The Vector-Ballot E-Voting Approach

Aggelos Kiayias[1] and Moti Yung[2]

[1] Computer Science and Engineering, University of Connecticut
Storrs, CT, USA. aggelos@cse.uconn.edu
[2] Computer Science, Columbia University
New York, NY, USA. moti@cs.columbia.edu

Abstract. Looking at current cryptographic-based e-voting protocols, one can distinguish three basic design paradigms (or approaches): (a) Mix-Networks based, (b) Homomorphic Encryption based, and (c) Blind Signatures based. Each of the three possesses different advantages and disadvantages w.r.t. the *basic properties* of (i) efficient tallying, (ii) universal verifiability, and (iii) allowing write-in ballot capability (in addition to predetermined candidates). In fact, none of the approaches results in a scheme that simultaneously achieves all three. This is unfortunate, since the three basic properties are crucial for efficiency, integrity and versatility (flexibility), respectively. Further, one can argue that a serious business offering of voting technology should offer a flexible technology that achieves various election goals with a single user interface. This motivates our goal, which is to suggest a new *"vector-ballot" based approach* for secret-ballot e-voting that is based on three new notions: *Provably Consistent Vector Ballot Encodings*, *Shrink-and-Mix Networks* and *Punch-Hole-Vector-Ballots*. At the heart of our approach is the combination of mix networks and homomorphic encryption under a single user interface; given this, it is rather surprising that it achieves much more than any of the previous approaches for e-voting achieved in terms of the basic properties. Our approach is presented in two generic designs called "homomorphic vector-ballots with write-in votes" and "multi-candidate punch-hole vector-ballots"; both of our designs can be instantiated over any homomorphic encryption function.

1 Introduction

There are three basic paradigms for cryptographic secure ballot elections. The first method is based on mix-networks where the tallying officials move the ballots between them and permute them in the process while changing their representation (e.g., partially decrypting them). Methods for doing this robustly and correctly have been designed in the last 20 years, starting with the initial work of Chaum [7]. In practical implementations, this approach in its fully robust form (i.e., proving the correctness of the shuffling) is still considered a slow tallying process, even though there have been several steps toward more efficient designs, see e.g. [33, 6, 20, 24, 1, 26, 18]. The second method is based on homomorphic encryption (started by Benaloh [9, 4, 3], and then followed by many other works

including [12, 10, 35, 16, 13, 2]). In this general approach the ballots are encrypted and then "compressed" via a homomorphic encryption scheme into a tally. This compression property allows fast tallying, and is what makes this approach attractive. However the drawback is that pure "compressible" homomorphic encryption is not suitable to deal with write-in ballots. In fact, compression of the write-in ballots content is *not possible* since, information theoretically, if this content has no redundancy (which is always possible in the write-in ballots case) compression will ruin it. A third approach is based on blind signatures, [17], and relies on the voters obtaining a certified secret ballot from the authorities by employing the blind signature module. This enables them to embed any form of ballot (including write-in). Subsequently, this approach requires the employment of an anonymous channel between the voter and the tallying authorities, to hide the identity of the user at the "ballot casting stage." This requirement may be inhibiting and thus people have suggested to combine this approach with the mix-net one so that the "anonymous channel" is implemented formally. Furthermore, while the previous paradigms support universal verifiability which assures robustness, this approach relies on tallier – voter interaction and does not support it.

In recent years, due to the aftermath of the USA 2000 presidential election debacle as well as other initiatives, e-voting technology has gained high level of interest (see [21]). One effort that took place is the joint Caltech-MIT electronic voting project. Rivest, who participated in this project, has raised the question whether it is possible to incorporate write-in ballots in homomorphic encryption based elections in a way that will still maintain its advantages and keep some of its computational gains. In fact, Rivest's question raises the more general concern that the cryptographic paradigms optimize different goals, and business wise it may be smart to combine them under a single user interface and hope to retain some of their individual advantages and to try to gain more by a combinational approach.

Homomorphic Vector-Ballot with Write-In Votes. Motivated by this question and issues, we started by attacking the problem of allowing write-in ballots as follows: since homomorphic encryption elections are based on a summation register (ciphertexts are modular-multiplied together which effectively sums-up the ballots under the encryption), write-in ballots need to be read individually.

To incorporate a write-in choice into a homomorphic encryption based scheme, we suggest the design of a composed ballot or a "vector ballot" that is cast by each user, and is either a regular (predetermined candidate) ballot or a write-in one with indistinguishable external representation in either case. This is the base of the vector-ballot approach. This sounds simple, but if done in a straightforward fashion this may give voters more "free choice" in ballot representation and in cheating, and it may also give more ways to distinguish between users' ballots. Thus, this new design leads to new concerns regarding ballot validity and ballot uniformity. In particular, it leads us to the simple yet crucial notion of *provably consistent vector ballot encodings*, which assures that in spite of the extended scenario the ballot is nevertheless legal, i.e. the voter is forced to ei-

ther select a write-in, or a predetermined choice, but *not* both at the same time. Further, whenever the voter makes one of the two choices, she is forced to enter a "neutral" value in the other portion of the vector ballot. The added validation proofs by the voter makes the ballot longer, however this price (constant increase in validity proof) is reasonable given the enhancement (to be described below) it enables. The ballot representation looks the same regardless of whether the user votes for a predetermined candidate or casts a write-in ballot. After the ballot casting, the vector ballot is split into a "supposedly regular portion" and a "supposedly write-in portion" and they are processed (tallied) independently.

What we have described so far is a combination of two voting approaches: homomorphic encryption based and mix-net based. While this is important (as it allows the unification under the same user-interface of the efficient homomorphic encryption based voting with the write-in "friendly" mix-net voting), by itself the resulting scheme as a whole is not more efficient than the two individual approaches (and clearly the real bottleneck is the slow tallying robust mix-net approach).

It is thus, perhaps surprising that our approach that is based on the vector ballots has the potential to achieve more efficient tallying than any previous proposal for e-voting that allowed write-in ballots and is universally verifiable at the same time. The two major points that allow this are explained below:

1. The predetermined candidate portions of all ballots can be compressed using the efficient homomorphic encryption based tallying.
2. The write-in portions of all ballots are based on an indicator and a write-in portion. Based on such indicators we show that they can be processed using the new efficient method of *shrink-and-mix* that we propose. The method takes advantage of the fact that the vector ballots are based on homomorphic encryption and the fact that, usually, *most* of the voters select one of the predetermined candidates. Thus, using the compressibility of indicators we can eliminate a great number of unused neutral write-in portions. We note that in a typical scenario, the method achieves a five-fold improvement over stand alone mix-network based election (and this will be a noticeable factor in practice, since the gain is within the system's performance bottleneck component).

