An Efficient Homomorphic Coercion Resistant and E2E Verifiable Voting Scheme

Vinodu George (Corresponding author)
Department of Computer Science and Engineering
National Institute if Technology, Calicut, Kerala, India
E-mail: vinodu.george@gmail.com

M. P. Sebastian
Information Technology & Systems Area
Indian Institute of Management Kozhikode, Kerala, India
E-mail: sebasmp@iimk.ac.in

Abstract

This paper proposes a voting scheme that coalesces many features of an efficient e-voting scheme like receipt-freeness, uncoercibility, E2E verifiable and write-in ballot. Some of the previous schemes in the past literatures provide most of these features at the cost of increased running time. This paper proposes a simple and efficient voting scheme that is coercion resistant and E2E verifiable. It also has the write-in property. Our protocol addresses some of the major drawbacks of previous popular voting schemes to make the proposed scheme applicable for any kind of real time elections.

Keywords: Cryptography, Electronic Voting, Receipt-Freeness, Write-in Ballots, Coercion-Resistance

1. Introduction

Electronic voting promises a convenient and efficient facility for recording and tallying votes. Lot of literatures has been developed on electronic voting over the last two decades. Several of these schemes concentrate mainly on security of electronic voting (Sampigethaya & Poovendran, 2006) The use of Internet and incorrect implementations of the schemes has resulted in many security violations. This gives a scope for lot of rework in many of the schemes. The desirable features of an efficient e-voting scheme includes receipt-freeness, uncoercibility and write-in ballot. An election protocol is said to be coercion-resistant if a voter V cannot prove to a potential coercer C that the voter V voted in a particular way. The write-in ballots allow voters to choose their own ballots. Some of the popular voting schemes satisfy most of these requirements at the cost of increased running time. This paper proposes a voting scheme with reduced running time, satisfying most of the desirable features of the existing write-in and coercion resistant voting scheme.

1.1 Related Work

One of the main requirements of a voting protocol is to ensure the privacy of a voter. Receipt-freeness, a stronger notion of privacy, restricts a voter to show others how he has voted. The first receipt-free protocol appeared in (Benaloh, J. & Tuinstra, D., 1994). Naturally receipt-freeness reduces vote buying and selling. Coercion resistant protocols, introduced by (Juels, Catalano & Jakobsson, 2005) guarantees that any powerful adversary cannot be convinced on the voting pattern even with the help of the voter. A drawback of this scheme is the overhead for tallying by the authorities is quadratic with the number of voters. Processing V votes by N voters requires $O(NV)$ steps for whole verification, which needs a minimum of N^2 steps. This makes the scheme is practicable only for small elections. Moreover, this scheme does not discuss the possibility of write-in ballots also.

(Kiayias & Yung 2004) proposed a vector ballot scheme, which allows write-in ballots. But this scheme does not have coercion resistance feature. (Acquisti, 2004) proposed a voting scheme, which guarantees receipt-freeness, uncoercibility and write-in ballot. But for validating and tallying of votes, this scheme requires all possible ballots for each credential. So the time complexity of this scheme is even much higher than that of the scheme proposed by (Juels, Catalano & Jakobsson, 2005) and hence is practicable only for small elections.

In recent years, a new class of voting scheme has been proposed in which the voter will be issued with a receipt after interaction with the voting system. This receipt can be used to check with a set of encrypted receipts published by the authority in the Bulletin Board. Also these receipts can be used by the voter to verify whether the vote cast by him is counted in the final tally. But it can’t be used to prove the manner in which he/she cast the vote. This type of voting scheme is called end-to-end or E2E voting system (Adida, B., 2006). Many of the E2E voting schemes like Punch scan (Popoveniuc, S. & Hosp, B., 2006), Scratch & Vote (Adida, B. & Rivest, R., 2006) and Three-ballot (Rivest, R., 2006) are paper based voting schemes. But even in such paper based voting schemes, coercion resistance is the most desirable feature to make a secure voting scheme.

So this paper proposes a simple and practical voting scheme that is both E2E verifiable and coercion resistant. It also has the write-in property (provision for casting the vote in favor of a pseudo-candidate). Since this scheme uses the Paillier crypto system (Pascal Paillier, 1999), the tallying of votes can also be done easily. Our protocol also addresses some of the major drawbacks of (Kiayias & Yung, 2004), (Juels, Catalano & Jakobsson, 2005) and (Acquisti, 2004). The proposed scheme is similar to (Acquisti, 2004), but it takes less amount of time to perform the entire cross checks and elimination of duplications.

