Receipt-Free Homomorphic Elections and Write-in Ballots

Alessandro Acquisti [*]
Carnegie Mellon University

Posted November 5, 2003
Revised: May 4, 2004

Abstract

Abstract. We present a voting protocol that protects voters' privacy and achieves universal verifiability, receipt-freeness, and uncoercibility without *ad hoc* physical assumptions or procedural constraints (such as untappable channels, voting booths, smart cards, third-party randomizers, and so on). We discuss under which conditions the scheme allows voters to cast write-in ballots, and we show how it can be practically implemented through voter-verified (paper) ballots. The scheme allows voters to combine voting credentials with their chosen votes applying the homomorphic properties of certain probabilistic cryptosystems.

Keywords. Electronic Voting, Receipt-Freeness, Uncoercibility, Write-In Ballots, Voter-verified Ballots, Homomorphic Encryption, Paillier cryptosystem.

1 Introduction

Since the seminal contributions by Chaum [Cha81], Demillo, Lynch, and Merritt [DLM82], and Benaloh [Ben87], electronic voting protocols have satisfied important requirements: protecting voters' *privacy*, ensuring election *robustness*, and guaranteeing *universal verifiability* of the correctness of the election tally.

However, certain cornerstones of conventional elections have proved difficult to replicate in electronic schemes. *Receipt-freeness* and *uncoercibility* imply that no voter should be able to prove to others how she voted, and no party should be able to force another party to vote in a certain way or abstain from voting (see [BT94]). These properties have been so far guaranteed under limiting *ad hoc* physical assumptions or procedural constraints (such as untappable channels, smart cards, voting booths, third-party randomizers, and so on). *Write-in ballots* are ballots in which a voter can insert a freely chosen message - a right protected in certain legislations and jurisdictions. This ability clashes with the need to maintain receipt-freeness in universally verifiable electronic protocols. Lastly, the *security* and *accountability* of electronic schemes deployed in insecure environments (such as the Internet) have recently raised significant concerns (see [Rub02], [Riv02], [Mer02], [KSRW03], [Sha04], and [JRSW04]).

In this paper we present a voting scheme that achieves privacy, universal verifiability, receipt-freeness, and uncoercibility without *ad hoc* physical assumptions that may undermine security,

[*]I thank Lorrie Cranor, Rachna Dhamija, Sven Dietrich, Brad Malin, Adrian Perrig, and Dawn Song for useful remarks and suggestions. Originally posted on November 5, 2003 and available at http://www.heinz.cmu.edu/ acquisti/research.htm. Email for comments: acquisti@andrew.cmu.edu

flexibility, robustness, trustworthiness, or ease of use. This makes the scheme flexible and eminently practical: it can be implemented in different physical configurations, from purely electronic to mixed paper/electronic elections, in Internet voting applications as well as in physical, controlled voting kiosks. In our scheme, (theoretical) universal verifiability is accompanied by (practical) accountability, since our scheme makes it possible to have voter-verified (printed) ballots. In addition, our protocol can be used for voting scenarios with yes/no questions (such as referenda), multiple options or l *out of* t options (such as elections where voters may have to choose on several issues or submit lists of choices), as well as to cast write-in ballots under the election design conditions that we discuss below.

The scheme we propose is based on the homomorphic properties of certain probabilistic encryption protocols (see [Pai99] and [DJ01]). The homomorphic properties are applied by voters at the same time to their *credentials* (that allow voters to make their ballots count in the tally) and their votes. The election authorities provide shares of credentials to each voter, along with designated verifier proofs of each share's validity. Using homomorphic encryption, the voter assembles the shares and *combines* them with her own vote, that is cast on a public bulletin board. All messages in the bulletin board can be decrypted by a coalition of the election authorities after the voting phase of the election is completed.

In the rest of this paper we first contrast our contribution to that of related research (Section 2). We then briefly present the cryptographic building blocks of our approach (Section 3; a more detailed discussion is provided in the Appendix). In Section 4 we present the actual scheme, and in Section 5 we discuss its properties and possible attacks. In particular, we discuss receipt-freeness in Section 5.1, write-in ballots in Section 5.2, and examples of practical implementations of the scheme with voter-verified (printed) ballots in Section 6.

2 Related Work and Contributions

Three approaches dominate the electronic voting literature.

Several voting schemes are based on Chaum's mix-nets [Cha81], which through several permutations obfuscate the link between a voter and the ballot she cast (see applications in [PIK93], [Pfi95], [SK95], [MH96], [Abe98], [HS00], [MBC01], and [JJR02]).

Other schemes have applied Chaum's blind signatures protocol [Cha83]. A voter encrypts and blinds her vote before presenting it to the election authority for validation, together with her proof of eligibility for that election. After the authority validates her vote, the voter unblinds the encrypted, signed message in order to reveal a signed vote that can no longer be associated to the original encrypted message (see [Cha88], [FOO92], [HMP95], [Oka97], [CC97], and [OMA$^+$99]).

Homomorphic voting schemes (see [BT94], [Ben87], [SK94], [CFSY96], [CGS97], [HS00], [BFP$^+$01], [LK02], [DJ01], and [DJN03]) apply certain properties of probabilistic cryptosystems where correspondences can be proved to exist between operations on a certain group in the *message* space and operations on the corresponding group in the *ciphertext* space. As new and efficient cryptosystems with appealing homomorphic properties have been proposed in the literature (El Gamal [ElG84], Naccache and Stern [NS97], and Paillier [Pai99], [DJ01]), voting systems based on them have received increasing attention. In the electronic voting literature such homomorphic properties have been used most often to tally votes as *aggregates*, without decrypting single votes (thus ensuring privacy: see [CGS97]), or to combine shares of a participant's vote (see [Ben87]).

A number of heterodox approaches to electronic voting have also been proposed. [KAGN98] present a probabilistic election scheme, where robustness in the strong sense is not achieved, since election results are only valid probabilistic. [RRN01] present a variation of a blind signatures scheme, but collusion between parties can compromise receipt freeness. [MMP02] propose a protocol that is not based on conventional cryptographic primitives and achieve informational-theoretic privacy, but not receipt-freeness. [KY02] achieve perfect ballot secrecy, but not receipt-freeness.

Several of these protocols achieve *many* of the most desirable properties of an electronic voting scheme described in Section 1 (for a complete review of desirable properties, see [FOO92], [BT94], and [BM03]). However, certain important requirements have proved difficult to satisfy.

A first challenge for electronic protocols is the format of permissible ballots. Many of the most robust and practical protocols were initially designed for simple binary choices. Over time, they have been modified to support multi-candidate or l out of t selections (see [CFSY96], [CGS97], [BFP$^+$01], [DJ01], and [DJN03]), and, only more recently, to also allow write-in ballots (see for example [Nef03], based on mix-nets, [KY04], based on homomorphic encryption).

A related challenge is represented by the need to guarantee receipt-freeness and uncoercibility. With the exception of [JJ02] (that we discuss below), all electronic voting protocols have attained those properties through *ad hoc* physical assumptions and trusted third parties (see [JJ02] and [MBC01]): for example, one- or two-way untappable channels and/or anonymous or private channels (as in [Oka97], [SK95], or [HS00]); third-party (trusted) honest verifiers (as in [LK00]); smart cards and encryption black-boxes (as in [MBC01]); tamper-resistant machines (as in [LK02]); third party randomizers (as in [Hir01], [BFP$^+$01], or [KY04]); voting booths (as in [BT94] and [Nef03]) with special visual encryption tools (as in [Cha02]). Also schemes based on *deniable encryption* (such as [CDNO97]), while addressing uncoercibility, are not receipt-free, because a voter may *choose* to signal her vote to an observer through certain random bits inserted in her messages (see [HS00]). Reliance on *ad hoc* physical assumptions or trusted third parties is problematic, because it undermines the security, flexibility, robustness, trustworthiness, and ease of use of an election scheme. Write-in ballots exacerbate the difficulty of guaranteeing receipt freeness: in a randomization attack (see [JJ02]), a vote buyer or coercer can ask the voter to insert a uniquely identifiable random string inside the ballot, so that that vote can be later recognized (or abstention can be provably forced upon the voter).

Four recently proposed electronic schemes have made exciting progresses towards the goal of combining receipt-freeness with universal verifiability and, in some case, write-in ballots: [JJ02], [Cha02], [Nef03], and [KY04].