Further, the two tallying procedures above are independent. Thus, the tallying can be performed in two phases. An *on-line* phase can just perform the homomorphic encryption tallying process of the predetermined candidate portions. This is a very efficient mechanism. In most cases the actual tally and winner(s) can be declared based on the these regular votes only and the slower tallying of the write-in portions can, in this case, be done *off-line* and a later time. Typically, the winner will be one selected among the leading predetermined candidates of the established parties, whereas, say, Mickey Mouse (a popular write-in candidate in US elections) can afford waiting a bit longer till he knows the actual number of votes he won.

The above is the first construction within the vector-ballot approach. It shows how we achieve simultaneously the basic properties of universal verifiability and

support for write-in ballots together with an efficient tallying procedure. Comparison to previous election paradigms is given in Figure 1.

Multi-Candidate Punch-Hole Vector-Ballot. A modular extension of our new approach employs our notion of *punch-hole vector ballots* which enables a more suitable scheme for voting with a large number of predetermined candidates. It extends the functionality of our vector-ballot encodings (and thus write-ins can still be incorporated). The method introduces a multitude of c summation ciphertext registers, one per candidate, while earlier schemes packed all the candidate tallies into a single summation register. Note that the vector ballot portions in the ballot design correspond to various candidates and to the corresponding summation registers. Note further that a ballot needs to have a consistent valid encoding and the voter has to prove this validity. This is different from the simplistic multi-election ballot in [4].

Employing separate registers relaxes the burden of participants by allowing them to deal with smaller ciphertexts. The gain is especially noticeable in case of many candidates. To formalize the ciphertext summation register size requirement, we introduce the notion of "capacity" of an homomorphic encryption function, which measures how many integers can be represented as plaintexts within a given summation register. Our punch-hole vector ballot design requires the capacity of the underlying encryption to be only n (the number of voters), instead of n^c required for a single summation register used previously. In fact, all leading proposed election methods in the literature that employ summation registers, [10, 16, 13], and allow for n voters and c candidates, indeed, require capacity of n^c. Note that this may cause problems in selecting the security parameter when the number of candidates is very large: e.g., if the security parameter is 768 bits, this restricts the capacity to 2^{768}, and if the number of candidates is large, e.g. $c = 50$, and the voting population close to $35,000$, then the capacity *cannot* contain the summation register.

An important and substantial gain in efficiency of tallying, results from the new approach when applied over the ElGamal encryption. The recovery of the final tally requires only time $O(cn)$ which is *polynomial in the number of candidates*, instead of $O(n^c)$ which is exponential in c, as is the case in the state of the art discrete-log based scheme of [10]. We remark that this exponential gain is traded against a quadratic – rather than linear – (in c) work done for validity checking of ballots; this is a reasonable price to pay for such a gain.

2 Preliminaries

Requirements For Voting Schemes. A voting-scheme needs to fulfill a variety of requirements to become useful. A brief presentation of these requirements follows.

Secrecy. Ensures the security of the contents of ballots. This is typically achieved by relying on the honesty of a sufficient number of the participating authorities and at the same time on some cryptographic intractability assumption. In particular, any polynomial-time probabilistic adversary that controls some arbi-

Approaches	Efficient Tallying	Univ. Verifiability	Write-ins
Homomorphic Encryption	√	√	×
Mix-networks	×	√	√
Blind-Signatures	√	×	√
Vector-Ballot approach	√	√	√

Fig. 1. A comparison of our new approach with previous work with respect to the following three important properties: (i) *efficient-tallying*: tallying does not require the application of a robust-mix to the total number of ballots; (ii) *universal-verifiability*: any interested third party may verify that the election protocol is executed correctly; (iii) *write-ins*: voters are allowed to enter write-in votes.

trary number of voters and a number of authorities (below some predetermined threshold) should be incapable of distinguishing which one of the predetermined choices a certain voter selected or whether the voter entered a write-in. In all voting schemes, once a certain number of votes have been aggregated into a partial tally, secrecy is not mandatory, e.g., once the votes of a precinct have been aggregated it is ok to reveal the partial tally (in fact in many cases it is not even desired to keep the partial tallies secret, if some regional statistics are to be extracted from the election results). Thus, voter secrecy will have an associated *Privacy Perimeter* b which will refer to the smallest number of votes that need to be aggregated into a partial tally before some information about the partial tally can be revealed; we will talk of secrecy with b-perimeter in this case. A more formal discussion of secrecy is deferred to the full version of this paper.

Universal-Verifiability. Ensures that any party, including an outsider, can be convinced that all valid votes have been included in the final tally.

Robustness. Ensures that the system can tolerate a certain number of faulty participants.

Fairness. It should be ensured that no partial results become known prior to the end of the election procedure.

Another property, which we do not deal with here explicitly, is **Receipt-Freeness** [5, 34, 27, 22, 25]. Standard techniques that use re-randomizers (see e.g. [2]) can be readily employed in our schemes to allow this property.

Homomorphic Encryption Schemes. An encryption scheme is a triple $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$. The key-generation $\mathcal{K}$ is a probabilistic TM which on input a parameter 1^w (which specifies the key-length) outputs a key-pair pk, sk (public-key and secret-key respectively). The encryption function is a probabilistic TM $\mathcal{E}_{\mathsf{pk}} : \mathbb{R} \times \mathbb{P} \to \mathbb{C}$, where $\mathbb{R}$ is the randomness space, $\mathbb{P}$ is the plaintext space, and $\mathbb{C}$ the ciphertext space. When $\mathbb{P} = \mathbb{Z}_a$ for some integer a, we will say that the encryption function has "*additive capacity*" (or just capacity) a. The basic property of the encryption scheme is that $\mathcal{D}_{\mathsf{sk}}(\mathcal{E}_{\mathsf{sk}}(\cdot, x)) = x$ for all x independently of the coin tosses of the encryption function $\mathcal{E}$. If we want to specify the coin tosses of $\mathcal{E}$ we will write $\mathcal{E}_{\mathsf{pk}}(r, x)$ to denote the ciphertext that corresponds to the plaintext x when the encryption function $\mathcal{E}_{\mathsf{pk}}$ makes the coin tosses r. Otherwise we will consider $\mathcal{E}_{\mathsf{pk}}(x)$ to be a random variable. For homomorphic encryption, we assume additionally

the operations $+, \oplus, \bigodot$ defined over the respective spaces $\mathbb{P}, \mathbb{R}, \mathbb{C}$, so that $\langle \mathbb{P}, + \rangle$, $\langle \mathbb{R}, \oplus \rangle$, $\langle \mathbb{C}, \bigodot \rangle$ are groups written additively (the first two) and multiplicatively respectively.