1.2 Comparison with Acquisti's Scheme

In Acquisti's scheme, the search space contains the entire vote cast, which is a variable. A voter can vote any number of votes, but only the last cast vote can be taken for tally. So the number of votes cast may become very large and thus the search space may also be huge. Whereas in the proposed scheme, search space contains only the issued credentials, which is fixed and only the encrypted credentials are compared for validation. In Acquisti's scheme, each valid credential is operated with all ballots and the resulting products are compared with each of the vote cast. But in the proposed scheme, a search tree is constructed for the issued credentials before the voting process, which would make the validation and tallying simple. The construction of such a tree is not possible in Acquisti's scheme, because the search space varies dynamically. Also in Acquisti's scheme, validation and tallying phase is very complex as compared to the proposed scheme. Another variation is that in the former scheme, the selection of last cast vote is done separately from the validation phase whereas in the proposed scheme it is done along with the validation phase. Other benefits of the proposed scheme over Acquisti's scheme are: The proposed scheme needs only $O\ (n \log n)$ steps to perform the entire verification and elimination of duplications, but in Acquisti's scheme, it will be in the order of $O\ (n^3)$. In the proposed scheme, the elimination of duplicate votes and validation of credentials can be performed in a single step, which will further reduce the running time. Since this scheme uses homomorphic encryption, the tallying time also gets reduced. Moreover, this scheme allows multiple casting so that a coerced voter gets a chance to cast an intended vote.

The rest of the paper is organized as follows: 'Section 2' describes the voting model and the voting lifecycle. 'Section 3' presents the voting protocol. 'Section 4' evaluates the performance of the proposed scheme and 'Section 5' concludes the paper.

2. The Voting Model

An election system consists of the following sets of entities:

Authorities: Denoted by $A = \{A_1, A_2, \ldots A_{nA}\}$, are responsible for jointly issuing keying material, i.e. credential and the candidate slate (Ballot) to the voters. See Brands (2002) for more details on credentials.

Validator: Denoted by D, is responsible for the construction of the Binary Search Tree (*BST*) and the validation of votes.

Talliers: The set of n_T Talliers is denoted by $T = \{T_1, T_2, \ldots T_{nT}\}$, are responsible for mixing the ballots, jointly counting the votes and publishing the final tally.

Voters: The set of n_V voters are denoted by $V = \{V_1, V_2, \ldots V_{nV}\}$. They are the entities participating in the given election.

2.1 The Voting Life Cycle

A simplified life cycle model of an electronic voting scheme consists of four phases:

Setup: This phase initializes the technical part of the election system and the organizational structure.

Voting: Votes are cast in this phase. The voter's digital ballot is to be authenticated maintaining ballot secrecy.

Validation: This phase validates all the votes; and the invalid votes are discarded.

Tallying: This phase finalizes the election result from the valid votes.

3. Voting Scheme

The authority includes the independently functioning servers that manage the registration and establish the cryptographic primitives. During the registration phase, each responsible authority creates the shares of voting credential for each eligible voter. Each authority posts these credential shares on the Bulletin Board (*BB*), encrypted with both authority's and voter's public key. These shares will be sent to the intended voters (along with the proof) and also to the validator (after proper encryption). The authority also publishes the candidate slate on the *BB*. The validator multiplies the shares of the encrypted credential and creates the *BST* with the search key as the credential encrypted with authority's public key.

Each voter multiplies the decrypted shares which he has obtained from the *BB,* using the homomorphic property of the Paillier cryptosystem (Pascal Paillier, 1999) with his private key. The voter sends the resulting credential (encrypted with authority's public key) along with the candidate choice (encrypted with tallier's public key) to the *BB*. At the end of the voting phase, the tallier mixes all ciphertext posted by the eligible voters, and the validator stores the verified ballots in the *BST*. The tallier is responsible for tallying the votes stored in the *BST*.

3.1 The Voting Protocol

Setup: Two sets of Paillier public/private key pairs are generated for the authority in a reliable manner as in (Acquisti, 2004).