Juels and Jakobsson [JJ02] directly address the problem of achieving receipt-freeness and uncoercibility without "unpractical" assumptions. Their scheme is the most similar to the one we propose in this paper, in that it relies on a mix of authorities issuing shares of credentials that voters can use to vote (or to *fake* votes in order to cheat coercers and vote-buyers). Also similarly to our approach, Juels and Jakobsson's authorities tally an election's result by comparing a list of encrypted credentials to a list of encrypted votes. Our scheme, however, departs from [JJ02] in several respects. In terms of design, [JJ02]'s properties are achieved through mixing and blinding of credential shares. In our protocol, regardless of the use of El Gamal or Paillier encryption, the desirable properties are achieved through mixing and homomorphic encryption - in particular, our scheme is based on the novel concept of allowing the voter to cryptographically combine her own vote together with her shares of credentials. In terms of functionality, this different design generates certain advantages compared to [JJ02]'s protocol. First, it shields our protocol against a possible attack on receipt-freeness in [JJ02] that we have not seen discussed elsewhere and therefore present here. In [JJ02]'s protocol, "[the authority] removes all but one ballot sharing the same [credential.]" (p. 12) and "[the voter] includes NIZK proofs of knowledge of [the credential and the ballot]" to her message. Because proofs of correctness of credentials are first verified and then duplicate ballots (associated to valid credentials) are removed, it is possible for a coercer to force a voter to submit x times the same (valid) credential with the vote chosen by the coercer. If the coercer observes the Authority removing $x-1$ ballots sharing the same representative in the list of credential that passed the zeroknowledge test, then the coercer acquires potentially identifying information about the presence of the vote he bought or coerced in the pool of accepted votes. For comparison, in our scheme a voter could simply oblige and send several times a false credential (see Section 5.2). Furthermore, since duplicate credentials are removed "[a]ccording to some pre-determined policy, e.g., timestamps on postings to the bulletin board" (p. 12), such timestamps, coupled with the NIZK proofs of knowledge that the voter may show to the coercer, offer a form of identification of what submitted ballots have been accepted by the election authority, that a coercer may exploit to recognize the vote he wanted to buy or coerce. For comparison, while our protocol allows only one vote to be counted per valid credential, it does not place limits to the number of (real or fake) ballots the voter may submit. Second, in [JJ02]'s protocol, each voter needs to attach a zero-knowledge proof of validity for the credential and the vote she wants to use - this computational burden is not necessary for the voter in the scheme we present. Our protocol is in some sense "open-ended," in that the voter can attach whatever she wants to the shares by using homomorphic encryption. So allowing write-in ballots is straightforward. On the other side, the conformity of a voter's ballot to the listed candidate slate, proved by the voter in zeroknowledge fashion in [JJ02] (p. 11), may create difficulties to the application of Juels and Jakobsson's schemes to write-in elections. While [JJ02] do not openly discuss the possibility of write-in ballots, allowing voters to choose their ballots would leave their scheme exposed to a forced-abstention attack based on randomization. The reason is that a coercer could force a voter to associate her one only credential to a ballot containing in fact a unique, secretely chosen string, that can be recognized by the coercer after the tally but cannot be counted in the election. As discussed above, our protocol instead allows only one vote to be counted per credential, but does not place limits to the number of (real or fake) ballots the voter attempts to submit. In addition, to be counted, a vote needs to be composed of all credential shares the voter has received, as well as the vote she has chosen to cast, rather than being based on a credential generated by a threshold of authorities.

David Chaum's most recent protocol [Cha02] satisfies several important properties. In this protocol, a voter casts her ballot inside a voting station and receives a printed receipt, whose encryption is based on visual cryptography. The receipt can be tested for authenticity and its presence in the batch of ballots about to be tallied can be verified. However, since the content of the receipt is encrypted, receipt-freeness (which refers to the inability to prove to others a certain vote) is still satisfied. There are three differences between our scheme and [Cha02]'s. While [Cha02]'s scheme addresses receipt-freeness and write-in properties, it is tied to a specific physical implementation (consisting of voting stations and visual cryptography) which may constrain its adoption, unlike the scheme we propose here, that does not rely on *ad hoc* physical assumptions. In addition, Chaum's protocol is not deterministically fair, in the sense that (as noted by the author) it is possible for a voting station to change the vote cast by a voter, with a 50 per cent probability of this manipulation being detected. [Cha02] notes however that this probability makes it extremely unlikely that a malicious voting station could alter several votes. However, *robustness* then becomes a concern. To explain why, we present here a robustness attack that we have not found discussed elsewhere by a malicious voting station that wants to disrupt an election. Since a voting station's attempt to alter a vote can be detected only after the vote is cast, and since there is a 50 per cent probability that the manipulation would not be discovered, a malicious voting station should wait till near the end of the election to start manipulating votes. As it manipulates more and more votes, the probability increases that one manipulated vote will be detected, and the voting station will be exposed as a malicious party. However, now it becomes unclear what policy should be implemented with regard to votes cast at that voting station prior to that moment. A few, some, many (although with decreasing probabilities), conceivably all (with negligible probability) of the votes cast *before* that moment could have been manipulated. There is no way to detect which ones and how many. Should the voting station be replaced and all voters called back to vote again at a different station? So, although in this protocol altering the results of an election may be difficult, disrupting it could be more likely. For comparison, in the scheme we present, eventual attempts to manipulate votes can be detected with 100 per cent probability before the vote is actually cast. Hence, they can be corrected without disrupting the election. Finally, [Cha02]'s protocol may be exposed to a forced abstention attack (see below), since each voter can only submit one ballot, and the write-in ballot content, once publicly verified, may expose the unique string chosen by the coercer to make the vote invalid but identifiable. This attack is instead neutralized in our approach.

[Nef03] has proposed a new efficient voting scheme based on his shuffle mix-net protocol [Nef01]. Neff's protocol is efficient and also allows write-in ballots. However, receipt-freeness in Neff's protocol depends on procedural, physical conditions: the voter must be monitored by an election authority so that she does not bring outside the voting booth a "codebook" which confirms the "unique, publicly verifiable correspondence" between the election codes and the voter's preferences (see [NA03], p. 9, footnote 11, and [Nef03], p. 7, Step v1). If the voter succeeded in bringing the codebook out of the voting booth, she would be able to prove to another party her vote. Furthermore, procedural assumptions are also needed to prevent the voting machine to recognize whether a user is a voter or an observer - without such assumptions, cheating is possible (see [NA03], p. 9). By converse, our protocol relies on designate verifier proofs to achieve receipt-freeness, and cheating by an observer is not possible (see Sections 5 and 6).

Finally, [KY04] have recently proposed a novel vector-ballot approach that can be instantiated over any homomorphic encryption function. As in our scheme, [KY04] make it possible for the voter to cast write-in ballots. However, unlike the scheme we propose, their protocol cannot achieve receipt-freeness without *ad hoc* physical assumptions such as a randomizer (p. 5).

Most theoretical protocols have focused on the concept of universal verifiability rather than on practical, voter-verified accountability.[1] This is not surprising - researchers know that universal verifiability also implies voter-verifiability, and therefore focus on the former. However, recent critiques of electronic systems have specifically pointed to the (real or psychological) need for physical ballots and auditable trails in actual implementations of electronic elections (see [JRSW04]). Criticisms have also pointed to the need for ballots and mechanisms that a human being could easily understand (see, for example, [Mer02]). While it is not immediately evident that a *paper* ballot cast through an *electronic* voting machine would offer higher guarantees and security than purely electronic ballots (the physical ballot could get lost, be manipulated, or be destroyed on its way to the tallying authority; see also [Sha04]), and while an human's ability to understand the printed receipt of a ballot cast by an electronic machine does not alter the underlying (digital) *representation* of the ballot inside the machine (non-human readable), some of the universally verifiable schemes proposed in the literature could be easily adapted to produce printed receipts. The underlying problem, however, is to provide a receipt which satisfies receipt freeness: in other words, a (possibly printed) receipt that satisfies the voter's verification needs, but nobody else's. Satisfying this requirement, as discussed above, has so far required additional *ad hoc* assumptions.

2.1 Our Contributions

Our protocol contributes to the literature mainly by presenting a scheme which guarantees receipt freeness and uncoercibility without ad hoc physical assumptions (like those in [Cha02], [Nef03], and [KY04]) and by addressing some of the issues in [JJ02]. By this we mean that there are no additional physical components created *specifically* to ensure receipt-freeness and that could fail, be attacked, or collude with malicious agents. Obviously, the absence of such *ad hoc* physical assumptions does not imply that physical considerations become irrelevant to the security of this electronic scheme when actually implemented. No electronic voting scheme can yet be securely deployed in insecure environments such as PCs and the Internet (see [Rub02], [Riv02], [KSRW03], and [JRSW04]). However, because our protocol relies on fewer physical constraints to achieve its properties, it is more reliable, efficient, less relying on trusted equipment to avoid the risk of vote buying, coercing, and force abstention, and can be deployed in a variety of physical configurations, depending on needs and available tools (from completely electronic voting to paper-based voting; from Internet voting to voting kiosks). This is a second, practical strength of our protocol.

