Multiplicative Homomorphic E-Voting*

Kun Peng[1], Riza Aditya[1], Colin Boyd[1], Ed Dawson[1], and Byoungcheon Lee[1,2]

[1] Information Security Research Centre
IT Faculty, Queensland University of Technology
{k.peng, c.boyd, e.dawson, r.aditya}@qut.edu.au
http://www.isrc.qut.edu.au
[2] Joongbu University, Korea
sultan@joongbu.ac.kr

Abstract. All the currently existing homomorphic e-voting schemes are based on additive homomorphism. In this paper a new e-voting scheme based on multiplicative homomorphism is proposed. In the tallying phase, a decryption is performed to recover the product of the votes, instead of the sum of them (as in the additive homomorphic e-voting schemes). Then, the product is factorized to recover the votes. The new e-voting scheme is more efficient than the additive homomorphic e-voting schemes and more efficient than other voting schemes when the number of candidates is small. Strong vote privacy and public verifiability are obtained in the new e-voting scheme.

1 Introduction

Two main methods have been applied to design e-voting schemes: mix network and homomorphic tallying. Both methods can protect vote privacy when threshold trust is assumed. In regard to efficiency, it is demonstrated in [2] that mix network is more suitable for elections with a large number of candidates or choices (e.g. preferential voting) and homomorphic tallying is more suitable for elections with a small number of candidates or choices (e.g. "YES/NO" voting) as the latter's cost is linear in the number of candidates or choices.

Current homomorphic e-voting schemes employ an additive homomorphic encryption algorithm (e.g. Paillier encryption) to encrypt the votes and exploit additive homomorphism of the encryption algorithm to recover the sum of votes for any candidate or choice with a single decryption. As no single vote is decrypted, vote privacy is protected. It is surprising that multiplicative homomorphism has never been employed to design any voting scheme, although it may lead to better performance.

The contribution of this paper is a design of a multiplicative homomorphic voting scheme. In a multiplicative homomorphic voting scheme, a multiplicative homomorphic encryption algorithm (e.g. ElGamal encryption) to encrypt the votes and a single decryption is performed to calculate the product of votes.

* The research in this paper was supported by Australian Research Grants DPO345458 and LX0346868

Then the product is factorized and the votes are recovered. Like in additive homomorphic voting, no single vote is decrypted in multiplicative homomorphic voting, so vote privacy is protected too. The most important advantage of multiplicative homomorphic voting is that it is always more efficient than additive homomorphic voting and more efficient than other voting schemes when the number of candidates is small. In brief, multiplicative homomorphic voting improves efficiency without compromising vote privacy or public verifiability.

2 Related Work

In an election, the voters select a certain number of winners from a few candidates. At first the identities of the candidates and the number of expected winners are declared. Then every bidder appoints some candidates in his bid, whose number is equal to the number of expected winners. Finally some talliers count the votes and declare the voting result. In an e-voting system, tallying must be performed without revealing any vote.

Definition 1 *If after the voting every vote is only known to distribute uniformly in the vote space (containing all the possible choices), we say that* ***complete vote privacy*** *is achieved. If after the voting every voter's choice is only known to be among a large number of published votes, whose number is much larger than the number of possible choices, we ay that* ***strong vote privacy*** *is achieved.*

So far, two methods have been employed to protect vote privacy in voting schemes. The first one is mix network. In voting schemes using this method [15, 34, 28, 31, 32, 27, 20, 7, 17, 36, 1, 23, 9] (called mix voting), the votes are shuffled in the mix network and then decrypted separately. Although every single vote is decrypted, they cannot be linked to the voters after being shuffled. So, vote privacy is achieved. The second is homomorphic tallying, which exploits the homomorphism of the encryption algorithm (used to encrypt the votes) to implement the tallying without decrypting any single vote. E-voting schemes employing the second method are called homomorphic voting and include [18, 5, 33, 6, 11, 12, 3, 26, 35, 19, 4, 21, 13, 22, 24, 25]. Since no single vote is decrypted, vote privacy is obtained. Homomorphic voting schemes are efficient when the number of candidates or choices is small. However, homomorphic voting has a drawback: each vote must be verified to be valid. Without the vote validity check, correctness of the tallying cannot be guaranteed. When the number of candidates or choices is large (e.g. in a preferential voting), computational and communicational cost for the proof and verification of vote validity is so high that homomorphic voting becomes less efficient than mix voting. So, it is widely believed that homomorphic voting is only suitable for elections with a small number of candidates or choices (e.g. "YES/NO" voting).