- The first set is for publishing the credential on the *BB*: (PK^{Ac} , SK_j^{Ac} , VK^{Ac} , VK_j^{Ac})
- The second set is for communicating the credential to the Voter: (PK^{Av} , SK_j^{Av} , VK^{Av} , VK_j^{Av})

(Where *PK* represents the public key; *SK,* the secret key and *VK,* the verification key)

All the public t keys are published on the *BB*. The list of permissible candidates slate is defined as $B^t = \{B^1, B^2, \ldots, B^{n_c}\}$, where B is an integer with the property $B > n_V$ (number of voters) and n_c is the number of candidates (Published on the *BB* before the commencement of the election). Each election authority A_j creates its own share of ballot for each of the permissible ballots, $B^t{}_j$. After proper encryption, all these ballot shares will be published in the specific location of the *BB* and would also be sent to each voter with the verification proof (just like the credential shares).

Registration: Each A_j uses a random number generator to schedule the generation of credential shares for voters. The generated random number i is given to each A_j to generate the credential share for voter V_i. The generated share will be posted on the *BB* in the area reserved for voter V_i. The credential share which is generated in random order is then issued to the validator in the same order so that the validator cannot connect the credentials with the voter.

The Authority A_j generates the string σ_{ij} for voter V_i that serves as the credential of voter V_i. More details about the creation of credential is given in (Brands, 2002). For each σ_{ij} it creates, A_j encrypts σ_{ij} using the public key PK^{Ac} with proper secret randomization; then signs the ciphertext with the secret key SK_j^{Ac} and publishes it on the designated location of the *BB* for the shares of credential of voter V_i. This data is

$$E\,(E(\sigma_{ij})_{PK}{}^{Ac})_{SKj}{}^{Ac} \tag{1}$$

The authority A_j again encrypts σ_{ij} using PK^{Av} and attaches a verifier proof PV_i to prove that $E(\sigma_{ij})_{PK}{}^{Av}$ and $E(\sigma_{ij})_{PK}{}^{Ac}$ are ciphertexts of σ_{ij}. The resulting message is encrypted with V_i's public key, PK^{Vi} and sent to the space allocated for V_i in the *BB* by the authority. This data is

$$E\,\left(E(\sigma_{ij})_{PK}{}^{Av},\ PV_i\right)_{PK}{}^{Vi} \tag{2}$$

Once again, the each authority A_j encrypts σ_{ij} using PK^{Av} without attaching any verifier proof. This ciphertext is again encrypted with the validator's public key PK^{VD} and is sent to the validator through a mixnet (Lee, B., Boyd, C., Dawson, E., Kim, K., Yang, J. & Yoo, S., 2003). This data is

$$E\,\left(\,E(\,\sigma_{ij})_{PK}{}^{Av}\right)_{PK}{}^{VD} \tag{3}$$

Since the credential shares are generated in random order by each authority A_j and the Validator receives them in the generated order, so, even with the complete set of encrypted credentials, mapping between a voter and his/her credential will not be feasible for the validator. Another thing is, since the authorities send the encrypted credential shares through mixnets, the validator will not be able to find out which authority is responsible for what credential share.

The validator calculates the respective credential for each voter by multiplying the credential shares as in (4).

$$\prod_{j=1}^{n_A} \left(E\,(\sigma_{ij})_{PK}{}^{Av}\right) = E\left(\sum_{j=1}^{n_A}(\sigma_{ij})\right)_{PK}{}^{Av} \equiv E(\,\sigma_i\,)_{PK}{}^{Av} \tag{4}$$

The validator then creates a *BST* as in Algorithm 1 with the encrypted credential as the search key for a node.

Tree Node Structure: A node in the *BST* has the following data stored: Encrypted Credential(*EC*), Time Stamp(*TS*), Encrypted Ballot(*EB*), write-in ballot bit (*WBB*) and pointers to left and right nodes as shown in Figure 1.

Additional information like time stamp, encrypted ballot and write-in ballot bit will set filled up at the time of voting.

Voting: A voter receives the credential share from the *BB* and verifies the designated proof PV_i. On successful verification, he/she multiplies together the shares $E\ (\sigma_{ij})_{PK}{}^{Av}$ as shown in (4). If the voter k selects a candidate B^k from the *BB* (approved by the Authority), then without setting the write-in ballot bit he/she encrypts the vote B^k with the public key PK^T of the tallier. Otherwise, he/she sets the write-in ballot bit and casts his/her vote in favor of his/her own candidate using PK^T . A voter V_i thus generates the vote as

$$(E(E\ (B^k)_{PK}T), TS, WBB,\quad (E\ (\sigma_i)_{PK}Av)_{PK}VD \qquad (5)$$

In (5), the first part is the ciphertext on the candidate choice of the voter, and the last part is the ciphertext on the credential of the voter. The voter wraps the resulting ciphertext with the public key PK^{VD} of the validator. The encryption using Paillier public Key (Pascal Paillier, 1999) PK^T of tallier needs to be semantically secure for preventing passive attacks. The resulting ciphertext is posted on the *BB* by the voter.