Definition 1. *An encryption function $\mathcal{E}$ is homomorphic if, for all $r_1, r_2 \in \mathbb{R}$ and all $x_1, x_2 \in \mathbb{P}$, it holds that $\mathcal{E}_{\mathsf{pk}}(r_1, x_1) \bigodot \mathcal{E}_{\mathsf{pk}}(r_2, x_2) = \mathcal{E}_{\mathsf{pk}}(r_1 \oplus r_2, x_1 + x_2)$.*

We will consider two examples of Homomorphic Encryption schemes: "additive ElGamal" and Paillier Encryption. Both have been employed in the design of e-voting schemes in the past, see [10] and [13, 2] respectively (which are also the current state-of-the-art schemes in the homomorphic encryption based approach). We define them below:

Additive ElGamal Encryption. It is defined by a triple $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$: the key-generation $\mathcal{K}$ outputs the description of a finite multiplicative group $\mathcal{G}$ of prime order q, with three generators $\langle g, h, f \rangle$ which are set to be the public-key of the system pk; the secret-key sk is set to the value $\log_g h$. For a public-key $\langle g, h, f \rangle$ the encryption function $\mathcal{E}(r, x)$ equals the tuple $\langle g^r, h^r f^x \rangle$, and the domains $\mathbb{P} := \mathbb{Z}_q$, $\mathbb{R} := \mathbb{Z}_q$ and $\mathbb{C} := \mathcal{G} \times \mathcal{G}$. The operations $+, \oplus$ are defined as addition modulo q and the operation $\bigodot$ is defined as point-wise multiplication over $\mathcal{G} \times \mathcal{G}$. The decryption function $\mathcal{D}$ for a secret-key $\log_g h$ given $\langle G, H \rangle$ it returns $H / G^{\log_g h}$, and then it performs a brute-force search over all possible values f^x to recover x. Observe that $\langle \mathbb{P}, + \rangle$, $\langle \mathbb{R}, \oplus \rangle$ and $\langle \mathbb{C}, \bigodot \rangle$ are all groups, and the encryption $\mathcal{E}$ is homomorphic with respect to these operations. Finally notice that the capacity of $\mathcal{E}$ is q.

Paillier Encryption. [28]. It is a triple $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$, defined as follows: the key-generation $\mathcal{K}$ outputs an integer N, that is a product of two safe primes, and an element $g \in \mathbb{Z}^*_{N^2}$ of order a multiple of N. The public-key of the system pk is set to $\langle g, N \rangle$ and the secret-key sk is set to the factorization of N. For a public-key $\langle g, N \rangle$, the encryption function $\mathcal{E}(r, x)$ equals the value $g^x r^N \pmod{N^2}$ and the domains $\mathbb{P} := \mathbb{Z}_N$, $\mathbb{R} := \mathbb{Z}^*_{N^2}$, and $\mathbb{C} := \mathbb{Z}^*_{N^2}$. The operation $+$ is defined as addition modulo N, and the operations $\oplus, \bigodot$ are defined as multiplication modulo N^2. The decryption function $\mathcal{D}$ for a secret-key p, q it operates as follows: first it computes $\lambda := \lambda(N)$ the Carmichael function of N, and given a ciphertext c, it returns $L(c^\lambda \pmod{N^2}) / L(g^\lambda \pmod{N^2})$ where $L(u) = \frac{u-1}{N}$ and L is defined over the set of integers $\{u \mid u \equiv 1 (\bmod N)\}$. Again, observe that $\langle \mathbb{P}, + \rangle$, $\langle \mathbb{R}, \oplus \rangle$ and $\mathbb{C}, \bigodot \rangle$ are all groups, and the encryption $\mathcal{E}$ is homomorphic with respect to these operations. Finally notice that the capacity of $\mathcal{E}$ is N.

Proofs of Knowledge. Proofs of knowledge are protocols between two players, the Prover and the Verifier. In such protocols there is a publicly known predicate Q for which the prover knows some witness x, i.e. $Q(x) = 1$. The goal of such protocols is for the prover to convince the verifier that he indeed knows such witness. We will concentrate on "3-move" protocols for which the prover acts in the first and third move, and the verifier challenges in the second move with a random value from the proper domain (see [11]). Conversations in such protocols will be of the form $\langle a, c, r \rangle$, and the verifier will accept provided that

a, c, r satisfy some conditions given as part of the specifications of the protocol. Proofs of knowledge can be made *non-interactive* by employing the Fiat-Shamir heuristics, [15] (and then, security is shown in the random-oracle model, or alternatively assuming a beacon, [32]). If the predicate Q accepts a non-interactive zero-knowledge proof, and an agent possesses a witness for Q that he wants to prove knowledge of, we will say that the agent "writes a proof for Q." Proofs of knowledge of the above type can be combined in "AND" and "OR" fashion in an efficient manner.

Proofs of Knowledge for Homomorphic Encryption. Let $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ be a homomorphic encryption scheme. Below, we identify useful proof protocols (which have been used in various settings and environments).

Proof of Knowledge of Properly Formed Ciphertext for a Public Plaintext. A useful proof of knowledge in the context of e-voting is a proof that shows that a ciphertext that encrypts a publicly known plaintext is properly formed. We define the predicate $Q^{m,V}_{\mathsf{cipher}}$ as follows $Q^{m,V}_{\mathsf{cipher}}(r) = 1$ if and only if $\mathcal{E}_{\mathsf{pk}}(r, m) = V$. We remark that proofs of knowledge for the two homomorphic encryptions that we consider here (additive ElGamal, and Paillier) are standard and can be done efficiently.

Proof of Knowledge for a Random Shuffle. Observe that using a homomorphic encryption scheme one can "re-randomize" a ciphertext C by computing $C' := \mathcal{E}_{\mathsf{pk}}(0) \bigodot C$ (i.e., C' is uniformly distributed over all ciphertexts that correspond to the plaintext of C). Suppose now that $C_1, \ldots, C_k$ is a sequence of ciphertexts and $C'_1, \ldots, C'_k$ is a random re-encrypted permutation of these ciphertexts. We define a predicate $Q^{C_1,\ldots,C_k,C'_1,\ldots,C'_k}_{\mathsf{shuffle}}$ so that $Q^{C_1,\ldots,C_k,C'_1,\ldots,C'_k}_{\mathsf{shuffle}}(r_1, \ldots, r_k, \pi) = 1$ if and only if $C'_{\pi(j)} = \mathcal{E}_{\mathsf{pk}}(r_j, 0) \bigodot C_j$, for $j = 1, \ldots, k$.