In addition, our protocol allows for flexible ballot formats to be used, including write-in ballots without the specific procedural constraints or physical assumptions needed in [Cha02] and [Nef03]. Granted, as in [Cha02], [Nef03], and [KY04], when write-in ballots are allowed our protocol becomes exposed to randomization attacks meant to force a voter to vote in a certain way. However, unlike [Cha02], [Nef03], and [KY04], our protocol can at least neutralize forced abstention randomization attacks (see Section 5.1). In addition, our protocol makes it straightforward to add *election design*

[1]An exception is in fact [Cha02], whose explicit goal is to produce receipts which are not verifiable by vote-buyers or coercers.

conditions (rather than *cryptographic scheme* conditions) in order to combine write-in ballots with receipt freeness (see Section 5.2).

Because receipt-freeness is guaranteed without physical constraints, the private steps in the protocol can be documented through voter-verified, even physical ballots, while the public steps can be stored (and therefore be auditable) on a public bulletin board, making the whole election verifiable. In a physical implementation through voting kiosks, for example, a voter could print out a receipt of her vote at the kiosk. Such receipt would not prove to others the choice made by the voter, but the voter could later compare it to the list of ballots about to be tallied on the bulletin board (or the election website) to make sure it is there (see Section 6).

Finally, the protocol proposes a somewhat novel voting application of the homomorphic properties of certain cryptosystems, in which shares of credentials (that allow voters to cast ballots that will be tallied) are combined by the voter to her own vote.

3 Cryptographic Primitives

In this section we briefly describe the cryptographic primitives we use in our scheme. (A much more detailed and formal discussion is presented for the interested reader in the Appendix, together with some of the proofs we apply in the protocol.)

Our election scheme is based on the homomorphic properties of probabilistic cryptosystems, and in particular the Paillier cryptosystem [Pai99].[2] The protocol applies Chaum's bulletin board and mix-nets [Cha81].

The Paillier cryptosystem is a probabilistic encryption system with two properties that we use in our election scheme: self-blinding (any Paillier ciphertext may be re-encrypted with a new random factor without altering the plaintext), and *additive* homomorphic properties (loosely speaking, product operations on a set of ciphertexts correspond to addition operations in the corresponding message space). More precisely, in the protocol we apply a threshold version of the Paillier cryptosystem (in which the private key that decrypts a group of ciphertexts can be a secret shared by several entities). We also apply [BFP$^+$01]'s proof of knowledge that two ciphertexts are encryption of the same plaintext in a designated verifier proof form (see [JSI96] and [HS00]).

A bulletin board is a public broadcast channel with memory where a party may write information that any party may read. The re-encryption mix network [Cha81] that guarantees privacy is a distributed protocol that takes as input a set of messages and returns an output consisting of the re-encrypted messages permuted according to a secret function (see Appendix).

4 A Receipt Free Electronic Voting Scheme

4.1 Voting Scheme Overview

The election Authority (or 'Authority') is composed of independent servers (or 'authorities') that supervise registration and tallying of votes through a bulletin board. Each authority creates a series of random numbers (one for each eligible voter), which represent *shares* of the voting credential a voter needs to associate to her vote in order to have it tallied. Each server posts on a bulletin board copies of the shares of credentials it creates, encrypted with a set of Paillier public parameters. Each server also provides voters with the same shares of credential, encrypted under a different set of Paillier public parameters. The server also attaches to its message a designated verifier proof of the equivalence between the encrypted share it has posted on the bulletin board and the one the voter has received. Each server also creates random numbers that are used as shares for the permissible ballots voters can cast in the election (effectively, the possible yes/no, multi-candidate, t out of l choices, or any countable set of choices that the voter can select). Those shares are also encrypted under the two different Paillier public parameters. Both resulting sets of encrypted

[2] An implementation under the El Gamal cryptosystem [ElG84] is also possible and it is sketched in the Appendix.

shares of permissible ballots are posted on the bulletin board together with zero-knowledge proofs that each pair of ciphertexts are encryptions of the same underlying share of ballot, and are then signed by the Authority.[3]

Using Paillier encryption, each voter multiplies the shares she has received from each authority together with the encrypted shares of the ballot, which she has selected from the board. Because of the homomorphic properties of Paillier cryptosystems, the resulting ciphertext includes the sum of those shares (which represents the voter's credential) and the ballot's shares (which represents her vote). The resulting ciphertext is sent to the bulletin board.

After the voting deadline expires, all ciphertexts posted by *allegedly* eligible voters are mixed by the authorities. The shares of credentials posted by the authorities are also combined (for each voter) and then mixed.

The authorities thus obtain two lists: a list of encrypted, mixed credentials the authorities themselves had originally posted on the board; and a set of encrypted, mixed sums of credentials and ballots, posted on the board by the voters. The two lists have been encrypted with different Paillier public parameters. Using threshold protocols for the corresponding sets of private keys, the authorities decrypt the elements in each list and then compare them through a simple search algorithm: for each credential they know to be valid, they seek which message (if any) cast by a voter includes such credential combined to a permissible ballot.

A simplified view of entities and flows of information in this protocol is presented in Figure 1.

4.2 Definitions and Assumptions

Formally, we define the election Authority $\mathcal{A}$ as composed of s authorities $A_1, ..., A_i, ..., A_s$. The authorities create and dispense shares of the credentials necessary to have ballots tallied, act as mixes in a mix-net, and tally the votes cast through the bulletin board $\mathcal{BB}$.

The election Authority has a list of eligible voters for which it knows or has come to know the respective public keys.[4] The l eligible voters are indicated by $v_j = v_1, ..., v_j, ..., v_l$. Their names are printed on the bulletin board $\mathcal{BB}$.

Ballot shares are defined as b_i^t, where $i = 1, ..., s$ represents the various authorities that create their shares of ballots and $t = 1, ..., T$ represents different choices for a vote: for example, a yes/no election may have $t = 2$ and $b^1 =$ "yes," $b^2 =$ "no." T can be arbitrarily large. The permissible ballots are the *sum* of the individual ballots shares across all authorities. Hence, to avoid clutter, we will refer to the actual ballots simply as $B^t = \sum_{i=1,...,s} b_i^t$.

We highlight the following assumptions. We assume that $k < s$ authorities may be corrupt (see Section 5). We define y as the number of authorities needed to decrypt a message encrypted under the threshold cryptosystem used for the election, and we assume that y authorities will collaborate. We also assume that $k < y$, that is, the number of corrupt authorities is less than the number needed to decrypt the ciphertexts. We assume that the private key of a voter remains, in fact, private.[5] Finally, we assume that an attacker cannot control *every* possible communication between the voter and an authority. A simple way to satisfy this assumption is through anonymous broadcasting (see [SA99]) or Chaum's mix-nets.[6]

4.3 The Scheme

Before the election, two sets of public/private Paillier keys are generated by $\mathcal{A}$ under the threshold cryptosystem described in A.1.2:

[3]The extension to write-in ballots is discussed in Section 5.2.

[4]It is not necessary for such public key to be the permanent or main public key of the voter. It could have been created on the fly during the initial interaction and registration with the election Authority. We discuss this further in Section 5.1

[5]We relax this assumption in Section 5.1.

[6]This assumption is not needed to ensure *privacy*, which is guaranteed even if voters' messages are not anonymous. This assumption is needed to protect voters against forced-abstention attacks. See Section 5.

- One set for the *credentials*, PK^C, SK_i^C, VK^C, and VK_i^C
- One set for the *votes*, PK^V, SK_i^V, VK^V, and VK_i^V

The two sets of keys 'C' and 'V' are based on $n_{V,C}$ RSA moduli $n_V = p_V q_V$ and $n_C = p_C q_C$ respectively, where $p_{V,C}$ and $q_{V,C}$ are large primes, as described in A.1 in the Appendix. For short, we will write $E^C()$ and $E^V()$ to represent respectively encryption with the *credentials* and *votes* public keys under the Paillier cryptosystem.[7]

$\mathcal{A}$ also generates a third set of public/private keys under the threshold version of a *non homomorphic* cryptosystem: PK^S, SK_i^S, VK^S, and VK_i^S. We will write $E^S()$ to represent encryption with this cryptosystem, which does *not* display homomorphic properties and whose domain must be bigger than the domain of the Paillier's schemes used for credentials and votes. A possible choice is an RSA in s, with $s > n_{V,C}$. All public keys are posted on $\mathcal{BB}$.