Tallying: To tally the ballots posted on *BB*, the tallier performs the following tasks:

- **Mixing the ballots:** To eliminate the correlation between the voter and his vote, a mixing of the ballots using mixnet (Lee, B., Boyd, C., Dawson, E., Kim, K., Yang J. & Yoo S., 2003) is carried out by the tallier and then is given to the validator.
- **Eliminating duplicates and checking credentials:** After the correct decryption of each vote, the validator performs a search in the *BST* for the encrypted credential of the voter. A hit in the *BST* indicates a valid credential (otherwise the vote is discarded). For a hit, depends upon the time stamp (only the last cast vote), store the encrypted ballot, the time stamp of the vote and write-in ballot bit in the corresponding nod of the tree. If it is not the last cast vote, then also discard it. Before tallying the time stamp of each ballot is removed to avoid identification of ballots. Algorithm 2 performs the elimination of duplication and validation of credentials.
- **Tallying:** The tallier multiplies all ballots in *BST* nodes whose write-in ballot bit is not set and decrypts the sum. The tallier also decrypts all encrypted ballots in the tree nodes whose bit is set since write-in ballots need to be read individually. By combining the output of the above two steps, the tallier generates the final result. Threshold decryption is applied by sharing the secret key among the Talliers. Each tallier runs the decryption algorithm and produces a partial decryption of $E\ (B^k)_{PK}T$, providing a proof of validity for the partial decryption. The tallier can then produce the decryption of ciphertext $E\ (B^k)_{PK}T$, if enough *PK* partial decryptions (t or more) are valid.

4. Performance Evaluation

4.1 Robustness/Collusion

Each authority sends the encrypted credential shares to each voter on *BB* along with a non- interactive zero knowledge proof for the authenticity of the credential. This allows the voters to identify malicious authorities. Since the credential is generated in a distributed manner, only if all the authorities have colluded, cheating the election process is possible (We have every reason to believe that there will be at least one honest authority). Since all vital data is stored on the *BB*, any malicious activity from the validator can be detected even if the validator and the tallier collude.

4.2 Universal verifiability

Since the list of cast ballots is published on the *BB*, a voter can verify whether his/her vote is taken into account or not. So the scheme is E2E verifiable. All the transactions made by the validator and all the outputs of mixing and tallying stages are stored on the *BB*. Since, the whole communication involved during the protocol execution is stored on the *BB*, the scheme remains as universally verifiable.

4.3 Correctness

Even though a voter is allowed multiple casting, only the last choice is stored in the *BST*. Duplicates are eliminated and hence one ballot is counted exactly once during tallying. Moreover, the validator cannot add or eliminate any vote since all functions are publicly verified.

4.4 Coercion Resistance and Receipt Freeness

This scheme gains coercion resistance by allowing the attacker to cast a vote by providing a fake credential with non-interactive zero knowledge proof for the credential validity. During the voting stage, a voter does not get any indication about the validity of his/her credentials, and the credentials are validated only during the tallying phase. Before the validation of credentials, a mixing of ballots is performed and the relation between a voter and the ballot is practically removed. See (Juels, Catalano & Jakobsson, 2005) for a rigorous proof.

Since this scheme supports multiple casting, the coerced voter can cast his choice during any instance of voting. However, only the last cast ballot is counted for the election result. Moreover, the coercer cannot distinguish the candidate choice of the voter, which is counted for the final tally. Also, the voter cannot be uniquely identified using the time stamp as it is removed from the ballot during the tallying stage. As the voter can furnish fake credential to the adversary, the coercer is unable to identify the candidate choice of the voter and hence the scheme is free from ”randomization attack” by Schoenmakers as given in (Juels, Catalano & Jakobsson, 2005).