A straightforward approach for a proof for $Q^{C_1,\ldots,C_k,C'_1,\ldots,C'_k}_{\mathsf{shuffle}}$ would require $\mathcal{O}(k^2)$ space. Discovering more efficient proofs is a very active area of research (as such proofs constitute the basic operation of a robust mix-network, a fundamental primitive for elections based on mixes) and several papers provided sophisticated techniques of shortening the proof as well as relaxing the robustness model to allow more efficient implementations, [20, 24, 26, 18]. Two of the most efficient recent protocols are that of [18] and [26], that allow $\mathcal{O}(k)$-size proofs with relatively small constant.

Threshold Homomorphic Encryption Schemes. A (t, m)-threshold homomorphic encryption scheme is a triple $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$ so that $\mathcal{K}$ is a protocol between a set of participants $A_1, \ldots, A_m$, that results in the publication of the public-key pk and the sharing of the secret-key sk so that any t of them can reconstruct it. Additionally, $\mathcal{D}$ is also a protocol between the participants $A_1, \ldots, A_m$ that results in the decryption of the given ciphertext in a publicly verifiable manner (i.e. each participant writes a proof that he follows the decryption protocol according to the specifications). Both Additive ElGamal and Paillier encryptions have threshold variants, see [30, 31, 19] and [16, 13] respectively.

3 The Vector Ballot Approach

The participants in our schemes are the voters $V_1, \ldots, V_n$, the authorities $A_1, \ldots, A_m$, and the bulletin board server which is responsible for maintaining an authenticated communication transcript. Voter eligibility as well as basic ciphertext processing operations are also handled by the bulletin board server. Our voting approach is divided in four major steps: **Setup**, **Ballot-Casting**, **Tallying**, **Announcement of the Results**.

In our approach, every encrypted ballot is in fact a "vector-ballot" that has three coordinates: the first is a ciphertext that contains possibly one of the predetermined election choices, the second is a flag-ciphertext that encrypts the information whether the voter selects a write-in choice or not; finally, the third coordinate possibly contains a write-in choice. A proof of "consistent ballot encoding" will be broken into a number of "consistency arguments" and will ensure that the vector ballot is formed properly (i.e., it either contains a predetermined choice in the first coordinate or a write-in choice in the last coordinate and furthermore the "flag" value is encrypted consistently). The tallying phase has two independent phases: (i) tallying the non-write-in election results using the homomorphic encryption function properties; (iia) shrinking the number of write-in votes using the flag-ciphertexts; (iib) employing a mix-net over the shrunk write-in ballot sequence. The general overview of these procedures is presented in Figure 2. We describe our approach in detail in the following subsections.

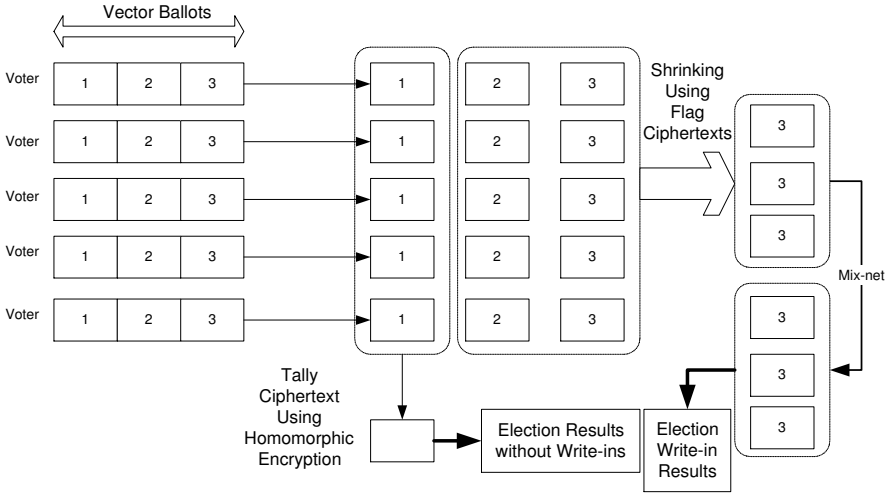

Fig. 2. The Vector-Ballot E-Voting Paradigm.

3.1 Setup and Capacity Assumption

In our approach we will employ a threshold homomorphic encryption function $\langle \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$. We will also employ the necessary assumption regarding the capacity of the encryption function:

Capacity Assumption. The capacity of the encryption function satisfies $a > M^c$ where c is the number of candidates, and M an integer with $M > n$ (the number of voters).

Setup. The authorities $A_1, \ldots, A_m$ execute the protocol $\mathcal{K}$ which results in the publication in the bulletin board of the public-key pk. At the same time the secret-key sk is shared amongst the authorities $A_1, \ldots, A_m$.

3.2 Ballot-Casting Step

Each eligible voter gets authorized to the bulletin board and reads the public-key pk of the system. The set of choices is defined as $\mathsf{Choices} := \{1, M, M^2, \ldots, M^{c-1}\}$ where M is an integer with the property $M > n$.

Formation the Vector-Ballot. Each voter V_i publishes a vector ballot $\langle C_1[i], C_2[i], C_3[i] \rangle$. If the voter wishes to select one of the predetermined choices of the election she selects $C_1[i]$ to be an encryption of one of the values in the set $\mathsf{Choices}$ while $C_2[i], C_3[i]$ are encryptions of 0; in particular $C_1[i] := \mathcal{E}_{\mathsf{pk}}(M^{\ell_i - 1})$ where $\ell_i \in \{1, \ldots, c\}$ is the personal choice of the voter, and $C_2[i] := \mathcal{E}_{\mathsf{pk}}(0), C_3[i] := \mathcal{E}_{\mathsf{pk}}(0)$. If the voter wishes to enter a write-in ballot she selects $C_1[i]$ to be an encryption of 0, $C_2[i]$ to be an encryption of 1, and $C_3[i]$ to be an encryption of some string string_i which is the voter's write-in entry. Formally, $C_1[i] := \mathcal{E}_{\mathsf{pk}}(0)$, $C_2[i] := \mathcal{E}_{\mathsf{pk}}(1)$ and $C_3[i] := \mathcal{E}_{\mathsf{pk}}(\mathsf{string}_i)$. Together with her vector ballot the voter must publish a proof of "consistent ballot encoding." In particular V_i writes a proof for the following predicate:

$$\left((Q^{1, C_1[i]}_{\mathsf{cipher}} \vee Q^{M, C_1[i]}_{\mathsf{cipher}} \vee \ldots \vee Q^{M^{c-1}, C_1[i]}_{\mathsf{cipher}}) \wedge Q^{0, C_2[i]}_{\mathsf{cipher}} \wedge Q^{0, C_3[i]}_{\mathsf{cipher}} \right) \vee (Q^{0, C_1[i]}_{\mathsf{cipher}} \wedge Q^{1, C_2[i]}_{\mathsf{cipher}})$$

The above proof can be done *efficiently* as discussed in section 2, since it is an AND/OR composition of the proof of knowledge for the predicate $Q^{m,V}_{\mathsf{cipher}}$ which can be done quite efficiently for either of the two homomorphic encryption functions that we consider. Moreover it only adds a small constant overhead compared to proofs of previous homomorphic-encryption based voting schemes.