An encryption scheme is additive homomorphic if $E(m_1+m_2) = E(m_1)E(m_2)$ for any messages m_1 and m_2 where $E()$ stands for the encryption function. In an additive homomorphic voting scheme, each bidder makes a choice for every candidate (1 for the candidate or 0 against the candidate), encrypts his choices as his encrypted vote. Then he proves that his vote is valid, namely every choice encrypts 0 or 1 and the number of 1s encrypted in his vote is equal to the expected number of winners specified in the voting rule. The talliers verify that each vote is valid. Then they decrypt the product of encrypted choices for each candidate or the product of all the encrypted votes (in some special voting schemes [24, 25] each voter combines his choices for all the candidates in one ciphertext) to find out the sum of votes for each candidate without decrypting any single vote.

One of two possible additive homomorphic encryption algorithm are usually employed: Paillier encryption or modified ElGamal encryption. Paillier encryption is inherently additive homomorphic and more frequently applied. The original ElGamal encryption scheme can be simply modified to be additive homomorphic: a message is used as an exponent in an exponentiation computation, then the exponentiation is encrypted using the original ElGamal encryption. A passive result of this modification is that a search for logarithm must be performed in the decryption function, which becomes inefficient when the searching space is not too small. The modified ElGamal encryption is employed in homomorphic voting schemes [18, 22, 24, 25], where the details of the modification and the consequent search are described in detail.

A disadvantage of additive homomorphic voting compared to multiplicative homomorphic voting is inefficiency due to the following reasons.

– If Paillier encryption is employed, the following drawbacks in efficiency exist.
  - Inefficient set-up
    In voting schemes, the private key of the encryption algorithm must be generated and shared by multiple talliers, so that it is not needed to trust any single party to achieve vote privacy. As the private key is a factorization secret in Paillier encryption, distributed key generation is highly inefficient. In comparison, distributed key generation in ElGamal (distributed generation of a secret logarithm as the private key) is much more efficient as described in [14, 30, 16].
  - Multiple encryption
    Usually, a voter has to perform an encryption for each candidate and prove each of his encryptions contains a valid message.
  - Inefficiency of multiplicative and exponentiation computations
    In Paillier encryption, each multiplication is performed modulo N^2 where N is the product of two large primes and its factorization is the private key (see [29] for details). In comparison, in original ElGamal encryption, each multiplication is performed modulo p, a large prime. If the same security strength is required, N and p should have the same length (e.g. 1024 bits). As the modulus in Paillier encryption scheme is a square and usually the computation for modular multiplication is quadratic in the operand size, multiplication in Paillier encryption scheme is more costly than that in ElGamal encryption scheme. Although Chinese Remainder Theorem can be employed to improve the efficiency of multiplicative computation with a composite modulus in Paillier encryption scheme, Paillier admitted this efficiency improvement is only available in key

generation and decryption when the factorization of N is known. Paillier indicated that a multiplication in Paillier encryption is more than three times as costly as a multiplication in ElGamal encryption when N and p should have the same length (e.g. 1024 bits). Usually distributed decryption is employed in voting schemes to minimize trust and strengthen robustness, so the factorization of N is not known to any single tallier, who performs the decryption. Therefore, we can assume that when the same security strength is required a multiplication in Paillier encryption with distributed decryption is at least three times as costly as a multiplication in ElGamal encryption.

– If the modified ElGamal encryption is employed, the following drawbacks in efficiency exist.
  - Multiple encryption
    Usually, a voter has to perform an encryption for each candidate and prove each of his encryptions contains a valid message.
  - Inefficient DL search
    As stated before, a search for logarithm is needed in the decryption function. Even though the (currently known) most efficient solution for DL in a certain interval — Pollard's Lambda Method — is employed, $0.5 \log_2 n$ exponentiations, $O(n^{0.5})$ multiplications and $O(0.5 \log_2 n)$ storage are needed where n is the number of voters. As the number of voters is often large in voting applications, this is a high cost. To make the search more efficient, the votes may be divided into multiple groups and a separate tallying is performed in each group. However, this division increases the number of decryptions as a separate decryption is needed for every candidate in each group.