Before the election, the list of permissible ballots B^t is created. Each election authority A_i creates its own share of ballot for each of the permissible ballots, b_i^t. Each A_i encrypts b_i^t once using PK^C and appropriate secret randomization, and a second time using PK^V and appropriate secret randomization. Both resulting encrypted ballot shares (let us call them $E^C(b_i^t)$ and $E^V(b_i^t)$) are signed by the authority, that also posts a public zero-knowledge proof that $E^C(b_i^t)$ and $E^V(b_i^t)$ are encryption of the same plaintext b_i^t (the proof is described in Section B in the Appendix). All ballot shares pairs are published on $\mathcal{BB}$ on an area reserved for the permissible ballots, that clearly shows which shares have been encrypted under PK^C and which have been encrypted under PK^V, and which actual "choice" t do those shares refer to and are associated with. While the set of possible choices t may be arbitrarily large, we consider here the case where they are all known in advance. We discuss in Section 5.2 the extension to write-in ballots.

The actual election takes place in three phases:

1. Preparation
2. Voting
3. Tallying

1. Preparation

Every authority A_i in $\mathcal{A}$ creates l random numbers c, representing *shares* of credentials, for each eligible voter v_j. We represent each share as $c_{i,j}$, with $j = 1, ..., l$ for each A_i. We also want: $c_{i,j}, b_i^t < n_{V,C}/2s$.[8] For each $c_{i,j}$ it creates, A_i performs two operations: first, it encrypts $c_{i,j}$ using PK^C and appropriate secret randomization, signs the resulting ciphertext with SK_i^C, and publishes it on $\mathcal{BB}$ on a row publicly reserved for the shares of credential of voter v_j:

$$(E^C(c_{i,j}))_{SK_{A_i}} \tag{1}$$

SK_{A_i} represents the signature of authority A_i.

Second, each A_i also encrypts $c_{i,j}$ using PK^V and appropriate secret randomization, without signing it, but attaching to it a designated verifier proof P_{v_j} of equality of plaintexts $E^C(c_{i,j})$ and $E^V(c_{i,j})$ derived from Section B in the Appendix. The proof is designated to be verifiable by voter v_j, whose public key is known by or has been revealed to the Authority (see Section 4.2 and Section

[7]The $E^C()$ notation is preferred to the more traditional $E_C()$, as it makes the overall description clearer in the rest of the scheme.

[8]We do not want the sum of s credential and ballot shares to be larger than the cryptosystem's domain. Possible attacks by malicious servers that attempt to create longer shares can be detected by simple observation of the size of the encrypted share, or by asking the authorities to attach proofs that an encrypted messages lies in a given set of messages - see [BFP+01].

5.1 for a discussion of the receipt-freeness property associated with this design). Each A_i encrypts this second message with v_j's public key and sends it v_j without signing it:

$$E^{v_j}(E^V(c_{i,j}), P_{v_j}) \tag{2}$$

E^{v_j} represents RSA encryption under v_j's public key. The reserved area of $\mathcal{BB}$ can be imagined as a table l by s: one row for each eligible voter and s encrypted shares of credentials on each row.

2. Voting

For each encrypted share of credential she receives, a voter v_j verifies the designated verifier proof of equality between $E^V(c_{i,j})$ and the corresponding $E^C(c_{i,j})$ that has been signed and published in her reserved area of $\mathcal{BB}$. Upon successful verification, she multiplies together the shares $E^V(c_{i,j})$:

$$\prod_{j=j,i=1,\ldots,s}(E^V(c_{i,j})) = E^V(\sum_{j=j,i=1,\ldots,s} c_{i,j}) \equiv E^V(C_J) \tag{3}$$

where with C_j we define the sum, mod n^2, of the various shares of credentials in the hands of the voter.

The voter then chooses the ballot shares $E^V(b_1^t), \ldots, E^V(b_s^t)$ (encrypted with the *votes* key) which correspond to her vote choice t from the list of permissible ballot published on the board, multiplies all the encrypted shares together in order to obtain the encrypted ballot B^t (thanks to the additive homomorphic properties of the Paillier's cryptosystem), that we will define as $E^V(B_j^t)$. Finally, the voter multiplies the resulting ciphertext times $E^V(C_j)$, obtaining:

$$E^V(C_J)E^V(B_j^t) = E^V(\sum_{i=1,\ldots,s} c_{i,j} + \sum_{i=1,\ldots,s} b_{i,j}^t) \equiv E^V(C_J + B_J^t) \tag{4}$$

With $(C_J + B_J^t)$ we define the sum, mod n^2, of the various shares of credential voter v_j received, *plus* the (sum of the shares of the) chosen ballot. The voter wraps the resulting ciphertext with the non-homomorphic RSA public key, PK^S, and sends $E^S(E^V(C_J + B_J^t))$ to the bulletin board. For short, we will loosely refer to a message $E^S(E^V(C_J + B_J^t))$ as the voter's *vote*.

3. Tallying

When the voting deadline is met, the election Authority signs the bulletin board,[9] $\mathcal{A}$ multiplies together the shares $E^C(c_{i,j})$ for each voter v_j (similarly to what each voter has done with the shares she received):

$$\forall j, \prod_{i=1,\ldots,s}(E^C(c_{i,j})) = E^C(\sum_{i=1,\ldots,s} c_{i,j}) \equiv E^C(C_J) \tag{5}$$

Then, it mixes all $E^C(C_J)$, for $J = 1, \ldots, l$, by re-encrypting (and self-blinding) the original ciphertexts using the *credentials* public parameters, PK^C.

Separately, $\mathcal{A}$ decrypts the $E^S(E^V(C_J + B_J^t))$, with $J = 1, \ldots l, \ldots x$ which have been posted on the bulletin board by allegedly eligible voters using the threshold *non-homomorphic* cryptosystem keys SK_i^S, VK^S, and VK_i^S. The number of ballots could be $x > l$ because some eligible or ineligible voters may have used invalid credentials or re-used valid ones. $\mathcal{A}$ then mixes the resulting ciphertexts, by re-encrypting (and self-blinding) the original ciphertexts using the *votes* public parameters, PK^V.

The election Authority thus obtains two lists: a list of encrypted, mixed credentials posted on the board by the authorities, $E^C(C_{\phi(J)})$; and a set of encrypted, mixed sums of credentials and ballots, posted on the board by the allegedly eligible voters, $E^V(C_{\phi(J)} + B_{\phi(J)}^t)$. The $\phi()$ operation refers to the mixing described in Section A.3, which hides the relation between a given J and the mixed ciphertexts.

[9]The content of the board could actually be signed at regular intervals, so that voters can verify their messages have been received by the Authority. See Section 6.

The election Authority also selects each ballot choice t at a time and multiplies together all the associated encrypted shares $E^C(b_1^t), ..., E^C(b_s^t)$ (encrypted with the *credentials* key) in order to obtain the encrypted ballots $E^C(B^t)$.

Under the threshold cryptosystem described in A.1.2, $y < s$ authorities may decrypt all elements in both lists using the respective private keys. Ballots cast with eligible credentials are retrieved by an algorithm that compares the $E^V(C_{\phi(J)} + B^t_{\phi(J)})$'s list on one side, and the $E^C(C_{\phi(J)})$'s list together with the list of permissible ballots B^t on the other side.

More precisely, recall that $D(E^V(C_{\phi(J)} + B^t_{\phi(J)}) \mod n_V^2) = \sum_{i=1,...,s} c_i + \sum_{i=1,...,s} b_i^t \bmod n_V$, and that $D(E^C(C_{\phi(J)}) \mod n_C^2) = \sum_{i=1,...,s}(c_i) \bmod n_C$ (see A.1.1). Then the search algorithm for the election authority involves:

1. Choosing a credential $C_{\phi(J)}$ from the still encrypted list $E^C(C_{\phi(J)})$.
2. For each credential $C_{\phi(J)}$, choosing a still encrypted ballot $E^C(B^t)$ from the list of all possible permissible ballots encrypted with the *credentials* key, and test whether: $D(E^C(C_{\phi(J)})E^C(B^t) \mod n_C^2) \equiv \sum_{i=1,...,s} c_i^C + \sum_{i=1,...,s} b_i^t \mod n_C = D(E^V(C_{\phi(J)} + B^t_{\phi(J)}) \mod n_V^2) \equiv \sum_{i=1,...,s} c_i^V + \sum_{i=1,...,s} b_i^t \mod n_V$
3. If in step 2 a match is found, a vote for t is counted in the tally, the credential $C_{\phi(J)}$ is removed from the list of valid credentials, and the algorithm restarts from 1. If in step 2 no match is found, the algorithm restarts from 2 with a different $E^C(B^t)$.
4. When all credentials $C_{\phi(J)}$ have been considered, the tallying is complete.

5 Properties and Attacks

We discuss in this section properties and possible attacks on the scheme, with particular attention to write-in and voter-verified ballots properties.

