{I}_{ij}^L = \begin{cases} w_{f(i,j)}^L & \\ \mathcal{I}_{ij}^U = \begin{cases} w_{f(i,j)}^L & \text{if } V_{ij} = 0; \\ \text{COMPLEMENT}(w_{f(i,j)}^L) & \text{if } V_{ij} = 1 \end{cases} \end{cases}$$

For odd pixels we interchange the role of U and L and use f' instead of f . So we define:

$$\mathcal{I}_{ij}^U = \begin{cases} w_{f'(i,j)}^U & \\ \mathcal{I}_{ij}^L = \begin{cases} w_{f'(i,j)}^U & \text{if } V_{ij} = 0; \\ \text{COMPLEMENT}(w_{f'(i,j)}^U) & \text{if } V_{ij} = 1 \end{cases} \end{cases}$$

It follows that for pixels where $V_{ij} = 0$, the superposition of \mathcal{I}^U and \mathcal{I}^L will be opaque, whereas if $V_{ij} = 1$ then the pixel is transparent. For well-designed bitmaps the ballot image will clearly show the user's vote.

The two images, when overlaid, provide a pictorial representation of the voter's choices, image \mathcal{I} .

Step C7: Construction of the envelopes \mathcal{E}^U and \mathcal{E}^L

As the decryption of the votes is done in a mixnet style, we use the mixnet analogy of nested envelopes to describe these strings. A string encrypted with a public key is thought of as a sealed envelope. Anyone can seal it, but only the entity holding the private key can open it. The values \mathcal{E}_i^U and \mathcal{E}_i^L are sealed envelopes for the i^{th} Mix, and each contains $i - 1$ other nested envelopes for the other Mixes.

The innermost envelopes, \mathcal{E}_1^U and \mathcal{E}_1^L are generated first and contain the seal e_1 for the first Mix. The $2i^{th}$ envelopes contain the $(2i - 1)^{th}$ envelope and a seal for the i^{th} Teller. At decryption time, the envelopes are opened in reverse order, the $2i^{th}$ envelope containing, when opened, the $(2i - 1)^{th}$ envelope and a seal for the i^{th} Authority. Only the i^{th} Authority can open the $2i^{th}$ and $(2i - 1)^{th}$ envelopes.

More formally, $\mathcal{E}^U := \mathcal{E}_{2k}^U$ and $\mathcal{E}^L := \mathcal{E}_{2k}^L$ represent the final encrypted strings, and \mathcal{E}^a contains enough information for each Authority to generate its share of the pseudo-random number sequence in layer a . The value \mathcal{E}_{2k}^a is computed recursively as follows:

$$\mathcal{E}_1^a = \{e_1^a; e_0^a\}_{\widehat{T}_1} \quad (2)$$

where a represents U or L . The seal e_{2i}^a and the envelope \mathcal{E}_{2i-1}^a for the i^{th} Mix are encrypted together inside \mathcal{E}_{2i}^a , which can only be decrypted by Mix i :

$$\mathcal{E}_{2i}^a = \{e_{2i}^a; \mathcal{E}_{2i-1}^a\}_{\widehat{T}_i} \quad (3)$$

Similarly for \mathcal{E}_{2i-1}^a .

Step C8: Printing of the ballot/receipt

The values of the ballot are not at all printed in the order in which they are computed; some don't get printed at all.

The ballot, \mathcal{B} , has two layers, \mathcal{B}^U and \mathcal{B}^L consisting of the following components, assuming the voter chooses the upper layer:

\mathcal{I}^U	q	\mathcal{E}^U	\mathcal{E}^L	s^U	z	RETAIN	UPPER
\mathcal{I}^L	q	\mathcal{E}^U	\mathcal{E}^L	--	--	DESTROY	LOWER

Recall that \mathcal{I}^a will be printed as an image, whereas $q, \mathcal{E}^U, \mathcal{E}^L, s^a$ and z are numbers. The vertical bars show how much is printed during the various steps of the protocol, as will be explained in the next section. For instance, the values right of the second vertical bar will only be printed after the voter chose a layer, and the value z , which depends on the layer chosen, will be explained in the next step.

Step C9: Computation of the digital signature on the receipt

After the voter has chosen the layer to keep but before leaving the Booth, the latter must sign the receipt the voter keeps in order to keep the Booth from disavowing it. Let \mathbf{c} be the layer chosen.

Define $\bar{\mathcal{B}}^{\mathbf{c}} = \langle \mathcal{I}^{\mathbf{c}}, q, \mathcal{E}^U, \mathcal{E}^L, s^{\mathbf{c}} \rangle$ as an encoding of the contents of the receipt. The booth's signature on the entire receipt is now calculated using the Booth's private key \widehat{Z} , so we get that $z = \{\bar{\mathcal{B}}^{\mathbf{c}}\}_{\widehat{Z}}$.

7.2 The Protocol

The protocol consists of four stages:

- I In the polling booth;
- II receipt broadcast and pre-count verifications;
- III counting;
- IV audit.

7.2.1 Stage I: In the polling booth

Step B1: Booth creates an empty ballot

See 7.1.2 for a description of how a ballot is created.

Step B2: Voter defines ballot image

The ballot image $\mathcal{I} = \langle \mathcal{I}^U, \mathcal{I}^L \rangle$ will be printed, together with the serial number q .

We assume for the text that the voter confirms his choice. If the voter changes his mind about the candidate, the ballot will be canceled and destroyed, and the process returns to Step 1, in which a new ballot is generated with a new q .