  In [24, 25], the modified ElGamal encryption and its additive homomorphism are exploited in a very special way. Only one encryption is needed in a vote, which is composed of several sections, each corresponding to one candidate. So only one decryption is needed to decrypt the product of all the encrypted votes. Although the numbers of encryptions and decryptions are reduced, they are not the main computational burden in the voting scheme. The main computational burden of the voting scheme increases as the computational cost for vote validity proof increases and the cost of the DL search increases to $O(mn^{m-1})$ multiplications and $O(nm)$ full length (e.g. 1024 bits) storage space where m is the number of candidates. As the number of voters is often large in voting applications, the cost for the search is intolerable. So the special additive homomorphic tallying in [24, 25] actually deteriorates efficiency although it was supposed to improve efficiency.

In comparison, as will be illustrated in Section 3, multiplicative homomorphic voting employs efficient distributed key generation, requires only one encryption per vote and needs no DL search, while it achieves vote privacy no weaker than that of additive homomorphic voting.

3 The Multiplicative Homomorphic Voting Scheme

A multiplicative homomorphic voting scheme exploits multiplicative homomorphism of the encryption algorithm used for vote encryption to tally efficiently without revealing any vote. Each voter only needs to encrypt with a multiplicative homomorphic encryption algorithm one integer as his vote. An encryption algorithm is multiplicative homomorphic if $E(m_1 m_2) = E(m_1)E(m_2)$ where $E()$ stands for the encryption function and m_1,m_2 are two random messages. A typical multiplicative homomorphic encryption algorithm is ElGamal encryption, which is employed in this paper. The product of the encrypted votes are then decrypted, so that the product of the votes is obtained if their product is not over the multiplicative modulus (certain mechanism is used to guarantee this assumption). Then the product is factorized to recover the votes. A voting protocol to elect one winner from m candidates is as follows.

1. Preparation phase
   Suppose there are m candidates $C_1, C_2, \ldots, C_m$. ElGamal encryption modulo p is employed for vote encryption where $p = 2q + 1$ and p, q are large primes. Several talliers cooperate to generate and threshold share the private key while the public key is published using the distributed key generation function in [16]. A set $Q = \{q_1, q_2 \ldots, q_m\}$ is chosen to represent the candidates as follows.
   (a) Two sets $Q_1 = \{1\}$ and $Q_2 = \Phi$ are initialised. Two integers s_1 and s_2 representing the sizes of the two sets respectively are initialised as $s_1 = 1$ and $s_2 = 0$. Index s is initialised to be 1.
   (b) The s^{th} smallest prime p_s is tested.
   If $p_s^q = 1 \bmod p$,
      – p_s is a quadratic residue;
      – p_s is put into Q_1 and set $s_1 = s_1 + 1$.
   If $p_s^q \neq 1 \bmod p$,
      – p_s is not a quadratic residue;
      – p_s is put into Q_2 and set $s_2 = s_2 + 1$.
   (c) If $s_1 < m$ and $s_2 < m$, set $s = s + 1$ and go to Step (b). Otherwise, go to next step.
   (d) If $s_1 = m$, $Q = Q_1$; If $s_2 = m$, $Q = Q_2$.
   With this setting-up, the members in Q are either all quadratic residues or all quadratic non-residues, so their encryptions are indistinguishable[3].
   The talliers set up the ElGamal encryption:
   – they cooperatively generate the public key g and y in G, which is the subgroup in Z_p^* with order q using the distributed key generation techniques in [14, 30, 16], such that the private key $x = \log_g y$ are shared by them;

[3] An alternative method to generate Q is to choose p and q such that the $m - 1$ smallest primes are quadratic residues modulo p. Different large primes are tested as possible choices of p until a satisfying p is found. So, Q contains 1 and the $m - 1$ smallest primes. However, it is not clear whether this method is feasible or efficient, especially when m is large.