Privacy. Voter's privacy is achieved through the use of Paillier's cryptosystem and the mixing of voters' credentials $E^C(C_J)$ (on the board) and cast ballots $E^V(C_J + B_J^t)$. Privacy is preserved also when ballots are not cast anonymously, because no single authority actually sees all shares composing a voter's credential. As in other election schemes based on threshold homomorphic cryptosystems (for example, [BFP+01]), we assume that $k < y$ - that is, collusion between k malicious authorities does not meet the threshold necessary to decrypt the unmixed credentials $E^C(C_J)$ originally posted on the board, decrypt the submitted ballots, $E^V(C_J + B_J)$, and then compare the two sets.

Correctness. Only one vote per credential can be counted through the search algorithm described in Section 4.3. The Authority cannot infer any credential, and credentials $E^C(C_{\phi(J)})$ are decrypted by threshold cryptography only in combinations with encrypted ballots $E^V(B^t)$. Hence, the Authority cannot vote on behalf of the voter (the credential shares posted on the board are encrypted with different Paillier's parameters than the shares of credentials used to vote and decrypt votes). Moreover, ballots cast on the board by voters are wrapped by non-homomorphic encryption to avoid attacks in which a credential is re-used.

A replay attack of a cast ballot by the voter or a third party duplicates the same ballot and does not affect the search algorithm, since only one vote can be counted for each valid credential. An attempt by a voter to re-use a credential for multiple votes is ineffective - each valid credential $E^C(C_J)$ can only be counted once.

Credentials and permissible ballots are chosen of necessary length to make negligible the probability that two different cast ballots will collide. Assuming that even just one authority does choose random k bit numbers for the shares it produces, the probability of a collision with l credentials being created and l ballots cast is $\sqrt{\frac{2l}{2^k}}$.

Verifiability and Transparency. All communications exchanged during the protocol can appear on the bulletin board.[10] Voters can verify that the credential shares they received, $E^V(c_{i,j})$, correspond to the signed and encrypted shares on the board, $E^C(c_{i,j})$. Voters can also verify that their cast votes $E^V(C_J + B^t_J)$ do appear on the list of votes the Authority will mix and then search for valid credentials. Since the mixing, decrypting, and searching steps are publicly stored on $\mathcal{BB}$, the scheme is transparent and universally verifiable (we discuss the concept of voter-verified ballots in Section 6). Moreover, all votes are decrypted, rather than only their sum (as in traditional homomorphic voting schemes).

Ease of use. A voter casts her ballot by posting one message $E^S(E^V(C_J + B^t_J))$ to the board $\mathcal{BB}$, with no need to provide proofs of validity of the cast ballot.

Robustness. An authority that does not post and sign its credential shares $E^C(c_{i,j})$ on $\mathcal{BB}$ can be detected and replaced before the voting phase begins.

An authority that does not provide voters with the encrypted credential shares $E^V(c_{i,j})$ and appropriate designated verifier proofs can also be detected and replaced: a voter who does not receive (or claim not to have received) a credential's share appearing on $\mathcal{BB}$ can protest before the voting phase begins. If after repeated protests the voter still claims it has not received a share, a new authority may be selected to send the share. If the problem persists after a certain number of authorities have been replaced, the Authority may have to conclude that the voter is lying.

It is not required that all registered voters actually vote for the election to be completed. After the election deadline has been met, the tallying phase can simply commence. However, a voter who cannot see her ballot on $\mathcal{BB}$ after she sent it, can re-send it and/or notify the Authority before the tallying phase begins.

Casting $E^S(E^V(C_J + B^t_J))$ to the bulletin board does not need to be anonymous (votes are only revealed after credentials have been mixed). Hence, voter's verification may be adopted to avoid denial-of-service (DoS) attacks. However, it would be preferable not to let a coercer know how many ballots a certain voter has cast to the board (see Section 5.1 below). Anonymous broadcasting could therefore be applied (see [SA99]), combined with alternative strategies to avoid DoS depending on the actual implementation of the scheme.

Complexity Analysis. In terms of communication complexity, the voter's burden is limited to one message. In terms of computation analysis, however, the Authority needs to perform two mixes and several (at least $2l$) Paillier decryptions, as well as at least l RSA decryptions.

Using mix-net protocols such as those proposed by [FS01] and [Nef01], we can reduce the number of exponentiations needed to perform the mixes to a factor of $18 * 2l$.[11] On the other hand, the mix-net calculations on credentials and vote+credentials can be done simultaneously, and the search algorithm is a simple linear search. Furthermore, the computational complexity for voters is attenuated as voters do not need to prove their ballots are correct (unlike in other homomorphic protocols such as [CGS97]). Finally, while receipt-freeness is built in our scheme, in other protocols (such as [BFP$^+$01]) it relies on additional physical assumptions that insert additional computational complexities.

5.1 Receipt Freeness and Uncoercibility

Since the authorities use designated verifier proofs to demonstrate the equivalence between $E^V(c_{i,j})$ and the corresponding $E^C(c_{i,j})$, a coercer (or vote-buyer) could not be convinced that the credential a voter is showing has not been fabricated. When fake credentials are created and submitted

[10] Also communications containing the designated verifier proof could appear on the board, unsigned, together with other invalid proves designed to cheat coercers: see 'Receipt Freeness and Uncoercibility' below.

[11] [Gro03] has recently proposed an efficient calculation for mix-nets based on homomorphic re-encryption.

together with proper ballots, a coercer or buyer could never know if the credential they have received or the vote they have seen the voter casting are truly correct (neither the voter nor the Authority know the actual credentials at the time of voting, and after the mixing phase no voter can no longer recognize the ballot they cast). Note also that the credentials are never really decrypted alone, but always summed to permissible ballots - which in turn are also never decrypted alone and generated in a distributed manner.

A coercer could collude with some authorities to verify share of credentials received from the voter. In this case, similarly to [JJ02], if the voter knows at least one honest authority, it can show to a coercer the actual shares for all other authorities, and a fake credential for the authority she knows to be honest. This makes ineffective the 'randomization attack' by Schoenmakers (as described in [JJ02]). In this attack, an attacker forces a voter to cast an irregular vote made up with an agreed random sequence, thus nullifying it in a verifiable way. In our scheme, the decryption phase does not reveal all the information about ballots that have *not* been counted because not matching any permissible ballot B_t. In fact, the actual ballots are not decrypted themselves - rather, only the sum of ballots and credentials is decrypted, thus limiting the information an observer can access. In addition, a forced abstention attack (where an attacker forces a voter not to cast any vote, see [JJ02]) is avoided if votes can be cast anonymously (in which case a coercer cannot know whether the voter has voted), or if the Authority does not reveal to the coercer the identities of voters who cast ballots (recall that while our protocol allows only one vote to be counted per valid credential, it does not place limits to the number of real or fake ballots the voter may submit - hence the voter could please the coerce by adding the unique string to a fake credential and then keep on voting with her own true credential).

Apparently, an adversary able to control a voter or obtain her private key is akin to the 'demon attack' described in [JSI96] and may verifiably coerce the voter to his wishes. Effectively, in our protocol a "private key" is any means through which the voter can identify herself to the authority and the authority can communicate privately with the voter. The scheme allows a voter to cheat the coercer or vote-buyer by revealing a fake private key - there is no need for the voter to truthfully reveal her key to a malicious party.

A first way show is relies on the Authority and the voter knowing one piece of information that the coercer cannot independently verify (or many pieces, whose knowledge is distributed across the authorities). The public/private key-rings the voter chooses to use in the election can be made dependent on these pieces of information. During the registration phase, the Authority can verify the identity of the voter by checking that information before the keys are exchanged.

Alternatively, a better way relies on the voter and the Authority simply *starting* the registration phase with the known public key of the voter, but then exchange new keys in a way that an external observer could believe a proof about which keys have been exchanged. For example, after using her known public/private keys to establish communication with the Authority, the voter can send the latter a new, temporary public key. The Authority answers by sending the designated verifier proof described in Section 4.3 on the basis of the *new* key it has just received. Note that both now and in the scheme we have described in 4.3 the Authority *never* sign the message containing the designated verifier proof. This means that anybody could have sent such message - and, in fact, a smart voter could create a message to herself based on a different public key than the temporary one she has actually communicated to the Authority in the "handshaking" protocol we are describing, in order to cheat a coercer.

That is why $E^{v_j}(E^V(c_{i,j}), P_{v_j})$ (that is sent through a tappable channel) and $E^S(E^V(C_j + B_j^t))$ (that is published on the bulletin board) are not a receipt. The coercer may order the voter to reveal how to create a vote $E^S(E^V(C_j + B_j^t))$ that is compatible with the receipt $E^{v_j}(E^V(c_{i,j}), P_{v_j})$ and $E^S(E^V(C_j + B_j^t))$, but the voter could use a fake credential and a fake designated verified proof built on a different temporary public key than the one she created on the fly while communicating with the Authority (after having created a fake message purportedly from the Authority itself to the voter).