Step B3: Ballot image generation

The voter's receipt consists of two layers, each of which has four fields: a binary image, and three strings. The three strings on both receipts are identical, and the images, when overlaid, produce an image representing the vote:

$$\begin{aligned} \text{upper layer: } & \langle \mathcal{I}^U, q, \mathcal{E}^U, \mathcal{E}^L \rangle \\ \text{lower layer: } & \langle \mathcal{I}^L, q, \mathcal{E}^U, \mathcal{E}^L \rangle \end{aligned}$$

One of the layers will be retained as a record of the vote, by the Voter and the Booth. We denote this as the *chosen* layer, by the letter $\mathbf{c} \in \{U, L\}$. The other layer, $\bar{\mathbf{c}} \in \{U, L\}$, is destroyed, but the information required to reconstruct the encryption bits in it is sent, in encrypted form, to the group of Tellers. Using these, the vote can be reconstructed by the Tellers from the chosen layer. The protocol ensures that the reconstruction can be done only with the cooperation of the k Tellers.

The encrypted seeds for both layers are communicated to the Tellers along with the encrypted vote. The pseudo-random pixels (i.e. the encryption bits) in the destroyed layer can be reconstructed using the seeds, leading to the reconstruction of the vote at these positions. The information bits, however, are determined by the vote image, V and hence cannot be reconstructed independently by the Tellers. Assuming the pixels are small enough, however, and the ballots well-designed, it is enough to reconstruct the vote at alternate pixel positions.

Pixels in alternate rows and columns of each layer are pseudo-random bits and placed so that bits from one layer are staggered by one unit from those of the other. The seals e_i for each Authority are encrypted sequentially and the envelopes \mathcal{E}^U and \mathcal{E}^L printed on both layers, along with the serial number, which serves as a commitment that can be checked later.

Step B4: Voter Check and Choice

The voter checks that the two superimposed layers provide a visual representation of her vote, i.e. that $\mathcal{I}^U \oplus \mathcal{I}^L = \mathcal{I}$. She checks that there are three numbers (q , \mathcal{E}^U and \mathcal{E}^L) also printed at the lower part of both layers, and that the numbers are the same on both layers. She chooses a layer to take away, and communicates her choice to the Booth. If the Booth gives her the wrong layer to take away, she can prove this if the methodology of communicating the choice is set up as follows.

There are two circles, perhaps filled in in different colors, one on each layer. The circles do not overlap and each contains text indicating the layer it is in. Because the layers are transparent, the voter can see both circles when the two layers are placed on top of each other, and the text in each. The voter chooses a layer by making a punch mark in the correct circle. If the Booth gives him the wrong layer, this can be proven because the circle will not be punched. If any layer has two circles, this can be caught with a non-zero probability when the voter chooses that layer, or the booth gives it.

After the voter chooses a layer, $s^{\mathbf{c}}$, the digital signature on the serial number *for the chosen layer* is printed. This is similar to opening a string commitment. However, the Booth cannot choose this string since it is a deterministic function of the serial number q . Also printed is a signature of the entire receipt, z , to prevent forgery and false accusations of election fraud. The checks are described in more detail in section 7.2.2

Step B5: Various checks printed and end of vote casting

The Booth now prints, only on the chosen layer, digital signatures

- $s^{\hat{\mathbf{c}}} = \{q\}_{\hat{\mathbf{c}}}$ - of the serial number, and

- $z = \{\mathcal{I}^c, q, \mathcal{E}^c, \mathcal{E}^{\hat{c}}, s^{\hat{c}}\}_{\hat{z}}$ - of the entire document,

where c represents the chosen layer. The voter gets the chosen layer, the other is destroyed. Even if the untrusted Booth does not destroy the other layer, it cannot change the vote count without being caught with a high probability for enough votes tampered with (see Theorem 6).

7.2.2 Stage II: Receipt broadcast and validity checks prior to counting

Step R1: Receipt broadcast

Receipts for $q \in \mathcal{Q}$ (where \mathcal{Q} represents the values of q for all cast votes) are displayed in a publicly accessible place. Those with receipts (voters or Interested Third Parties) can check that their receipts (and hence votes) have been correctly retained.

This step is not optional, and a “large enough” fraction of receipts must be checked for confidence in election results. See section 9 for comments on how large is “large enough”.

Step R2: Validity check

Any Interested Third Party can check that all the receipts broadcast were correctly generated. This is done by checking the commitments as follows. Let $\langle \mathcal{I}, q', E^U, E^L, s, z \rangle$ represent a receipt. Then the following 4 checks must be performed:

Check 1: The signature s on the serial number q' should be valid, so the test $\{s\}_C = q'$ should be true. Here C stands for the public key of the booth corresponding to layer c .

Check 2: For the layer chosen, the string s should have been used as input to the functions g_i and h , resulting in the values d_i^c , whose xor results in w^c . These bit values are depicted as pixel values in the ballot image \mathcal{I}^c ; for the upper layer they correspond to the odd pixels. For the lower layer to the even pixels.

Check 3: Of the two envelopes printed on the receipt, one now can be completely verified, since it should have been generated as specified by equations (2) and (3).

Check 4: The receipt should have been correctly signed, i.e. check that $z = \{\mathcal{I}^c, q, \mathcal{E}^c, \mathcal{E}^{\hat{c}}, s^{\hat{c}}\}_{\hat{z}}$

These checks are the only way to detect Booth cheating, and are hence not “optional”. Again, a large enough fraction of receipts must pass these checks for confidence in the election results. Typically the election authority itself, as well as Independent Third Parties, will run the check for all displayed receipts.