Regarding the above proof of consistent ballot encoding it is easy to prove the following lemma:

Lemma 1. *The only ballot encodings for $\langle C_1[i], C_2[i], C_3[i] \rangle$ allowed are:*
(i) The second ciphertext encrypts a 0, the first ciphertext contains a value from the set $\mathsf{Choices}$ and the third ciphertext encrypts a 0.
(ii) The second ciphertext encrypts a 1, the first ciphertext encrypts a 0 and the third ciphertext is unrestricted.

3.3 Tallying Step

The Non-write-in Part. The vector ballots are parsed so that the first component is collected and the sequence of ciphertexts $C_1[1], \ldots, C_1[n]$ is formed. The "tally ciphertext" is defined as $C_{\mathsf{tally}} = C_1[1] \bigodot \ldots \bigodot C_1[n]$. It is easy to see that due to the homomorphic property and the capacity assumption, C_{tally} is a ciphertext that hides a value T that satisfies $T = \sum_{i \in V} M^{\ell_i}$ (as an integer), where $V \subseteq \{1, \ldots, n\}$ is the set of voters that did not select the write-in option. Observe that if $k_0, \ldots, k_{c-1}$ are the tallies won by each of the c candidates, it holds that $T = k_0 + k_1 M + \ldots + k_{c-1} M^{c-1}$, and $k_0, \ldots, k_{c-1} < M$, i.e., if we write T as an integer in base M we can obtain the counts for each candidate.

Dealing with the Write-ins — Shrink-and-Mix networks. Write-in ballots are not "compressible" into a single ciphertext like regular ballots and thus they have to be mixed and revealed one by one. Nevertheless our approach allows for a significant efficiency improvement that we call a **Shrink-and-Mix** network. A shrink-and-mix network for voting is a mix-network that attempts to shrink the input ballot sequence prior to the mix procedure in order to gain efficiency. (Indeed gaining efficiency in settings where it is possible is crucial, given the state of the art of Mix networks, see [20]). Shrink-and-mix is a concept that naturally binds to our approach that combines write-in ballots with regular homomorphic encryption based e-voting. This is because in our approach the following unique properties are true:
1. Most voters will not cast a write-in ballot, but rather select one of the predetermined choices of the election.
2. There is a way to employ the homomorphic properties of the encryption function to test whether a small batch of encrypted vector ballots contains a write-in without violating the privacy of the voters (assuming a given security perimeter b).
3. There is a way to find the exact number of write-in votes prior to opening them, without violating the secrecy of the voters (assuming a given security perimeter b).
Justification. For item *1*, observe that in most settings the write-in option will be used sparingly by the voters who will typically select one of the predetermined candidates for the election. For item *2*, recall that in the vector-ballot approach, each vector ballot $\langle C_1[i], C_2[i], C_3[i] \rangle$ contains a "flag-ciphertext" (the value $C_2[i]$) that encrypts the value 0 or 1 depending on whether the voter voted with one of the predetermined choices (in $C_1[i]$) or entered a write-in (in $C_3[i]$). Suppose now that we have a set of voters $i_1, \ldots, i_b$ and we want the authorities to check whether one of them entered a write-in without violating the privacy of the voters. Then, simply the authorities collect the flag ciphertexts from the vector ballots of these voters $C_2[i_1], \ldots, C_2[i_b]$ and decrypt the ciphertext $C_{i_1,\ldots,i_b} := C_2[i_1] \bigodot \ldots \bigodot C_2[i_b]$. Now observe that the decryption of $C_{i_1,\ldots,i_b}$ is the number of write-in votes entered by the voters $\{i_1, \ldots, i_b\}$ and thus the authorities are capable of deducing whether there is a write-in entry among the ciphertexts $\{C_3[i_1], \ldots, C_3[i_b]\}$.

For item *3*, observe that the ciphertext $C_{1,\ldots,n} := C_2[1] \bigodot \ldots \bigodot C_2[n]$ is an encryption of the number of write-in votes. Thus if the authorities wish to find efficiently the exact number of write-ins they have to compute $C_{1,\ldots,n}$ and decrypt it (recall that linear computations in the number of voters constitute practically optimal complexity for the tallying phase).

Given the above properties we can now describe the shrink-and-mix method which is divided in two separate stages (perhaps not surprisingly named): (a) shrink and (b) mix.

Shrink Stage. First we describe the shrink stage in detail below:

`Functionality` *Input*: the sequence of all vector-ballots. Let $V \subseteq \{1,\ldots,n\}$ be the subset of voters that entered a non-write-in ballot and denote by V' the set $\{1,\ldots,n\} - V$ (the subset of voters that entered a write-in).

`Output`: a set V^* such that $V' \subseteq V^* \subseteq \{1,\ldots,n\}$.

`Initialize`. The authorities $A_1,\ldots,A_m$ compute the number of write-ins h (feasible by item *3* above). Let p denote the probability that an arbitrary voter enters a write-in defined as $p := h/n$. Let b be the desired privacy perimeter for the elections.

`Shrink`. Let $\sigma := \langle C_2[1],\ldots,C_2[n]\rangle$ be the sequence of the second components of all ballot vectors and let V^* initially defined to $\{1,\ldots,n\}$. The authorities divide σ into n/b batches so that each batch contains b ciphertexts. Since the probability of an arbitrary voter to enter a write-in is p it follows that the probability that a batch contains no write-in is $(1-p)^b$. The authorities test whether each one of the n/b batches contains a write-in or not (as described in the item *2* above). If the batch of flag-ciphertexts that corresponds to the voters $\{i_1,\ldots,i_b\}$ does not contain a write-in we modify $V^* = V^* - \{i_1,\ldots,i_b\}$. Assuming that each batch is independent from the other, it follows that the expected number of batches without a write-in is $\frac{n}{b}(1-p)^b$, so the expected size of V^* will be $n - n(1-p)^b$. Observe that the correctness of the shrink stage (i.e. $V' \subseteq V^* \subseteq \{1,\ldots,n\}$) follows easily. The closeness of V^* to V' can be calibrated by lowering the parameter b.

Mix Stage. The mix-stage follows.