   – public key g and y are published.
2. Voting phase
   Each of the n voters $V_1, V_2, \ldots, V_n$ chooses a vote from Q. Voter V_i encrypts his vote v_i to $c_i = E(v_i) = (a_i, b_i) = (g^{r_i}, v_i y^{r_i})$ where r_i is randomly chosen from Z_q. V_i proves that an element in Q is encrypted in c_i without revealing his vote using the following honest-verifier ZK proof:

   $$\log_g a_i = \log_y(b_i/q_1) \ \lor \ \log_g a_i = \log_y(b_i/q_2) \ \lor \ldots \lor \ \log_g a_i = \log_y(b_i/q_m)$$

   This proof is based on the ZK proof of equality of logarithms [8] and the ZK proof of partial knowledge [10].
3. Tallying phase
   The talliers verify that every vote is valid. Then they randomly divide the encrypted votes $c_1, c_2, \ldots, c_n$ to groups of size k, so that $Max(Q)^k < p$ where $Max(Q)$ stands for the largest element in set Q. If $Max(Q)^n < p$, the division is not necessary and all the votes are in the same group. In each group the following multiplicative homomorphic tallying is performed.
   (a) Suppose $c'_1, c'_2, \ldots, c'_k$ are the encrypted votes in a group.
   (b) The talliers cooperate to calculate $v = D(\prod_{i=1}^{k} c'_i)$ where $D()$ denotes decryption.
   (c) v is factorized[4].
      – If $1 \notin Q$, $v = \prod_{j=1}^{m} p_j^{t_j}$ and the number of votes in this group for the j^{th} candidate is t_j for $j = 1, 2, \ldots, m$.
      – If $1 \in Q$, $v = \prod_{j=1}^{m-1} p_j^{t_j}$ and the number of votes in this group for the j^{th} candidate is t_{j-1} for $j = 2, 3, \ldots, m$ while the number of votes in this group for the first candidate is $k - \sum_{j=1}^{m-1} t_j$.

   The talliers sum up the results in all the groups to get the final result.

4 Analysis

The new voting scheme is analysed in this section to show that it is correct and efficient.

Theorem 1. *The multiplicative homomorphic tallying in each group with encrypted votes $c'_1, c'_2, \ldots, c'_k$ is correct.*

Proof: In the multiplicative homomorphic tallying in each group with encrypted votes $c'_1, c'_2, \ldots, c'_k$,

$$D(\prod_{i=1}^{k} c'_i) = v = \prod_{j=1}^{m-1} p_j^{t_j}$$

where $\prod_{j=1}^{m-1} p_j^{t_j}$ is a factorization of v.
As ElGamal encryption is multiplicative homomorphic,

$$D(\prod_{i=1}^{k} c'_i) = \prod_{i=1}^{k} D(c'_i) \bmod p$$

[4] This factorization is very efficient as each prime in Q is very small.

When the encrypted votes are divided into groups, it is guaranteed that $Max(Q)^k < p$. So $\prod_{i=1}^{k} D(c'_i) < p$ Therefore,

$$\prod_{i=1}^{k} D(c'_i) = D(\prod_{i=1}^{k} c'_i) = \prod_{j=1}^{m-1} p_j^{t_j}$$

As $D(c'_i)$ for $i = 1, 2, \ldots, k$ are verified to be in Q in the voting phase, $\prod_{i=1}^{k} D(c'_i)$ is also a factorization of v.

As there is a unique factorization for any integer, $\prod_{i=1}^{k} D(c'_i)$ and $\prod_{j=1}^{m-1} p_j^{t_j}$ are the same factorization. Namely, each prime factor in $\prod_{i=1}^{k} D(c'_i)$ is also a prime factor in $\prod_{j=1}^{m-1} p_j^{t_j}$ and each prime factor in $\prod_{j=1}^{m-1} p_j^{t_j}$ is also a prime factor in $\prod_{i=1}^{k} D(c'_i)$.

Therefore, all the non-one votes encrypted in $c'_1, c'_2, \ldots, c'_k$ and only these votes are prime factors in $\prod_{j=1}^{m-1} p_j^{t_j}$. That means every non-one vote is correctly recovered.