7.2.3 Stage III: Vote Tallying

Step T1: Vote batch enters Mix

All strings except the required envelope are stripped off each vote, and the collection, $q \in \mathcal{Q}$, is passed on to the k^{th} Teller. For each vote, the k^{th} Teller gets the voter’s chosen layer (receipt) and the entire envelope for the other layer, so that, together, the Tellers may decrypt the vote. Thus the k^{th} Teller gets a sequence of pairs (consisting of the ballot image and the associated strings) ordered by the corresponding value of $q \in \mathcal{Q}$: $(\langle \mathcal{I}^c, \mathcal{E}_{2k}^{\hat{c}} \rangle_r)_{r \in \mathcal{Q}}$, where the first term is the r^{th} ballot image obtained by the k^{th} Teller, c is the chosen layer for serial number r , and $\mathcal{E}_{2k}^{\hat{c}}$ is the nested encryption of information read by the Teller to decrypt the chosen layer. Though there is no longer a serial number on the ballot, the order of the ballots can reveal information on the original serial number.

The encryption bits on the destroyed layer need to be generated to recreate the ballot image. If, for example, \mathcal{I}^L is taken by the voter, \mathcal{E}_{2k}^U will be opened by the Tellers, and information obtained to compute the encryption bits in \mathcal{I}^U . This allows for \mathcal{I}_{odd} to be computed. The other envelope, \mathcal{E}_{2k}^L , contains the seeds required to generate the pseudo-random numbers in the lower layer, \mathcal{I}^L . It will serve as a commitment to check that the Booth is indeed communicating correctly with the Tellers.

The encryption bits in the destroyed layer are generated in parts by each Teller, and added on to the chosen layer sequentially. After the last Teller adds on its part, decrypted ballot images are obtained. These may now be counted, in public. Each Teller shuffles his output images, so the original order is not retained and voting is anonymous.

We denote by π_i the permutation applied by the i^{th} Mix, and by \mathcal{J}_{i+1} a single ballot image input by the i^{th} mix. Thus, $\mathcal{J}_{2k+1} = \mathcal{I}^c$. Let the sequence of pairs

$$(\langle \mathcal{J}_{i+1}, \mathcal{E}_i^{\bar{c}} \rangle_1, \langle \mathcal{J}_{i+1}, \mathcal{E}_i^{\bar{c}} \rangle_2, \dots, \langle \mathcal{J}_{i+1}, \mathcal{E}_i^{\bar{c}} \rangle_{|\mathcal{Q}|})$$

be the input sequence of ballots for the i^{th} Mix. Then

$$(\langle \mathcal{J}_i, \mathcal{E}_{i-1}^{\bar{c}} \rangle_{\pi^{-1}(1)}, \langle \mathcal{J}_i, \mathcal{E}_{i-1}^{\bar{c}} \rangle_{\pi^{-1}(2)}, \dots, \langle \mathcal{J}_i, \mathcal{E}_{i-1}^{\bar{c}} \rangle_{\pi^{-1}(r)})_{r=1}^{|\mathcal{Q}|}$$

is the output sequence. The i^{th} Mix computes each output pair by opening the corresponding envelope passed on by the previous Mix. The envelope contains the seed for the PRNG share of the Mix, and envelope for the next Mix.

The following two steps, T2 and T3, are performed by each Mix in sequence.

Step T2: Envelope Opening and Decryption for each receipt

Step T2a: Envelope Opening

The Mix decrypts the envelope to obtain the values α_i and an envelope $\mathcal{E}_{i-1}^{\bar{c}}$ to pass on to the next Mix, see (3): $\{\mathcal{E}^{\bar{c}}\}_{\hat{A}_i} = (\alpha_i, \mathcal{E}_{i-1}^{\bar{c}})$

Step T2b: Xor pseudo-random contribution

The Mix generates the pseudo-random bit sequence and adds it to the appropriate pixel values; this is the Mix's contribution to the decryption key.

$$\mathcal{J}_i := \mathcal{J}_i \oplus h(\alpha_i) \tag{4}$$

where both \mathcal{J}_i and $h(\alpha_i)$ are sequences. If the envelopes are correctly constructed, $\alpha_i = g_i(\langle q \rangle_{\hat{c}})$, and $\mathcal{J}_i = (\mathcal{I}_i)_{\text{even}}$ or $\mathcal{J}_i = (\mathcal{I}_i)_{\text{odd}}$.

Step T3: Shuffle

Each Mix shuffles the entire set of votes it receives. The shuffle is retained for the audit. Shuffled output pairs are broadcast to Public from where the next Mix obtains them. The sequence of shuffled output pairs is:

$$(\langle \mathcal{J}_i, \mathcal{E}_{i-1}^{\bar{c}} \rangle_{\pi^{-1}(1)}, \langle \mathcal{J}_i, \mathcal{E}_{i-1}^{\bar{c}} \rangle_{\pi^{-1}(2)}, \dots, \langle \mathcal{J}_i, \mathcal{E}_{i-1}^{\bar{c}} \rangle_{\pi^{-1}(|\mathcal{Q}|)})$$

It can be seen that, at the very end, $w^{\bar{c}}$ is reconstructed by all Mix contributions:

$$\mathcal{V}_{\text{odd}} = (\mathcal{I}^c)_{\text{odd}} \oplus \sum_{i=1}^{2k} h(\alpha_i)$$

(The right hand side of the above equation can be seen to be the sum of two sequences). Similarly for $\mathcal{V}_{\text{even}}$.

Recall that, if the envelopes are correctly constructed, $\alpha_i = g_i(\langle q \rangle_{\hat{c}})$, and $\mathcal{J}_i = (\mathcal{I}_i)_{\text{even}}$ or $\mathcal{J}_i = (\mathcal{I}_i)_{\text{odd}}$.