`Functionality` *Input*: A sequence of ciphertexts $\sigma^* := \langle G[i]\rangle_{i=1,\ldots,n^*}$, where $n^* = |V^*|$, V^* is the output of shrink and $\langle G[1],\ldots,G[n^*]\rangle = \langle V_3[i] \mid i \in V^*\rangle$.

`Output`: a sequence of ciphertexts $\langle G'[i]\rangle_{i=1,\ldots,n^*}$ so that there is a permutation π on $\{1,\ldots,n^*\}$ that satisfies $G'[i]$ is a random re-encryption of $G[\pi(i)]$. `Mix.` The authorities $A_1,\ldots,A_m$ execute a "robust mix" for the sequence of ciphertexts $\sigma^* = \langle G[1],\ldots,G[n^*]\rangle$. This can be accomplished by employing any existing robust-mix method, [20, 24, 26, 18]. The most straightforward robust mix technique has each authority re-encrypting each ciphertext $G[i]$ and permuting the whole sequence randomly to obtain the sequence $\langle G'[1],\ldots,G'[n^*]\rangle$ and also writing a proof for $Q_{\mathsf{shuffle}}^{G[1],\ldots,G[n^*],G'[1],\ldots,G'[n^*]}$. Note that authorities perform the above steps in sequence by acting on the output of the previous authority.

We remark that robust mixes are expensive in terms of computation and space, for this reason the shrink stage that our model allows can be crucial for the improvement of the efficiency of the mixing. The *shrink ratio* for a shrink-

and-mix network is the expected reduction percentage of the given sequence of ciphertexts $\langle C_3[i] \rangle_{i \in V^*}$ i.e., the fraction $(n - |V^*|)/n$. Observe that it holds that $(n - |V^*|)/n = (1-p)^b$ and thus the shrink ratio equals is $(1-p)^b$ (employing average-case analysis). To illustrate the gain we obtain using the shrink-and-mix network consider the following scenario: in many elections it is reasonable to expect a write-in probability 1/100, so by setting the privacy perimeter $b = 20$ (which is reasonable for the privacy of the voters in most settings) we obtain a shrink ratio of approximately 0.81, which means that 81% of the ciphertexts will be discarded prior to the execution of the robust mix. This translates to a significant gain in the efficiency of the mixing procedure.

3.4 Announcement of the results

First the authorities announce the results for the non-write-in part of the election (in fact, this step can be performed prior to the execution of the shrink-and-mix network). The authorities $A_1, \ldots, A_m$ execute the protocol $\mathcal{D}$ on the ciphertext C_{tally} to reveal the the value T. Due to the properties of the value T it holds that if T, as an integer, is written in base M, then the tallies for each candidate are revealed (cf. section 3.3); note that due to the capacity assumption there will be no wrap-arounds during the computation of the tally ciphertext C_{tally}.

Subsequently the authorities execute the shrink-and-mix network (and frequently this will be done after the winner of the election is already determined from the non-write-in votes) and then they execute the protocol $\mathcal{D}$ for each of the ciphertexts $G_*[1], \ldots, G_*[n^*]$ that belong to the output of the shrink-and-mix network. This will reveal all the strings string_i for $i \in V^*$, where V^* is the output of the shrink stage. Since $V' \subseteq V^* \subseteq \{1, \ldots, n\}$ all entered write-ins will be revealed (with, perhaps, a number of 0's that correspond to the ciphertexts that were entered by the voters in $V^* - V'$). When $\mathsf{string}_i = 0$ the entry will be removed from the write-in vote listing (recall that "0" is not considered a valid write-in vote). The final elections results consist of the counts for each of the pre-determined candidates as well as counts for the write-in selections.

3.5 Properties of the Paradigm

Efficiency. First note that our vector-ballot approach can be readily instantiated over the two homomorphic encryption functions (additive ElGamal or Paillier) that we describe in section 2.

The Voters' Perspective. The activity of each voter in the vector-ballot approach includes the following operations: after the setup phase each voter must be authenticated to the bulletin board server. The bulletin board server maintains the listing with all eligible voters. After authentication, the voter reads from the bulletin board the public-key of the authorities and all other information that is pertinent to the election, i.e., the listing of predetermined candidates. The voter privately decides on one of the predetermined candidates or to a certain write-in choice and publishes her encrypted ballot which consists of the three ciphertexts as described in section 3.2. Further she needs to publish the proof of consistent

ballot-encoding. This is done by writing in the bulletin board the non-interactive zero-knowledge proof as described in section 3.2. This proof has size linear to the number of predetermined candidates and can be generated very efficiently for the two homomorphic encryption schemes that we consider.
The Authorities' Perspective. The work of the authorities is divided in two separate stages. (i) Before ballot-casting the authorities execute the **Setup** stage of the election that requires them to run the key-generation protocol of the employed threshold homomorphic encryption scheme. (ii) After the ballot-casting phase the authorities proceed to the tallying phase. The aggregation of the non-writein part of voters' encrypted ballots is a linear operation in the number of voters that employs the homomorphic property of the underlying encryption scheme. Observe that this task can be arbitrarily distributed to any number of entities. Given the aggregated ciphertext the authorities decrypt it by executing the decryption protocol of the underlying homomorphic encryption scheme; this reveals the counts for the predetermined candidates. It is highly likely that a winner of the election can be already determined at this stage. Subsequently, the authorities execute the shrink-and-mix protocol. This requires the authorities to execute a robust-mix protocol, but *only* over the encrypted writein ballots that remain after the shrinking phase. The shrinking phase by itself is efficient as it is only linear in the number of encrypted ballots. Subsequently the execution of the robust-mix is performed in the shrunk writein encrypted ballot sequence which allows a significant gain as it is argued in section 3.3. Furthermore any robust mix can be used in a black-box fashion by our shrink-and-mix method; thus we can take advantage of any sophisticated robust shuffling protocol, e.g. the schemes of [20, 24, 26, 18].

Comparison to Previous Work. We first observe that the efficiency of our scheme is comparable to previous approaches in the homomorphic encryption based election. In fact the only difference is the small constant overhead that is introduced in the part of the voter since she has to provide a proof of a consistent ballot encoding. In previous homomorphic encryption based solutions the "proof of ballot-validity" is also linear in the number of candidates; note that this cannot be improved further if we use encrypted-ballots coupled with a "1-out-of-c" non-interactive zero-knowledge proof (which has by definition length linear in c). Going beyond the homomorphic-encryption approach, our approach allows the incorporation of writein votes. In this respect, we first observe that our schemes achieve universal-verifiability, unlike the previous writein approach based on blind signatures. When compared to the mix-network approach, we also employ a "robust-mix" but we do so with a significant gain compared to the previous mix network protocols: indeed since the great majority of the voters will not cast a write-in vote, our *shrink-and-mix* approach will achieve a *five-fold*[1] improvement in many typical scenarios. This is a very significant improvement in any practical implementation.