As the number of vote in each group is a constant k, the number of "1" votes is also correctly recovered if there are any. □

Theorem 2. *Multiplicative homomorphic tallying does not reveal any vote.*

Sketch of proof:

- Semantically secure encryption
  The usage of ElGamal encryption in this paper is semantically secure due to the choice of message space Q. (Either all members are quadratic residues or no member is quadratic residue where $p = 2q + 1$) So, without the private key to decrypt the votes, it is difficult to get any information about any vote.
- Private key (decryption) security
  As the private key is protected by a threshold key sharing mechanism, no single vote is decrypted if a threshold trust on the talliers is assumed.
- Unlinkability (No bidder can be linked to his bid.)
  As a result, the only message decrypted from the encrypted votes is the product of votes in each group, which links no vote to the corresponding voter. The revealed information tells no more than that a voter in every group may have submitted any vote in the group.
- The group size is large enough for strong vote privacy.
  As homomorphic tallying is only applied to elections with a small number of candidates, m and $Max(Q)$ are small[5]. As p is large (e.g. with a length of 1024 bits), $\lceil \log_{Max(Q)} p \rceil$, the size of a group, is large compared to m where $\lceil x \rceil$ denotes the smallest integer no smaller than a real number x. For example, when $m = 2$ and $|p| = 1024$ where $||$ stands for bit length, we get $Q = \{1, 2\}$ (for simplicity, assuming 2 is a quadratic residue), $Max(Q) = 2$

[5] $Max(Q)$ is no larger than the $(2m-1)^{th}$ smallest prime, which is no more than a small multiple of m when m is small.

and the group size is larger than 1024. When there are only two candidates and more than 1024 votes are mixed together in each group, strong vote privacy is achieved.

□

Every operation in the voting scheme can be publicly verified by anyone. (Note that public proofs of vote validity and correctness of decryption are provided by the voters and talliers respectively.) The computational cost of additive homomorphic voting employing Paillier encryption and that of the proposed multiplicative homomorphic voting are listed in Table 1. As stated in Section 2, the DL search in the decryption of the modified ElGamal encryption in [18, 22, 24, 25] is too inefficient[6]. So the modified ElGamal encryption is not considered as a good choice in additive homomorphic voting. As only small primes are employed to stand for the votes, the computational cost of the final factorization in multiplicative homomorphic voting is negligible compared to full length exponentiation. To make a precise comparison of the efficiency of the two kinds of homomorphic voting, it is supposed the same strength of encryption security is required in both kinds of voting, so N in Paillier encryption employed in additive homomorphic voting and p in ElGamal encryption employed in multiplicative homomorphic voting have the same length. An exponentiation in multiplicative homomorphic voting (employing ElGamal encryption) is called a standard exponentiation while an exponentiation in additive homomorphic voting (employing Paillier encryption with distributed decryption) is accounted as three standard exponentiations. The number of standard exponentiations is accounted in every operation in Table 1. This table clearly illustrates that multiplicative homomorphic voting is always more efficient than additive homomorphic voting in key generation, vote encryption and vote validity check. When the number of voters is not too large, multiplicative homomorphic voting is also more efficient than additive homomorphic voting in tallying. For example, when $m = 2$, $|p| = 1024$ and $n = 1024$, the needed number of standard exponentiation in tallying in additive homomorphic voting is 12 or 6, while the needed number of standard exponentiation in tallying in multiplicative homomorphic voting is 3. Even if multiplicative homomorphic tallying is less efficient than additive homomorphic tallying when the number of voters is large, it has a trivial influence on the total cost of the voting scheme as will be shown in Table 2.

A more comprehensive efficiency comparison is presented in Table 2, where the efficiency of MV (mix voting), AHV (additive homomorphic voting) and the proposed MHV (multiplicative homomorphic voting) are compared. In this comparison, [17] (one of the most efficient mix voting) is taken as an example of mix voting where ElGamal encryption is employed. It is assumed that the additive

[6] Although some computation in the Pollard's Lambda Method can be pre-computed, precomputation can be employed in most voting schemes. For example, the exponentiation computation in vote encryption and all the computation in the proof of vote validity (if necessary) can be precomputed in mix voting, Paillier-based additive homomorphic voting and multiplicative homomorphic voting.