Step T4: Tallying

The decrypted ballots are on Public and may be tallied by anyone.

7.2.4 Stage IV: Audit and Certification

The only way for the i^{th} Mix to affect the vote count is by generating false values of \mathcal{J}_i . The probability of this can be decreased exponentially with the number of votes cheated on by performing the following audit (see Theorem 8).

Step A1: Teller audit

For the i^{th} Teller, check that the sequence of output pairs

$$(\langle \mathcal{J}_{2i-1}, \mathcal{E}_{2i-2}^{\bar{c}} \rangle_{\pi^{-1}(1)}, \langle \mathcal{J}_{2i-1}, \mathcal{E}_{2i-2}^{\bar{c}} \rangle_{\pi^{-1}(2)}, \dots, \langle \mathcal{J}_{2i-1}, \mathcal{E}_{2i-2}^{\bar{c}} \rangle_{\pi^{-1}(|\mathcal{Q}|)})$$

was correctly constructed from the sequence of input pairs

$$(\langle \mathcal{J}_{2i+1}, \mathcal{E}_{2i}^{\bar{c}} \rangle_1, \langle \mathcal{J}_{2i+1}, \mathcal{E}_{2i}^{\bar{c}} \rangle_2, \dots, \langle \mathcal{J}_{2i+1}, \mathcal{E}_{2i}^{\bar{c}} \rangle_{|\mathcal{Q}|})$$

with high probability, where π is the permutation used by the Teller.

This is done by providing each Mix with a set of randomly chosen votes numbering half the total number of votes, and requiring it to prove each decryption was performed correctly. Consecutive Mixes corresponding to a single Teller prove correct decryption on complementary sets of votes, so no single vote can be linked from the input to the Teller to its output. Further, each decryption set presented to a particular Teller consists of half of each of the two sets presented to the previous Teller. This may be stated more formally as below.

The i^{th} Teller is provided a subset \mathcal{Q}_i of its output ballots, i.e. output ballots of Mix $2i - 1$. For these output ballots, the Teller is required to prove correct decryption, by providing the corresponding input ballot positions for the Mix, and the decrypted seeds. The Teller is also required to similarly prove correct decryption by Mix $2i$, on the ballot set \mathcal{Q}_i – ballots not opened in the first Mix. Hence, Teller i is required to provide the sequence of pairs $\{\langle \pi_{2i-1}^{-1}(q), \alpha_1 \rangle_q\}_{q \in \mathcal{Q}_i}$, and $\{\langle \pi_{2i}^{-1}(q), \alpha_2 \rangle_q\}_{q \in \bar{\mathcal{Q}}_i}$, where π_j is the permutation used by Mix j .

The values provided by each Teller are checked to be consistent with the public values of the input and output of the respective Mix. That is, it is checked that the values of each pair $\{\langle \pi_j^{-1}(r), \alpha_j \rangle_r\}$ are correct with respect to equations (3) and (4), i.e. that

$$[\mathcal{E}_{2i}^{\bar{c}}]_r = [\langle \alpha_{2i-1}, \mathcal{E}_{2i-1}^{\bar{c}} \rangle_{A_i}]_{\pi_{2i-1}(r)}, \quad \forall r \in \mathcal{Q}_i$$

and:

$$[\mathcal{J}_{2i-1}]_{\pi_i(r)} = [\mathcal{J}_{2i} \oplus h(\alpha_{2i-1})]_r \quad \forall r \in \mathcal{Q}_i$$

and similarly for the second Mix operated by the Teller.

In determining how \mathcal{Q}_i is chosen, there are two cases to consider, depending on whether i equals 1 or not:

- If $i = 1$, i.e. the Teller is the one that produces the plaintext votes, \mathcal{Q}_1 is chosen uniformly at random. That is, $\mathcal{Q}_1 \subset \mathcal{Q}$ is chosen uniformly at random such that $|\mathcal{Q}_1| = \frac{1}{2}|\mathcal{Q}|$.
- For $i \neq 1$, the Teller is required to provide the same information for a random set of output ballots \mathcal{Q}_i such that \mathcal{Q}_i consists of an equal proportion of votes from \mathcal{Q}_{i-1} and its complement. More formally, $\mathcal{Q}_i \subset \mathcal{Q}$ is chosen such that:
 - \mathcal{Q}_{i1} and \mathcal{Q}_{i2} are of equal size and form a partition of \mathcal{Q}_i , that is:
 - * $\mathcal{Q}_i = \mathcal{Q}_{i1} \cup \mathcal{Q}_{i2}$
 - * $\mathcal{Q}_{i1} \cap \mathcal{Q}_{i2} = \emptyset$
 - * $|\mathcal{Q}_{i1}| = |\mathcal{Q}_{i2}| = \frac{1}{4}|\mathcal{Q}|$
 - \mathcal{Q}_{i1} and \mathcal{Q}_{i2} are chosen from \mathcal{Q}_{i-1} and $\bar{\mathcal{Q}}_{i-1}$ respectively:

- * $Q_{i1} \subset Q_{i-1}$
- * $Q_{i2} \subset \overline{Q}_{i-1}$
- Q_{i1} and Q_{i2} are chosen uniformly at random.

In a single audit, each vote a Teller cheats on is detected with probability $\frac{1}{2}$. If the Teller cheats on t votes, the probability that he is not detected is $(\frac{1}{2})^t$.