[1] Assuming writein probability of 1/100, see section 3.3.

Security. Regarding the security properties of our scheme we make the following claim: The e-voting approach described above satisfies secrecy with b-perimeter, universal-verifiability, robustness and fairness provided that (i) less than t authorities are malicious, (ii) the underlying homomorphic encryption scheme is semantically secure, (iii) participants can consult a beacon for the purpose of generating challenges for the zero-knowledge proofs.
Justification. First we argue about Universal-verifiability: a third party auditor can verify that all votes have been counted by performing the following three steps: (i) *verifying the non-writein part*: the auditor recomputes the tally ciphertext C_{tally} from the first portion of every voter's vector-ballot and verifies that the authorities decrypted C_{tally} properly by checking the non-interactive zero-knowledge proof of decryption; (ii) *verifying the shrinking phase*: the auditor recomputes all ciphertexts $C_{i_1,\ldots,i_b}$ that were used in the shrinking stage of the shrink-and-mix network and verifies their decryption as in (i). (iii) *verifying the robust mix*: the auditor checks all mixing proofs given by the shuffling authorities during the mixing procedure.

Regarding fairness, we observe that no partial sum can be revealed to any third party due to the semantic-security of the homomorphic encryption function and the zero-knowledge properties of the proofs of consistent ballot encodings. Regarding robustness observe that it is guaranteed unconditionally for voters: any eligible voter may fail without having any impact on the protocol; furthermore, any number of authorities below the threshold t may fail without affecting the protocol. Note that we do not deal with failures explicitly affecting the bulletin board server which is a formalism used in a black-box fashion in all the cryptographic e-voting literature, [3]. Finally, secrecy with b-perimeter of our scheme is justified based on the semantic security of the underlying homomorphic encryption scheme. More details will be provided in the full version.

4 Punch-Hole / Write-in Ballots

In settings where the number of candidates c and the number of voters n is large it could be the case that it might be detrimental (in terms of efficiency) to use any scheme based on the homomorphic encryption approach ([10, 16, 13]) as well as our approach of the previous section. This is because the capacity assumption (employed by all the above protocols) mandates that the capacity a of the encryption function satisfies the condition $a > n^c$. Even worse if the additive ElGamal instantiation is used (as e.g. in the case of the scheme of [10]) the tallying phase would require a brute-force step proportional to n^c which is very expensive. For such cases we introduce an alternative generic vector ballot design for our e-voting approach that is capable of dealing with such settings very efficiently. In the variant of our approach of this section, the ballot of each voter consists of $c + 2$ ciphertexts (instead of 3) and the only allowed ballot encodings are the following (i) encrypt a single "1" in the first c ciphertexts and "0" everywhere else, or (ii) enter a write-in ballot in the last ciphertext, encrypt a "0" in the first c ciphertexts and encrypt a "1" in the $(c + 1)$-th ciphertext

(which plays the role of the "flag-ciphertext"). The encoding can be thought of as "punch-hole/write-in" voting because the voter either "punches" a hole in the first c locations (by voting "1") or enters his write-in choice in the last location. In the remaining we briefly explain the approach, mentioning only the cases where there is significant difference from our paradigm of section 3. More details will be provided in the full version.

First we note that the capacity assumption will be relaxed as follows:

Relaxed Capacity Assumption. The capacity a of the encryption function satisfies $a > n$ (the number of voters).

Formation the Vector-Ballot. Each voter V_i publishes a vector ballot $\langle C_1[i], C_2[i], \ldots, C_{c+1}[i], C_{c+2}[i]\rangle$. If the voter wishes to select one of the predetermined choices $\{1, \ldots, c\}$ of the election she selects $C_{\ell_i}[i] := \mathcal{E}_{\mathsf{pk}}(1)$, where $\ell_i \in \{1, \ldots, c\}$ is her choice, and then sets $C_\ell[i] := \mathcal{E}_{\mathsf{pk}}(0)$ for all $\ell \in \{1, \ldots, c+2\} - \{\ell_i\}$. On the other hand, if the voter wishes to enter a write-in she selects $C_{c+2}[i] := \mathcal{E}_{\mathsf{pk}}(\mathsf{string}_i)$ where string_i is her write-in choice, and sets $C_{c+1}[i] := \mathcal{E}_{\mathsf{pk}}(1)$ as well as $C_\ell[i] := \mathcal{E}_{\mathsf{pk}}(0)$ for $\ell = 1, \ldots, c$. Together with her vector ballot the voter publishes a proof of a consistent vector ballot encoding to ensure that her ballot is formed properly. More specifically this is done as follows:

(Consistency Argument #1) V_i shows that the first $c+1$ locations of her vector ballot contain only a single 1 among c 0's; this is accomplished as follows: V_i publishes a random re-encrypted shuffle of the c+1 ciphertexts $C_1[i], C_2[i], \ldots, C_{c+1}[i]$ denoted by $\langle C'_1[i], C'_2[i], \ldots, C'_{c+1}[i]\rangle$ and proves its correctness. Then, the voter opens all ciphertexts $C'_\ell[i]$ (by showing their coin-tosses); this allows any third party to verify that the plaintexts encrypted in $C'_1[i], \ldots, C'_{c+1}[i]$ are exactly a single 1, among c 0's; due to the zero-knowledge proof this shows that the same is true about the corresponding ciphertexts in the vector-ballot; note that due to the zero-knowledge properties of the shuffle this step does not reveal any information about the location of the 1.

(Consistency Argument #2). The voter shows that either the two last ciphertexts in the vector ballot encrypt 0, or that the $(c+1)$-th ciphertext encrypts a 1, i.e., V_i writes a proof for the predicate $(Q_{\mathsf{cipher}}^{0,C_{c+1}[i]} \wedge Q_{\mathsf{cipher}}^{0,C_{c+2}[i]}) \vee (Q_{\mathsf{cipher}}^{1,C_{c+1}[i]})$

It is easy to verify that the above consistency arguments enforce the intended ballot-encodings. In the tallying phase, the vector ballots are parsed so that the first c components are collected and the c sequences of ciphertexts $C_\ell[1], \ldots, C_\ell[n]$ are formed for $\ell = 1, \ldots, c$. We define c "tally ciphertexts" as $C_{\mathsf{tally}}^\ell = C_\ell[1] \bigodot \ldots \bigodot C_\ell[n]$. It is easy to see that due to the homomorphic property C_{tally}^ℓ is a ciphertext that hides a integer value T_ℓ that equals the number of votes that were won by the predetermined election candidate $\ell \in \{1, \ldots, c\}$. Decrypting these ciphertexts reveals the votes accumulated by each predetermined candidate. Dealing with the write-in part of each vector-ballot is as in the paradigm of section 3.