	Additive homomorphic voting	Multiplicative homomorphic voting
Distributed key generation	highly inefficient	efficient
Encryption per vote	$6m$	2
Vote validity proof per vote	$12m + 6$	$4m - 2$
Vote validity verification per vote	$12m + 6$	$4m$
Tallying computation per tallier	$9m$ or[a] $9(m - 1)$	$3\lceil n \log_p Max(Q) \rceil$

[a] It is often assumed that a decryption is necessary for every candidate. However, when n, the total number of voters is known and each vote has been verified to be valid, $m - 1$ decryptions are enough. The talliers randomly choose $m - 1$ candidates and decrypt the sum of votes for each of them. The vote of the remaining candidate is n minus the sum of the votes for the $m - 1$ chosen candidates. We call this economical tallying

Table 1. Computational cost of the two kinds of homomorphic voting

homomorphic voting (no existing example is referred to) employs distributed Paillier encryption and performs every necessary operation listed in Table 1. For simplicity, voters' signature on the votes are omitted, so voters' signature generation and talliers' signature verification are not taken into account. In Table 2, t is the number of talliers. The number of standard exponentiation is accounted in computational cost and the number of transported bits is accounted in communicational cost. An example is given in Table 2, where $t = 5$, $m = 2$, $|p| = 1024$ and $n = 1000000$. For simplicity, it is assumed that 2 is a quadratic residue modulo p, so $Q = 1, 2$. In this example, it is shown that even when the number of voters is large, multiplicative homomorphic voting is still more efficient than mix voting and additive homomorphic voting. When the number of voters is not large and grouping is not necessary in tallying, efficiency advantage of multiplicative homomorphic voting is more obvious. So multiplicative homomorphic voting is the most efficient voting solution when the number of candidates is small.

5 Conclusion

The voting scheme in this paper employs a new tallying method: multiplicative homomorphic tallying. It achieves the highest efficiency when the number of candidates is small and guarantees strong vote privacy and public verifiability.

	Key generation	Voter's computation	Tallier's computation	Communication
MV	efficient	8	$18n$ $= 18000000$	$1024(6n + 18tn)$ $=98304000000$
AHV[a]	highly inefficient	$18m + 6$ $= 42$	$(12m + 6)n$ $+9(m - 1)$ $= 30000009$	$2048(6t(m - 1)+$ $(10m + 4)n)$ $=49152061440$
MHV	efficient	$4m$ $= 8$	$4mn + 3\lceil n$ $\log_p Max(Q)\rceil$ $= 8003000$	$1024(2n(m + 1)+$ $3\lceil n \log_p Max(Q)\rceil)$ $=6147072000$

[a] It is assumed economical tallying in Table 1 is employed.

Table 2. Efficiency comparison

References

1. Masayuki Abe and Hideki Imai. Flaws in some robust optimistic mix-nets. In *Advances in Cryptology—ACISP 03*, pages 39–50, 2003.
2. R. Aditya, C. Boyd, E. P. Dawson, and K. Viswanathan. Secure e-voting for preferential elections. In *Proceedings of EGOV 03 Conference*, pages 246–249, Berlin, 2003. Springer-Verlag. Lecture Notes in Computer Science Volume 2738.
3. James M. Adler, Wei Dai, Richard L. Green, and C. Andrew Neff. Computational details of the votehere homomorphic election system. Technical report, VoteHere Inc, 2000. Available from http://www.votehere.net/technicaldocs/hom.pdf, last accessed 22 June 2002.
4. Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard. Practical multi-candidate election system. In *Twentieth Annual ACM Symposium on Principles of Distributed Computing*, pages 274–283, 2001.
5. Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 544–553, 1994.
6. Josh Daniel Cohen Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Faculty of Graduate School, Yale University, 1996.
7. Dan Boneh and Philippe Golle. Almost entirely correct mixing with applications to voting. In *9th ACM Conference on Computer and Communications Security—CCS 02*, pages 68–77, 2002.
8. D. Chaum and T. P. Pedersen. Wallet databases with observers. In *CRYPTO '92*, pages 89–105, Berlin, 1992. Springer-Verlag. Lecture Notes in Computer Science Volume 740.
9. David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38–47, January/February 2004.
10. R. Cramer, I. B. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94*, pages 174–187, Berlin, 1994. Springer-Verlag. Lecture Notes in Computer Science Volume 839.