8 Formal statements about the protocol

We can now state our results. First we state the most important characteristics of our protocol in the main theorem. Then we will state more detailed properties and theorems related to our protocol in order to show how well the protocol satisfies the requirements formulated in Subsection 3.3. The proof of the main result follows easily and is presented at the end of this section.

To simplify the formulation of our results, we introduce the following terminology to express more precisely what we mean when consider a system secure. We say that a system or protocol *leaks no useful information* about random variable X if, given all the data output by the system or protocol, it is computationally infeasible to distinguish the prior distribution of X from the posterior distribution of X with a probability sufficiently distinct from one half. In particular, when X is uniformly distributed, it is computationally infeasible to determine X with a probability sufficiently distinct from that of a guess made uniformly at random.

We can now formulate the main result of this paper.

THEOREM 1 (MAIN RESULT)

If Assumptions A through F are satisfied, then the protocol presented in Section 7 has the following general characteristics:

- a) The receipt that the voter retains leaks no useful information about her vote.*
- b) The mixing process reveals no useful information about individual votes. Combined with a), this implies that our protocol guarantees the privacy of the voter.*
- c) If a voter has a (signed) receipt which is not posted, then this constitutes an irrefutable proof that the election system is cheating.*
- d) Since each ballot is decrypted and posted by a deterministic process, the system can only cheat by trying to alter ballots. For each correctly-posted ballot there are only two strategies in which the system, no matter how it operates, has a chance of altering it without being detected: (1) by printing an incorrect layer and hoping that the voter chooses the other layer; or (2) by incorrectly performing a step in the mixing process and hoping that this particular step will not be selected for audit. For either strategy the chances of not being detected is $1/2$ (for (1) the probability may be smaller; in general, if the probability of the upper layer being chosen is β , then the value $1/2$ in the statement must be replaced by γ , where γ is the smaller of $1 - \beta$ and β).*
- e) It follows from d) that if each voter chooses his layer with probability $1/2$, and verifies that his receipt is indeed posted, then the probability of the system altering t votes without being detected is $1/2^t$. This statement is true unconditionally, so even with unlimited computational resource the system cannot increase this probability. In other words, our protocol guarantees the correctness of the vote count unconditionally, except with negligible probability.*

Now we list our claims about the full protocol as presented in the previous Section, by verifying to what extent our protocol satisfies Requirements A to H as listed in Subsection 3.3 on Page 5. Our claims are formulated as Properties and Theorems. The difference is that a Property is more self-evident, for instance following from common-sense notions. A Theorem is followed by a proof.

Requirements A and B say that only authorized persons, called voters, can create a ballot and deposit it in the ballot box, and that a voter can cast at most one vote. Our protocol does not provide new solutions to these two requirements, and we make no special claims about this subject. Proper voter authentication is

outside the scope of this paper. Hence, ballot stuffing, false electoral rolls, and the separation between voter and assigned ballot card would have to be addressed through different means.

Requirement C, about individual ballot security, reads: *Filling in the ballot and putting it in the ballot box is a confidential act, and under no circumstance, not even with the connivance of the voter, should it be possible to deduce for whom or for what the voter casted his vote.*

We present three theorems to prove our protocol satisfies this requirement. The first one states that the receipt alone reveals no useful information, the second that the mixing process preserves the anonymity of the posted votes, and the third that, in the absence of information on links between any encrypted and decrypted votes, that the audit process also preserves the anonymity.

THEOREM 2 (RECEIPT ALONE REVEALS NO USEFUL INFORMATION)

If Assumptions A, B, E and F are satisfied, then the following holds: when the voter leaves the booth with his receipt, i.e. when Step B6 has been completed, then his receipt alone reveals no useful information about his vote. Consequently, displaying (copies of) the receipt images in a public place does not reveal useful information about the vote.

PROOF: As shown in Figure 3, the vote is contained in the layer chosen because the latter contains the value of $v \oplus w^{\bar{c}}$ encoded in the image. We must therefore argue that $w^{\bar{c}}$ cannot be computed and is indistinguishable from random.

According to the protocol, only one of the values s^U or s^L gets revealed, namely s^c . It is obvious that if $s^{\bar{c}}$ were revealed also, then $w^{\bar{c}}$ could be computed.

The fact that $s^{\bar{c}}$ cannot be computed from q results from the assumption A, which states that $\{\cdot\}_U$ is a good signature function.

Also, $s^{\bar{c}}$ cannot be computed from $\mathcal{E}^{\bar{c}}$ because of assumption D, which implies that the values $e_i^{\bar{c}}$ are well-protected. (We will see later that some of the e_i will be revealed during the audit of the mixing process, but this will not be a problem either.)

And finally, $s^{\bar{c}}$ cannot be computed from the other information on the ballot, s^c , w^c and \mathcal{E}^c , since these values are all related to the other layer, and any attack along these lines would be at least as complicated as one of the attacks outlined above.

$w^{\bar{c}}$ cannot be guessed because of the unpredictability of the results of the functions g_i and the bits generated by h , (see B and C). ◀

THEOREM 3 (MIXING REVEALS NO USEFUL INFORMATION) *If Assumptions A, B, C and G are satisfied, then the following holds: Given any vote in the input batch, Stage III, the vote tallying, does not reveal any useful information about the corresponding vote in the input batch.*

PROOF:

This follows from the fact that mixing in general does not reveal any information about the index of the vote in the input batch (from Assumption D and the discussion in the remaining part of that section.) ◀

Stage IV, the audit process, does reduce the anonymity of the protocol slightly, in the following manner. The permutation of the votes is *no longer* pseudorandom after the audit, because exactly half the votes in \mathcal{Q}_i originate in \mathcal{Q}_{i-1} , which restricts the set of possible permutations. Even though the permutation is no longer pseudorandom, the construction of the audit ensures that the probability distribution on the input vote corresponding to any specific output ballot is uniform (this is in contrast to other randomized partial audits [6] where the distribution is not uniform). When the information revealed by the audit is enhanced by the knowledge of links between certain input/output ballot pairs, certain permutations may further be ruled out or deemed less likely, and this may change the probability distribution on the remaining input votes to a non-uniform one. Note that links between certain input/output pairs does not change the distribution on the remaining links in a mixnet that is not audited thus.