So, even if a coercer asked the voter to provide a decrypted zero-knowledge proof, it could not be sure that the proof actually corresponds to the *right* credential and private key.

5.2 Write-in Ballots

In the proposed scheme, permissible ballots B^t are sums of random numbers representing possible choices: yes/no, multiple candidates, l out of t, and any countable set. B^t is, in other words, open-ended. For a vote to be detected and counted by the search algorithm, however, it must be in the list of permissible ballots. Write-in ballots can be implemented by having voters send, together with or separately from their encrypted ballots $E^S(E^V(C_J + B^t_J))$, also suggested ballot or ballots B^j.[12] Once the proposed ballots have been added to the list of permissible ballots, the election authority will include them in the search process described in Section 4.3.

Obviously, unconditionally write-in ballots clash with the properties of receipt-freeness and universally verifiability. Consider a randomization attack: a voter inserts random and therefore uniquely identifiable data into her ballot, in order to signal her vote to the buyer or coercer. This means that as in [Cha02], [Nef03], and [KY04], when write-in ballots are allowed our protocol becomes exposed to randomization attacks meant to coerce a voter to vote in a certain way. However, unlike [Cha02], [Nef03], and [KY04], our protocol can at least neutralize forced abstention randomization attacks (for the same reasons discussed above Section 5.1).

In addition, our protocol makes it straightforward to add *election design* conditions (rather than *cryptographic scheme* conditions) in order to combine write-in ballots with receipt freeness.

In particular, the cryptographic scheme can be combined with an *election design* modelled after real elections. Many elections consider void ballots which contain obscenities, random elements, and other inappropriate material - depending on what is specified in the election's regulations. Since suggested ballots have to be submitted by voters, design guidelines can specify the format that will make those suggestions admissible. The format should be designed to minimize the opportunities for a cheater to include unrelated information inside the ballot to make it identifiable. For example, the content may be no longer than x characters. Or, for a U.S. presidential election, proposed ballots could only contain names of U.S. citizens eligible to become president. Or, the number of multiple questions in the same ballot should be limited and calibrated so that particular combinations of answers are not likely to be unique. In this way an electronic election becomes closer to the process of traditional write-in elections.

These "short" write-ins are only proposed as an election design condition. The cryptographic design makes alternative designs possible: open-ended write-ins (which may or may not be publicly decrypted to preserve receipt-freeness), or a combination of a countable ballot part (to be tallied automatically, and with universal verifiability), and a non countable, write-in part to be decrypted separately are also entirely possible options.

More precisely, it may be possible to split the non write-in and the write-in votes. Imagine that the election design allows voters to cast *both* a non write-in part (e.g., n selections from a list of candidates) as well as a write-in part (e.g., $n-1$ selections from a list of candidate, as well as a write-in candidate). Then at registration time the voter could opt to receive only one credential (valid for the n vote) or two separate credentials (one valid only for a $n-1$ vote and the other valid only for a write-in vote). The three credentials are then used exactly as in the scheme above - only, now, in a sense, three schemes are run in parallel, each scheme for a set of credentials type (n, $n-1$, and write-in). The three types of credentials would be encrypted under different parameters so that they cannot be mixed. Note that if there is no initial relation between the $n-1$ list credential and the write-in credential given to the same voter, then also the relation between the $n-1$ choice and the write-in choice by the same voter cannot be detected after the mixing phase.[13]

[12] If messages can be sent anonymously (see Section 4.2), privacy is trivially guaranteed. But since B^j is *not necessarily* the ballot the voter is actually casting inside $E^S(E^V(C_J + B^t_J))$, these messages could also be sent when the communication is not anonymous. A voter could actually choose to vote with an already published permissible ballot, and attach to her message a fake one. In addition, proposed ballots could also be submitted separately through a mix-net.

[13] We are grateful to an anonymous referee for this suggestion.

6 Implementation and Voter-Verified Ballots

It is beyond the scope of this paper to discuss physical implementations of its scheme, although we have mentioned that the protocol may be instantiated in different physical configurations - from Internet voting to voting kiosks.

It is by now widely accepted that no existing voting protocol can yet be securely deployed in an insecure environment, such as one made of PC voting stations communicating via the Internet. In this section, therefore, we only comment on two conditions necessary and sufficient to detect certain types of election corruptions and attacks on implementations of the scheme. We then discuss the possibility of voter-verified ballots.

We consider an attacker that can control k authorities, the machine from which the voter is connecting to the Authority, and monitor all network traffic. Attempts by such attacker to interfere with the voting process may be detected if the voter could independently know the real signature of the Authority (thereby being able to verify the shares it receives and the content of the bulletin board), and had access to a secure machine for all cryptographic computations (thereby being able to verify the Authority's signature and the correct encryption of the voter's own messages).

Because our protocol does not rely on *ad hoc* physical assumptions, these two conditions could be satisfied in several implementations, including voting kiosks approaches that may guarantee secure computing and verification of the Authority's signature. In such implementation, the kiosk may also print out a receipt containing the message - $E^S(E^V(C_J + B^t_J))$ - that the voter wants to see (and be counted) on the board. This printed receipt would not be directly humanly readable (although it may also be presented in a way that it is). It is conceptually similar to the receipt in [Cha02], in that the voter can verify that the code matches the code later appearing on the public board (or election web site). This verifiability, however, is stronger than that guaranteed by [Cha02]'s receipts, as well as by paper-only approaches, in that the voter can first verify that the encryption is correct on any secure machine (in fact, a voter could forecast what her vote should look like even before entering the kiosk and voting, if she had already chosen the randomizing encryption component and the ballot), and thereafter can recognize the presence of her ballot in the list of submitted ballots (which is not possible with paper-based ballots). Even in a kiosk application, the protocol does not lose its receipt-freeness: a voter could create and submit several fake ballots while in the kiosk, and take the respective receipts out of the kiosk inside order to cheat a coercer. Different hybrid combinations of electronic and paper ballots and receipts are therefore possible, making the scheme practical.

7 Concluding Remarks

We have presented a novel electronic voting scheme that achieves privacy, robustness, universal verifiability, ease of use, and receipt freeness. The scheme can be used to cast different types of ballots that include yes/no, multi-candidate, l out of t choices, as well as write-in ballots.

Our protocol contributes to the literature by presenting a scheme which guarantees receipt freeness and uncoercibility without *ad hoc* physical assumptions and by addressing some of the issues in existing, comparable protocols. Because our protocol relies on fewer physical constraints to achieve its properties, it is more reliable, efficient, less relying on trusted equipment to avoid the risk of vote buying, coercing, and force abstention, and can be deployed in a variety of physical configurations, depending on needs and available tools (from completely electronic voting to paper-based voting; from Internet voting to voting kiosks), which is a second, practical strength of this protocol.

In addition, our protocol allows for flexible ballot formats to be used in receipt-freeness and universal verifiability elections.

Because receipt-freeness is guaranteed without physical constraints, the private steps in the protocol can be documented through voter-verified, even physical ballots, while the public steps can be stored (and therefore be auditable) on a public bulletin board, making the whole election verifiable.

Finally, the protocol proposes a somewhat novel voting application of the homomorphic properties of certain cryptosystems, in which shares of credentials (that allow voters to cast ballots that will be tallied) are combined by the voter to her own vote.

References

[Abe98] Masayuki Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. In *EUROCRYPT '98*, pages 437–447. Springer-Verlag, LNCS 1403, 1998.

[Ben87] Josh C. Benaloh. Verifiable secret-ballot elections. PhD Thesis, Yale University, Department of Computer Science, 1987. Number 561.

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

[BM03] Mike Burmester and Emmanouil Magkos. Towards secure and practical e-elections in the new era. In *Advances in Information Security - Secure Electronic Voting*, pages 63–76. Kluwer Academic Publishers, 2003.

[BT94] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections. In *STOC '94*, pages 544–553, 1994.

[CC97] Lorrie Cranor and Ron Cytron. Sensus: A security-conscious electronic polling system for the Internet. In *Proceedings of the Hawaii International Conference on System Sciences*, 1997.

[CDNO97] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In *CRYPTO '97*, pages 90–104. Springer-Verlag, LNCS 1294, 1997.

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

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *EUROCRYPT '97*, pages 103–118. Springer-Verlag, LNCS 1233, 1997.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[Cha83] David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology - Crypto '82*, pages 199–203. Plenum Press, 1983.

[Cha88] David Chaum. Elections with unconditionally- secret ballots and disruption equivalent to breaking rsa. In *EUROCRYPT '98*, pages 177–182. Springer-Verlag, LNCS 330, 1988.