11. Ronald Cramer, Matthew Franklin, Berry Schoenmakers, and Moti Yung. Multi-authority secret-ballot elections with linear work. In *Advances in Cryptology—EUROCRYPT 96*, pages 72–83, 1996.
12. Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology—EUROCRYPT 97*, pages 103–118, 1997.
13. Ivan Damgaård and Mats Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography—PKC 01*, pages 119–136, 2001.
14. P Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science*, pages 427–437, 1987.
15. Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology—AUSCRYPT 92*, pages 244–251, 1992.
16. R Gennaro, S Jarecki, H Krawczyk, and T Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *EUROCRYPT '99*, pages 123–139, Berlin, 1999. Springer-Verlag. Lecture Notes in Computer Science Volume 1592.
17. Philippe Golle, Sheng Zhong, Dan Boneh, Markus Jakobsson, and Ari Juels. Optimistic mixing for exit-polls. In *Advances in Cryptology—ASIACRYPT 02*, pages 451–465, 2002.
18. Alejandro Hevia and Marcos Kiwi. Electronic jury voting protocols. 2000. http://eprint.iacr.org/2000/035/.
19. Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *Advances in Cryptology—EUROCRYPT 00*, pages 539–556, 2000.
20. Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *11th USENIX Security Symposium*, pages 339–353, 2002.
21. Jonathan Katz, Steven Myers, and Rafail Ostrovsky. Cryptographic counters and applications to electronic voting. In *Advances in Cryptology—EUROCRYPT 01*, pages 78–92, 2001.
22. Aggelos Kiayias and Moti Yung. Self-tallying elections and perfect ballot secrecy. In *Public Key Cryptography, 5th International Workshop—PKC 02*, pages 141–158, 2002.
23. Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *to appear in Information Security and Cryptology, ICISC 2003*, 2003.
24. Byoungcheon Lee and Kwangjo Kim. Receipt-free electronic voting through collaboration of voter and honest verifier. In *JW-ISC 2000*, pages 101–108, 2000.
25. Byoungcheon Lee and Kwangjo Kim. Receipt-free electronic voting scheme with a tamper-resistant randomizer. In *Information Security and Cryptology, ICISC 2002*, pages 389–406, 2002.
26. C. Andrew Neff. Conducting a universally verifiable electronic election using homomorphic encryption. White paper, VoteHere Inc, 2000.
27. C. Andrew Neff. Verifiable, secret shuffles of elgamal encrypted data for secure multi-authority elections. In *8th ACM Conference on Computer and Communications Security—CCS 01*, pages 116–125, 2001.
28. Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proc. Security Protocols, 5th International Workshop 1997*, pages 25–35, 1997.
29. P Paillier. Public key cryptosystem based on composite degree residuosity classes. In *EUROCRYPT '99*, pages 223–238, Berlin, 1999. Springer-Verlag. Lecture Notes in Computer Science Volume 1592.

30. Torben P. Pedersen. A threshold cryptosystem without a trusted party. In *EUROCRYPT '91*, pages 522–526, Berlin, 1991. Springer-Verlag. Lecture Notes in Computer Science Volume 547.
31. Andreu Riera and Joan Borrell. Practical approach to anonymity in large scale electronic voting schemes. In *Network and Distributed System Security Symposium—NDSS 99*, pages 69–82, 1999.
32. Andreu Riera, Josep Rifà, and Joan Borrell. Efficient construction of vote-tags to allow open objection to the tally in electronic elections. *Information Processing Letters*, 75(5):211–215, October 2000.
33. Kazue Sako and Joe Kilian. Secure voting using partially compatible homomorphisms. In *Advances in Cryptology—CRYPTO 94*, pages 411–424, 1994.
34. Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme: A practical solution to the implementation of a voting booth. In *Advances in Cryptology—EUROCRYPT 95*, pages 393–403, 1995.
35. Berry Schoenmakers. Fully auditable electronic secret-ballot elections. *XOOTIC Magazine*, July 2000.
36. Douglas Wikström. How to break, fix and optimize "optimistic mix for exit-polls". Technical report, Swedish Institute of Computer Science, 2002. Available from http://www.sics.se/libindex.htlm, last accessed 08 October 2003.