The following theorem provides the statement and proof of the anonymity provided when no input/output vote links are known.

THEOREM 4 (AUDITING PRESERVES ANONYMITY IN THE ABSENCE OF ANY KNOWN LINKS) *If Assumptions A, B, C and G are satisfied, then the following holds: The probability distribution on the input vote that corresponds to any specific output vote is uniform, in the absence of any information on links between input and output votes.*

PROOF:

First we show that the opening of links does not change the anonymity of the mixnet. This is easily seen as follows, and is shown in [31]: of the entire set of votes entering the i^{th} Teller, \mathcal{Q} , exactly half are opened by the first mix during the audit, \mathcal{Q}_i . Its complement, $\overline{\mathcal{Q}}_i$, is opened by the second mix. It is known whether a given entering vote is in \mathcal{Q}_i or $\overline{\mathcal{Q}}_i$. Hence, at the output of the second mix (i^{th} Teller), after the audit, it is known which subset of output votes (for the i^{th} Teller) a given input vote (for the i^{th} Teller) is in. Hence the votes are no longer uniformly distributed in the output.

Of the output votes, half are independently chosen to be opened at Teller $i + 1$, \mathcal{Q}_{i+1} . At the output of the second Mix run by Teller $i + 1$, a particular vote is known to have come from \mathcal{Q}_{i+1} or $\overline{\mathcal{Q}}_{i+1}$. But \mathcal{Q}_{i+1} itself is half from \mathcal{Q}_i and half from $\overline{\mathcal{Q}}_i$, i.e. a single vote in \mathcal{Q}_{i+1} is half as likely to be from \mathcal{Q}_{i+1} as it is from $\overline{\mathcal{Q}}_{i+1}$.

Second, we must show that the values revealed during the audit in Step A1 of the protocol do not help to compute $w^{\bar{c}}$ and therefore v . But this can be easily seen as follows. Each Teller runs two mixes, and the Audit protocol requires him to open at most one value e related to the same vote; according to Step A1, Teller i either shows $e_{2i-1}^{\bar{c}}$ or $e_{2i}^{\bar{c}}$. So as long as at least one Teller remains honest and never divulges the value not opened during the audit, the v cannot be computed and the privacy of the ballot is assured. ◀

Recall that Requirement D about individual voter verifiability reads as follows: *The voter i) can verify that he created a valid vote, ii) can revise his vote before committing, and iii) can convince himself that his vote is included in the set of votes tallied.* Subrequirement ii) is easily verified; but the first and the last one require more effort. Let us first present two theorems that deal with Subrequirement i).

THEOREM 5 (WHEN THE BOOTH IS HONEST, THE BALLOT IS WELL-FORMED AND UNAMBIGUOUS)

The images on the two layers together unambiguously define an image of a vote, as seen by the voter. This image corresponds to the vote encrypted in two different envelopes, \mathcal{E}^U and \mathcal{E}^L .

THEOREM 6 (A BOOTH THAT MODIFIES MANY VOTES GETS CAUGHT WITH OVERWHELMING PROBABILITY)

The following statements hold, independent of any computational assumption:

- a) If a dishonest Booth tries to modify a vote, then his best strategy is to print a falsified layer and hope that the voter chooses the other layer, or hope that the voter doesn't verify his receipt.*
- b) If the voter chooses the falsified layer and verifies his receipt, this will constitute an irrefutable proof that the Booth is cheating.*
- c) Consequently, no matter how the Booth cheats, for each vote that the Booth falsifies and that the voter verifies, it has a chance of $\frac{1}{2}$ of being caught.*

PROOF: We can worry about various kinds of cheating. For instance, we could think that the Booth encodes the image of vote 1, but vote 2 in one of the envelopes \mathcal{E}^a , or both. Or that it can take advantage of displacing the voting image a few pixels, something that the voter might not be able to perceive.

However, all these worries are misplaced, since we can prove that a cheating Booth will be caught with probability 1/2 for each vote that it modifies, provided that the voter verifies her receipt.

The point to observe is this:

- The protocol requires that the Booth print $\mathcal{B}, q, \mathcal{E}^U$ and \mathcal{E}^L **before** it knows which layer the voter chooses.

- All these values depend on the serial number q . That is, the Booth has freedom in choosing the value of q , **but once this value is fixed, all other values are deterministic.**
- For the layer that the voter chooses, the Booth must print the value of s^c on the ballot as well, together with a digital signature on all the data. This allows anybody who has access to (a copy of) the ballot to check the integrity of the image and all the values printed on it.

In other words, given q , the Booth has no freedom whatsoever in constructing the ballot, and any bit or pixel that it modifies will be detected as fraud if the corresponding layer is chosen and somebody verifies the construction of the ballot. Observe that this property is independent of any cryptographic or computational assumption: even if the Booth has unlimited computational power this will not help him when trying to cheat.