[Cha02] David Chaum. Secret-ballot receipts and transparent integrity. Draft, 2002. `www.vreceipt.com/article.pdf`.

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography '01*, pages 119–136. Springer-Verlag, LNCS 1992, 2001.

[DJN03] Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. A generalization of paillier's public-key system with applications to electronic voting, 2003.

[DLM82] Richard DeMillo, Nancy Lynch, and Michael J. Merritt. Cryptographic protocols. In *Proceedings of the 14th Annual Symposium on the Theory of Computing*, pages 383–400, 1982.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO '84*, pages 10–18. Springer-Verlag, LNCS 196, 1984.

[FOO92] Atshushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *Auscrypt '92*, pages 244–251. Springer-Verlag, LNCS 718, 1992.

[FPS00] Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In *Financial Cryptography '00*. Springer-Verlag, 2000.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO '86*, pages 186–194. Springer-Verlag, LNCS 263, 1987.

[FS01] Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *CRYPTO '01*, pages 368–387. Springer-Verlag, LNCS 2139, 2001.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and Systems Science*, 28:270–299, 1984.

[Gro03] Jens Groth. A verifiable secret shuffle of homomorphic encryptions. In *PKC '03*, pages 145–160. Springer-Verlag, LNCS 2567, 2003.

[Hir01] Martin Hirt. Multi-party computation: Efficient protocols, general adversaries, and voting. PhD Thesis, ETH Zurich, 2001.

[HMP95] Patrick Horster, Markus Michels, and Holger Petersen. Blind multisignature schemes and their relevance to electronic voting. In *Proc. 11th Annual Computer Security Applications Conference*, pages 149–156. IEEE Press, 1995.

[HS00] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *EUROCRYPT '00*, pages 539–556. Springer-Verlag, LNCS 1807, 2000.

[JJ02] Ari Juels and Markus Jakobsson. Coercion-resistant electronic elections, 2002. `citeseer.nj.nec.com/555869.html`.

[JJR02] Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In D. Boneh, editor, *USENIX '02*, pages 339–353, 2002.

[JRSW04] David Jefferson, Aviel D. Rubin, Barbara Simons, and David Wagner. A security analysis of the secure electronic registration and voting experiment (SERVE). Technical report, 2004.

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT '96*, pages 143–154. Springer-Verlag, LNCS 1070, 1996.

[KAGN98] Hiroaki Kikuchiy, Jin Akiyamaz, Howard Gobioff, and Gisaku Nakamuraz. stochastic voting protocol to protect voters privacy, 1998.

[KSRW03] Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, and Dan S. Wallach. Analysis of an electronic voting system. Johns Hopkins Information Security Institute Technical Report TR-2003-19, 2003.

[KY02] Aggelos Kiayias and Moti Yung. Self-tallying elections and perfect ballot secrecy. In *PKC '02*, pages 141–158. Springer-Verlag, LNCS 2274, 20002.

[KY04] Aggelos Kiayias1 and Moti Yung. The vector-ballot e-voting approach, 2004. Mimeo, University of Connecticut and Columbia University.

[LK00] Byoungcheon Lee and Kwangjo Kim. Receipt-free electronic voting through collaboration of voter and honest verifier, 2000. `citeseer.nj.nec.com/lee00receiptfree.html`.

[LK02] Byoungcheon Lee and Kwangjo Kim. Receipt-free electronic voting scheme with a tamper-resistant randomizer. In *ICISC2002*, pages 405–422, 2002.

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

[Mer02] Rebecca Mercuri. A better ballot box? *IEEE Spectrum*, 39:46–50, 2002.

[MH96] Markus Michels and Patrick Horster. Some remarks on a receipt-free and universally verifiable mix-type voting scheme. In *ASIACRYPT '94*, pages 125–132. Springer-Verlag, LNCS 1163, 1996.

[MMP02] Dahlia Malkhi, Ofer Margo, and Elan Pavlov. E-voting without 'cryptography'. In *Financial Cryptography '02*, 2002.

[NA03] Andrew Neff and Jim Adler. Verifiable e-voting, 2003. `http://www.votehere.net/`.

[Nef01] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 116–125. ACM Press, 2001.

[Nef03] Andrew Neff. Detecting malicious poll site voting clients, 2003. `http://www.votehere.net/`.

[NS97] David Naccache and Jacques Stern. A new public-key cryptosystem. In *EUROCRYPT '97*, pages 27–36. Springer-Verlag, LNCS 1233, 1997.

[Oka97] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Security Protocols Workshop*, pages 25–35. Springer-Verlag, LNCS 1361, 1997.

[OMA$^+$99] Miyako Ohkubo, Fumiaki Miura, Masayuki Abe, Atsushi Fujioka, and Tatsuaki Okamoto. An improvement on a practical secret voting scheme. In *ISW '99*, pages 225–234, 1999.

[OU98] Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In *EUROCRYPT '98*, pages 308–318. Springer-Verlag, LNCS 1403, 1998.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *EUROCRYPT '99*, pages 223–238. Springer-Verlag, LNCS 1592, 1999.

[Pfi95] Birgit Pfitzmann. Breaking an efficient anonymous channel. In *EUROCRYPT '94*, pages 332–340. Springer-Verlag, LNCS 950, 1995.

[PIK93] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. All-nothing election scheme and anonymous channel. In *EUROCRYPT '93*, pages 248–259. Springer-Verlag, LNCS 765, 1993.

[Riv02] Ronald L. Rivest. Electronic voting. In *Financial Cryptography '01*, pages 243–268. Springer-Verlag, LNCS 2339, 2002.

[RRN01] Indrajit Ray, Indrakshi Ray, and Natarajan Narasimhamurthi. An anonymous electronic voting protocol for voting over the internet. In *Proceedings of the Third International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems*, 2001.

[Rub02] Avi Rubin. Security considerations for remote electronic voting. *Communications of the ACM*, 45:39–44, 2002.

[SA99] Frank Stajano and Ross J. Anderson. The cocaine auction protocol: On the power of anonymous broadcast. In *Information Hiding Workshop*, pages 434–447. Springer Verlag, LNCS 1768, 1999.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[Sha04] Michael Ian Shamos. Paper v. electronic voting records - an assessment, 2004. Mimeo, Carnegie Mellon University.

[SK94] Kazue Sako and Joe Kilian. Secure voting using partial compatible homomorphisms. In *CRYPTO '94*, pages 248–259. Springer-Verlag, LNCS 839, 1994.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme. In *EUROCRYPT '95*, pages 393–403. Springer-Verlag, LNCS 921, 1995.

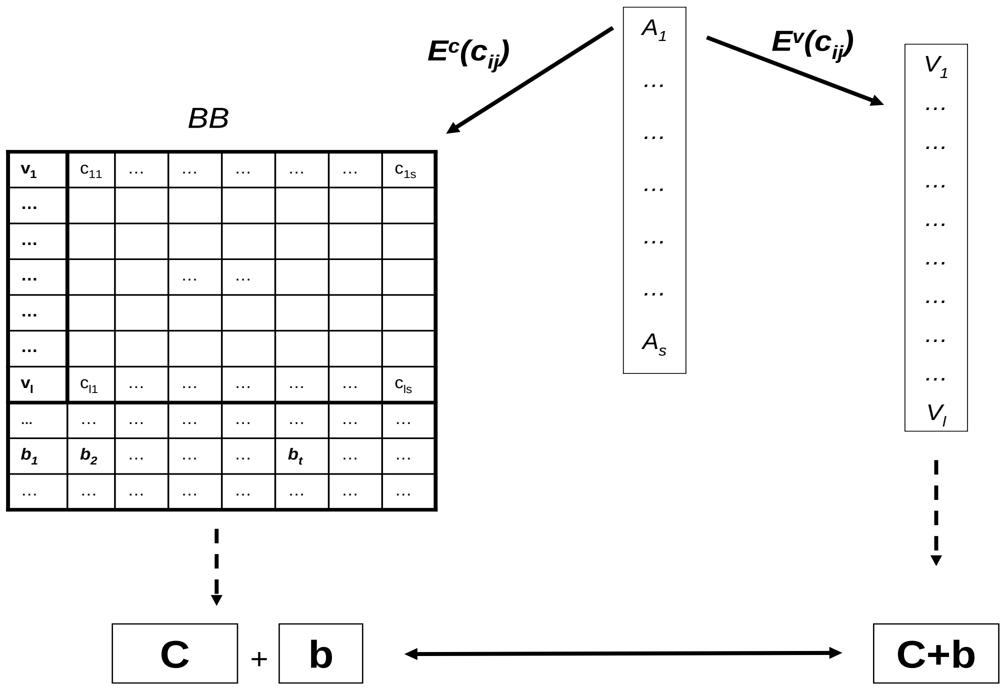

Figure 1: Simplified conceptual view of the entities and information flows composing the election scheme.