Security and Efficiency. The security of the punch-hole/write-in version of our paradigm can be argued in similar terms as the main paradigm. Regarding efficiency, the main difference between the punch-hole paradigm and the general vector-ballot paradigm is that the encrypted ballot contains $c+2$ ciphertexts

instead of 3. While this may sound as a substantial increase in space it is not so: indeed, the security parameter in the vector-ballot paradigm (as well as in any homomorphic encryption scheme) must have a linear dependency on c, whereas the security parameter in the punch-hole approach is independent of c. Thus the two approaches do not differ (in the asymptotic sense) in terms of space. In terms of time-efficiency, the punch-hole approach requires more work from the voter in the proof of the vector-ballot consistency, but it yields a significant gain from the fact that the security parameter does not have to be proportional to the number of candidates and that tallying (as described below) can be done very efficiently over additive ElGamal encryption — in fact, an exponential gain.

Exponential gain for the additive-ElGamal instantiation. Observe that when the above protocol approach is instantiated with additive ElGamal encryption the announcement of the results requires c brute-force searches of a space of size n instead of a brute-force step of a space of size n^{c-1} as it is the case with previous ElGamal-based encryption-schemes (e.g. [10]). This emphasizes further the usefulness of the "punch-hole" approach to increase the efficiency of the system. We remark that this significant gain is independent of the addition of the writein part of the election and in fact it can be also executed in the non-writein setting of [10].

Acknowledgement. It has been brought to our attention that a scheme related to the punch-hole approach (without the combination of vector-ballots/ write-in votes, though) appeared in the Ph.D. Thesis of M. Hirt [23].

References

1. Masayuki Abe and Fumitaka Hoshino, *Remarks on Mix-Networks Based on Permutation Networks*, PKC 2001.
2. Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Guillaume Poupard and Jacques Stern, *Practical Multi-Candidate Election system*, In the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), 2001.
3. Josh Benaloh, *Verifiable Secret-Ballot Elections*, PhD Thesis, Yale University, 1987.
4. Josh Benaloh and Moti Yung, *Distributing the Power of a Government to Enhance the Privacy of Voters*, In the proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), 1986.
5. Josh Benaloh and Dwight Tuinstra, *Receipt-Free Secret-Ballot Elections*, STOC 1994.
6. Dan Boneh and Philippe Golle, *Almost Entirely Correct Mixing With Applications to Voting*, 9th ACM-CCS Conference, 2002.
7. David Chaum, *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*, Communications of the ACM 24(2): 84-88, 1981.
8. David Chaum, *Elections with Unconditionally-Secret Ballots and Disruption Equivalent to Breaking RSA*, Eurocrypt 1988.
9. Josh D. Cohen (Benaloh) and Michael J. Fischer, *A Robust and Verifiable Cryptographically Secure Election Scheme*, FOCS 1985.
10. Ronald Cramer, Rosario Gennaro and Berry Schoenmakers, *A Secure and Optimally Efficient Multi-Authority Election Scheme*, Eurocrypt 1997.
11. Ronald Cramer, Ivan Damgård and Berry Schoenmakers, *Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols*, Crypto 1994.

12. Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers and Moti Yung, *Multi-Autority Secret-Ballot Elections with Linear Work*, Eurocrypt 1996.
13. Ivan Damgård and Mats Jurik, *A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System*, Public Key Cryptography 2001, pp. 119-136.
14. Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, Moti Yung, *On Monotone Formula Closure of SZK*, FOCS 1994.
15. Amos Fiat and Adi Shamir, *How to Prove Yourself: Practical Solutions to Identification and Signature Problems*, Crypto 1986.
16. Pierre-Alain Fouque, Guillaume Poupard and Jacques Stern, *Sharing Decryption in the Context of Voting or Lotteries*, Financial Cryptography 2000.
17. Atsushi Fujioka, Tatsuaki Okamoto and Kazuo Ohta: *A Practical Secret Voting Scheme for Large Scale Elections*, ASIACRYPT 1992.
18. Jun Furukawa and Kazue Sako, *An Efficient Scheme for Proving a Shuffle*, CRYPTO 2001, pp. 368-387.
19. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk and Tal Rabin, *Secure Distributed Key Generation for Discrete-Log Based Cryptosystems* Eurocrypt 1999.
20. P. Golle, S. Zhong, D. Boneh, M. Jakobsson and A. Juels, *Optimistic Mixing for Exit-Polls*, Asiacrypt 2002.
21. Dimitris Gritzalis (Ed.), **Secure Electronic Voting**, Advances in Information Security, Volume 7, Kluwer 2002.
22. Martin Hirt and Kazue Sako, *Efficient Receipt-Free Voting Based on Homomorphic Encryption*, Eurocrypt 2000.
23. Martin Hirt, *Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting*, Ph.D. Thesis, ETH Zurich, 2001.
24. Markus Jakobsson, Ari Juels and Ronald L. Rivest *Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking*, USENIX Security Symposium 2002, pp. 339-353.
25. Byoungcheon Lee and Kwangjo Kim, *Receipt-Free Electronic Voting Scheme with Tamper-Resistant Randomizer*. ICISC 2001.
26. C. Andrew Neff, *A verifiable secret shuffle and its application to e-voting*, ACM Conference on Computer and Communications Security 2001, pp. 116-125.
27. Tatsuaki Okamoto, *Receipt-Free Electronic Voting Schemes for Large Scale Elections*, Workshop on Security Protocols, 1997.
28. Pascal Paillier, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, Eurocrypt 1999.
29. Choonsik Park, Kazutomo Itoh and Kaoru Kurosawa, *Efficient Anonymous Channel and All/Nothing Election Scheme*, Eurocrypt 1993.
30. Torben P. Pedersen, *A threshold Cryptosystem without a Trusted Third Party*, Eurocrypt 1991.
31. Torben P. Pedersen, *Distributed Provers and Verifiable Secret Sharing Based on the Discrete Logarithm Problem*, PhD Thesis, Aarhus University 1992.
32. Michael Rabin, *Transactions protected by beacons*, Journal of Computer and System Sciences, Vol. 27, pp 256-267, 1983.
33. Kazue Sako and Joe Kilian, *Secure Voting Using Partially Compatible Homomorphisms,* Crypto 1994.
34. Kazue Sako and Joe Kilian, *Receipt-Free Mix-Type Voting Scheme - A Practical Solution to the Implementation of a Voting Booth*, Eurocrypt 1995.
35. Berry Schoenmakers, *A Simple Publicly Verifiable Secret Sharing Scheme and its Applications to Electronic Voting*, Crypto 1999.