We can therefore draw the following conclusions:

- If the Booth publishes Anne’s serial number but posts an otherwise inconsistent ballot, the Booth is caught with certainty if at least one person (not necessarily Anne) checks it.
- If the Booth tries to cheat on both layers (either by publishing a different serial number, or posting inconsistent layers using the same serial number) and the voter checks, it gets caught with certainty.
- If the Booth tries to cheat on one layer (either by publishing a different serial number, or posting inconsistent layers using the same serial number) and the voter checks, it is gets caught with probability γ , which is the smaller of β and $1 - \beta$, where β is the probability with which a voter chooses the upper layer. ◀

We can conclude therefore that a voter can verify that he created a valid vote, i.e. that Subrequirement i) of D is satisfied.

Another worry is that not all voters might take the effort to verify their receipt. If on average only a δ fraction of the voters verifies, it can be shown that the chance of catching a cheating Booth is roughly $1 - (\delta/\gamma)^N$.

We now need to show that Subrequirement iii) of D is satisfied, i.e. that the voter can be convinced with high probability that his ballot is included in the count. The reasoning here is as follows: the voter can convince himself that his receipt is posted, the posted receipts form the input batch for the mixing process, and cheating during the mixing process is almost impossible.

PROPERTY 7

If a voting receipt is not properly posted, it is physical and irrefutable evidence that the Booth is cheating.

PROOF: This is true because all receipts come with a digital signature from the Booth. ◀

THEOREM 8 *Any teller that tries to cheat by modifying a vote in the mixing process has a chance of $\frac{1}{2}$ to be caught for each modified vote. Therefore, if a teller modifies t votes, the chance of catching him is $1 - 1/2^t$.*

PROOF: This is a property of the mixing protocol, as explained in section 7. ◀

Requirement E speaks about the integrity of the ballot box. A virtual ballot box has integrity when votes cannot be added, deleted or changed, and when they cannot be associated with individual voters. Deducing the vote from the ballot is impossible because of Theorems 2, 3 and 4; the impossibility of modifying or removing it follows from Theorems 5, 6 and 8; whereas the impossibility of inserting votes is ensured by additional procedures – such as those that match the number of voters with the number receipts issued by a Booth – not addressed in this protocol.

Requirement F states that only the votes in the ballot box will be counted. This follows from Theorem 8 and the last phrase of the preceding paragraph.

Requirement G says that the tallying must be public. This is satisfied since the votes will leave the mix completely unencrypted, and anyone can verify the tally. Further, the audits are also public, so the entire process of decryption may also be public and is publicly verifiable.

Requirement H: see section 9

PROOF OF THE MAIN THEOREM: This follows easily now: Statements a) and b) follow from Theorems 2, 3 and 4, Statement c) is identical to Property 7, and Statement d) and e) follow from Theorems 6 and 8. ◀

9 Assurance in voting systems

A subtle but very important point to appreciate is the nature of the assurance of integrity provided by cryptographic, voter-verifiable schemes. Assurance of accuracy does not depend on the correct functioning of mechanisms of the correct running of code or hardware. Instead, it depends only on the correctness of the mathematical arguments and the integrity of the checks performed on the audit trail. In other words, run-time errors in the functioning of software during the vote capture and processing that might lead to violation of the accuracy requirement will, with high probability be detected. More precisely, the chance of the corruption of t votes say, falls off exponentially with t , typically roughly as $(1/2)^t$. Thus the chance of twenty votes being altered undetected is roughly one in a million.

This is in stark contrast to the type of assurance provided by other voting technologies. For example, with a touch screen device, malfunctions of the code can easily lead to essentially undetectable violations of accuracy, as the Johns Hopkins and Princeton, [32, 33] reports make clear. Similarly, with a lever machine, any malfunctions of the mechanisms, cogs etc, can lead to votes being incorrectly counted. Many, essentially undetectable ways have been identified to corrupt the functioning of such devices. No amount of verification and testing can eliminate the possibility of corrupt software being inserted or triggered during the election phase.

We should also emphasize that the assurance of accuracy provided by the schemes described here is *unconditional*. That is to say, it does not depend on assumptions as to the computational power of the adversary. Even if the adversary were assumed to have unbounded computational power and hence the ability to break the cryptographic algorithms, he would still not be able alter the tally with impunity. Of course, such an adversary could undermine the secrecy of the ballots, but that is a different matter.

All of this is not to say that we should do away with verification and testing of the system. It will be important to perform such checks with rigor if only to ensure the smooth running of the election. To reiterate the point, our guarantees of accuracy will in no way depend on these checks.

10 Conclusions

We have introduced a new paradigm in polling-place voting: voters being able to verify that their own votes are recorded as cast and that all votes recorded are tallied correctly.

Starting from the natural properties of familiar room voting, we developed a formal statement of properties that should be satisfied by any election. We then introduced the concept of the proposed election technique in a simplified though complete and potentially workable form. We extended that to the most general form of polling place automation that has been proposed. Finally, we prove that this full system satisfies all the secret-ballot and integrity properties.

References

- [1] D. L. Chaum, “Untraceable electronic mail, return address, and digital pseudonym,” *Communications of ACM*, vol. 24, pp. 84–88, February 1981.