A Building Blocks

A.1 Paillier Cryptosystem

Since the work of [GM84], several probabilistic encryption schemes have been proposed in the literature (see [OU98], [NS97], [Pai99], and [DJ01]).

The security of these schemes is based on the difficulty of various types of 'residuosity' problems. A message m is encrypted with a public component and a randomizing component. Only a 'trapdoor' allows the owner of a private key to decrypt the ciphertext without knowing the randomizing component.

The trapdoor discrete logarithm mechanism that we use in our voting scheme is Paillier's (see [Pai99]). It can be described in the following way: set n to be an RSA modulus $n = pq$, where p and q are large primes. Let g be some element of $\mathbb{Z}^*_{n^2}$. Let $\lambda(n) = \mathrm{lcm}((q-1)(p-1))$. Let the set $\mathcal{S}_n = u > n^2 | u = 1 \bmod n$ be a multiplicative subgroup of integers modulo n^2 so that a function $\mathrm{L}(u) = (u-1)/n$ is clearly defined $\forall u \in \mathcal{S}_n$. Then (n, g) will be the public parameters and $\lambda(n)$ the private one. To encrypt a plaintext $m < n$, one needs to choose a random $r < n$ and calculate:

$$c = g^m r^n \bmod n^2. \tag{6}$$

To decrypt, one needs to compute:

$$m = \frac{\mathrm{L}(c^\lambda \bmod n^2)}{\mathrm{L}(g^\lambda \bmod n^2)} \bmod n. \tag{7}$$

A.1.1 Self-Blinding and Homomorphic Encryption

The Paillier cryptosystem displays two properties that we use in our election scheme.

First, any Paillier ciphertext may be re-encrypted with a new random r without altering the plaintext. More formally,

$$D(E(m)g^{nr} \bmod n^2) = m \tag{8}$$

for $\forall m \in \mathbb{Z}_n$.

Second, it has additive homomorphic properties. Two encryption functions $m \rightarrow g^m r^n \bmod n^2$ and $m \rightarrow g^{m+nr} r^n \bmod n^2$ are additively homomorphic on $\mathbb{Z}_n$, which means that, for example:

$$D(E(m_1)E(m_2) \bmod n^2) = m_1 + m_2 \bmod n \tag{9}$$

and

$$D(E(m_1)^m_2 \bmod n^2) = m_1 m_2 \bmod n \tag{10}$$

for $\forall m_1, m_2 \in \mathbb{Z}_n$ (see [Pai99]).

Below we denote applications of the second case informally as: $E(m_1)E(m_2) = E(m_1 + m_2)$.

A.1.2 Threshold Paillier Cryptosystems

In *threshold* cryptosystems, the private key that decrypts a group of ciphertexts can be a secret shared by several entities, so that the collaboration between a sub-set of them is necessary for decryption. This feature, often adopted in electronic voting schemes, has been implemented in the Paillier cryptosystem by Baudron, Fouque, Pointcheval, Poupard and Stern [BFP$^+$01] and Fouque, Poupard, and Stern [FPS00].

We use [BFP$^+$01]'s threshold decryption model, in which different entities use a distributed key generation algorithm such as [Sha79]'s to create a public key PK, secret shares SK$_i$, and the verification keys VK, VK$_i$. Anybody can encrypt a message using the public key, but to decrypt a ciphertext c a combiner must send c to the servers, who use SK$_i$, VK, and VK$_i$ to output partial decrypted shares c_i. The combiner recovers the original plaintext only if a sufficient number of partial decryptions are actually valid.

A.2 Bulletin Board

Chaum [Cha81] introduced the concept of a bulletin board, a public broadcast channel with memory where a party may write information that any party may read. Since then, bulletin boards have been often used in election schemes. All communications with the bulletin board are public and therefore can be monitored. In the application we consider, no party can erase any data.

A.3 Mix-nets

A re-encryption mix network [Cha81] is a distributed protocol that takes as input a set of messages, $\boldsymbol{m} = m_1, ..., m_n$ and returns an output consisting of the re-encrypted messages m_s, permuted according to a secret function ϕ: $\boldsymbol{m'} = m'_{\phi(1)}, ..., m_{\phi(n)}$. Thanks to this permutations it is not possible to associate a given m to any specific m'.

Several efficient and secure (in the sense that the claimed permutations are actually operated) mix-net protocols have been proposed in the literature, such as [Nef01], [FS01], and [JJR02]. In particular, [Gro03] has presented an efficient scheme for homomorphic encryption schemes (such as Paillier) that can be used as a building block for our protocol.

B Proof of Knowledge that Two Ciphertexts are Encryption of the Same Plaintext

[BFP$^+$01] provide an interactive zero-knowledge proof of the equality of plaintexts under Paillier encryption. Under their scheme, the prover can convince the verifier that, given p encryptions $c_i = g_i^m r_j^{n_j} \bmod n_j^2$, the various c_j's encrypt the same message m. We use a variation of this proof in our scheme - see below.

Based on their proof, we provide here the steps each authority has follow to prove to a voter that the share of credential she has received from it is also published under a different encryption scheme on the bulletin board.

Above we defined a share of credential created by a given authority as c. Imagine that $n_V = p_V q_V$ and $n_C = p_C q_C$ are the k-bit RSA moduli used respectively for the votes public key PK^V and the credentials public key PK^C. Then, define:

$$f_V = g_V^c r_V^{n_V} \bmod n_V^2 \tag{11}$$

and

$$f_C = g_C^c r_C^{n_C} \bmod n_C^2 \tag{12}$$

as two ciphertexts corresponding to the same λ-bit plaintext c, where g_V, g_C are public components derived from $n_V = p_V q_V$ and $n_C = p_C q_C$ respectively, as described in Section A.1, and r_V, r_C are randomizing components.

We look for a zero-knowledge proof that f_V and f_C are encryption of the same c. Assume that c lies in an interval $[0, 2^\lambda[$ and that the parameters A, t, λ are such that $1/A^t$ and $2^\lambda - kA$ are negligible. Then:

1. The authority picks at random $\rho \in [0, 2^k[$, $s_C \in \mathbb{Z}^*_{n_C^2}$, and $s_V \in \mathbb{Z}^*_{n_V^2}$, computes $u_C = g_C^\rho s_C^{n_C} \bmod n_C^2$ and $u_V = g_V^\rho s_V^{n_V} \bmod n_V^2$, and commits to u_V, u_C.
2. The voter picks a random challenge d in $[0, A[$ and sends it to the authority.
3. The authority computes $z = \rho + cd$, $w_C = s_C r_C^d \bmod n_C$, and $w_V = s_V r_V^d \bmod n_V$, and sends z, w_C, w_V to the voter.
4. The voter checks that $z \in [0, 2^k[$, $g_C^z w_C^{n_C} = u_C f_C^d \bmod n_C^2$, and $g_V^z w_V^{n_V} = u_V f_V^d \bmod n_V^2$.

After t iterations of this protocol, the probability that an honest authority is comparing $f_{C,v}$ that encrypt the same c is overwhelming, while the probability that f_C and f_V satisfy the steps above while f_C encrypts c_1 and ec_V encrypts c_2 and $c_1 \neq c_2$ is negligible. The proofs follow directly [BFP$^+$01]'s proof and are not reported here.

Note that under the Fiat-Shamir scheme [FS87], it is possible to replace the interaction of the verifier with a hash. Security would be guaranteed under the random-oracle. However, this proof would also be transferable. To reduce the number of interaction without producing a transferrable proof we will assume the existence of a designated-verifier proof [JSI96] of the scheme above, as in [HS00], [BFP$^+$01], and [JJ02].

C Sketch on El Gamal Variation

The El Gamal cryptosystem [ElG84] has properties in common with the Paillier's cryptosystem that we have applied in our scheme - in particular, self-blinding and homomorphism. So, we may replace in Section 4 Paillier encryption with El Gamal encryption at the cost of reducing the overall efficiency of the scheme. The advantage, however, would lie in the following consideration: consider two primes p and q and a generator g of G_q, which are the parameters of the system. The election authorities can then select two different private keys, s_1 and s_2, and use the corresponding public keys $h = g^{s_1}$ and $h = g^{s_2}$ for the votes and credentials respectively (similarly to E^V and E^C in the terminology we used for the Paillier's system). The difference is that those keys can be part of the same domain - hence credential shares created by the authorities do not need to be smaller than the domain of the cryptosystem: they can be larger than the domain, and still the evaluation in Step 2 of the Tallying phase (see Section 4.3) is possible, since the remainders are in the same domain, mod q. The calculation in mod q, however has the advantage of revealing even less information about the underlying parameters bs and cs.