- [2] J. D. Cohen and M. J. Fischer, “A robust and verifiable cryptographically secure election scheme,” in *Proceedings of the 26th. IEEE Symposium on Foundations of Computer Science*, pp. 372–382, 1985.
- [3] J. C. Benaloh, *Verifiable Secret Ballot Elections*. PhD thesis, Yale University, 1987.
- [4] J. Benaloh and D. Tuinstra, “Receipt-free secret-ballot elections,” in *Proceedings of the 26th ACM Symposium on Theory of Computing*, pp. 544–553, 1994.
- [5] A. Neff, “A verifiable secret shuffle and its application to e-voting,” in *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pp. 116 – 125, 2001.
- [6] M. Jakobsson, A. Juels, and R. L. Rivest, “Making mix nets robust for electronic voting by randomized partial checking,” in *Proceedings of the 11th USENIX Security Symposium*, pp. 339 – 353, 2002.
- [7] K. Sako and J. Kilian, “Receipt-free mix-type voting scheme – a practical solution to the implementation of a voting booth,” in *Advances in Cryptology – EUROCRYPT ’95*, LNCS 921, pp. 393–403, Springer-Verlag, 1995.
- [8] K. Sako and J. Kilian, “Secure voting using partially compatible homomorphisms,” in *Advances in Cryptology – CRYPTO ’94*, LNCS 839, pp. 411–424, Springer-Verlag, 1994.
- [9] R. Cramer, R. Gennaro, and B. Schoenmakers, “A secure and optimally efficient multi-authority election scheme,” in *Advances in Cryptology – EUROCRYPT ’97*, LNCS 1233, pp. 103–118, Springer-Verlag, 1997.
- [10] A. Fujioka, T. Okamoto, and K. Ohta, “A practical secret voting scheme for large scale elections,” in *Advances in Cryptology – AUSCRYPT ’92*, LNCS 718, pp. 244–251, Springer-Verlag, 1993.
- [11] T. Okamoto, “Receipt-free electronic voting schemes for large scale elections,” in *Proceedings of the 5th International Workshop on Security Protocols*, LNCS 1361, pp. 25–35, Springer-Verlag, 1997.
- [12] M. Burmester and E. Magkos, “Towards secure and practical e-elections in the new era,” in *Advances in Information Security - Secure Electronic Voting*, pp. 63–76, Kluwer Academic Publishers, 2003.
- [13] Caltech-MIT Voting Technology Project, “Report: Voting – what is, what could be.” <http://www.vote.caltech.edu/>, July 2001.
- [14] R. Greenstadt, “A bibliography on electronic voting.” <http://theory.lcs.mit.edu/~cis/voting>, 2004.
- [15] D. Gritzalis, ed., *Secure Electronic Voting*. Kluwer Academic Publishers, USA, October 2002.
- [16] D. W. Jones, “Voting and elections – tutorials.” <http://www.cs.uiowa.edu/~jones/voting/index.html>.
- [17] R. Kofler, R. Krimmer, and A. Prosser, “Electronic voting: algorithmic and implementation issues,” in *Proceedings of the 36th Hawaii International Conference on System Sciences*, 2003.
- [18] Z. Rjaskova, “Electronic Voting Schemes,” Master’s thesis, Comenius University, Bratislava, 2003.
- [19] T. Kohno, A. Stubblefield, A. D. Rubin, and D. S. Wallach, “Analysis of an electronic voting machine,” Tech. Rep. TR-2003-19, Johns Hopkins Information Security Institute, July 23 2003.
- [20] D. Chaum, “Secret-ballot receipts: True voter-verifiable elections,” *IEEE Security and Privacy*, vol. 2, pp. 38–47, January/February 2004.
- [21] C. A. Neff, “Practical high certainty intent verification for encrypted votes.” <http://www.votehere.net/vhti/documentation/vsv-2.0.3638.pdf>, 2004.
- [22] P. Ryan and J. Bryans, “A simplified version of the Chaum voting scheme,” Tech. Rep. CS-TR: 843, School of Computing Science, University of Newcastle, May 2004.
- [23] J. van de Graaf, “Adapting Chaum’s voter-verifiable election scheme to the Brazilian system,” in *Annals of IV WSeg - Workshop em Segurana de Sistemas Computacionais*, pp. 187–198, 2004.
- [24] P. Ryan, “A variant of the Chaum voter-verifiable scheme,” Tech. Rep. CS-TR: 864, School of Computing Science, University of Newcastle, October 2004.
- [25] P. L. Vora, “David Chaum’s voter verification using encrypted paper receipts.” Cryptology ePrint Archive, Report 2005/050, 2005.
- [26] M. I. Shamos, “Electronic voting - evaluating the threat,” *Computers, Freedom and Privacy*, 1993.
- [27] A. Young and M. Yung, “The dark side of black-box cryptography, or: Should we trust Capstone?,” in *Crypto’96*, pp. 89–103, Springer-Verlag, 1996.
- [28] M. Gogolewski, M. Klonowski, M. Kutylowski, P. Kubiak, A. Lauks, and F. Zagorski, “Kleptographic attacks on e-voting schemes,” in *Proceedings of the International Conference on Emerging Trends in Information and Communication Security*, LNCS 3995, pp. 494–508, Springer-Verlag, 2006.
- [29] M. Naor and A. Shamir, “Visual cryptography,” LNCS 950, pp. 1–12, Springer-Verlag, 1994.

- [30] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.
- [31] M. Gomułkiewicz, M. Klonowski, and M. Kutylowski, “Rapid mixing and security of Chaum’s visual electronic voting,” in *ESORICS '03*, LNCS 2808, pp. 132–145, Springer-Verlag, 2003.
- [32] T. Kohno, A. Stubblefield, A. D. Rubin, and D. S. Wallach, “Analysis of an electronic voting system,” in *Proceedings of the IEEE Symposium on Security and Privacy*, 2004.
- [33] A. J. Feldman, J. A. Halderman, and E. W. Felten, “Security analysis of the Diebold Accuvote-TS voting machine,” in *Proceedings of the 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'07)*, 2007.