4.5 Complexity Analysis

The registration phase is with a complexity of *O* (*No. of Voters*No. of Authorities*), as the validator has to multiply the credential share of each authority in order to get the credential of each voter. After issuing the credential, the validator creates the *BST* with a complexity of *O* (*No. of Voters * log* (*No. of Voters*)). As the authority generates the credential in a random order, the *BST* will be a randomly built *BST*. The height of a randomly built *BST* is *log* (*No. of voters*) (Knuth, D. E., 1973). The validation and duplication elimination of *n* votes can be done by traversing the *BST* in *O* (*n* log No. of voters*) time. Traversing through the tree can do the tallying. Hence the overall running time will *O* (*n * log*(*No. of voters*)) or *O* (*n log*(*n*)). The addition of write-in property will not cause any additional running time, as the tallying can be completed with a single traversal of the *BST*. So the reduced complexity makes this scheme applicable to any real time elections.

5. Conclusion

In this paper, we present an electronic voting scheme that achieves privacy, uncoercibility, receipt-freeness and write-in property with less running time compared to the popular coercion resistant receipt-free voting schemes. The proposed scheme allows different types of ballots that include yes/no, multi-candidate, *1* out of *t* choices, as well as write-in ballots. The protocol also allows flexible ballot formats to be used for receipt-freeness and universal verifiability. Thus, the scheme offers a simple and secure method for any generic real time electronic voting. Introduction of methods to reduce the running time further for processing *n* votes, maintaining the security is suggested as a topic for further research.

References

Aggelos Kiayias & Moti Yung. (2004). The vector-ballot e-voting approch, in: *Financial cryptography, LNCS*, vol. 3110 Springer-Verlag; pp. 72-89.

Adida, B. & Ne, C.A. (2006). Ballot casting assurance. In: EVT’06: *USENIX/Accurate Electronic Voting Technology Workshop 2006 on Electronic Voting Technology Workshop*, Berkeley, CA, USA, USENIX Association.

Adida, B. & Rivest, R.L. (2006). Scratch & vote: self-contained paper-based cryptographic voting. in: WPES *’06: Proceedings of the 5th ACM workshop on Privacy in electronic society*, New York, NY, USA, ACM Press, pp. 29–40

Alessandro Acquisti. (2004). Receipt-free homomorphic elections and write-in ballots, Cryptology ePrint Archive, Report 2004/105, <http://eprint.iacr.org/>

Ari Juels, Dario Catalano & Markus Jakobsson. (2005). Coercion resistant electronic elections, in: *WPES*, pp. 61-70.

Benaloh, J. & Tuinstra, D. (1994). Receipt-free secret-ballot elections, in: *26th ACM Sympoium on the Theory of Computing (STOC)*, ACM, pp. 544-553.

Donald, E. Knuth. (1973). Sorting and Searching, volume 3 of *The art of computer programming*, Addison-Wesley.

Krishna Sampigethaya & Radha Poovendran. (2006). A framework and taxonomy for comparison of electronic voting schemes, *Computers & Security25*, pp. 137-153.

Lee, B., Boyd, C., Dawson, E., Kim, K., Yang, J. & Yoo S. (2003). Providing receipt freeness in mixnet- based voting protocols, in: *ICISC 03*, pp. 26174.

Pascal Paillier. (1999). Public-key cryptosystems based on composite degree residuosity classes, J. Stern, editor, *EUROCRYPT '99*, Springer-Verlag, LNCS no. 1592, pp. 223-238.

Popoveniuc, S. & Hosp, B. (2006). An introduction to punchscan. in: *Work-shop on Trustworthy Elections (WOTE).*

Rivest, R. (2006). The three ballot voting system. MIT http://theory.lcs.mit.edu/~rivest/Rivest-TheThreeBallotVotingSystem.pdf.

Stefan Brands. (2002). A technical overview of digital credentials, Technical Report.

Algorithm 1. Credential Insertion to BST

```
for i = 1 to $n_V$
    root ← insert(root, $E(\sigma_i)_{PK^{Av}}$)
```

Algorithm 2. Validation and Elimination of Duplicate Credentials

```
for i = 1 to n                    //n is the total number of vote cast
    node = search ( root , Ballot[EC]   )
    if node ≠ 0 then
        if node[TS]< Ballot[TS] then
            node [TS, EB, WBB] ≡ Ballot [[TS, EB, WBB]
        else discard (Ballot)
    else discard (Ballot)
```

Encrypted credential	Time stamp	Encrypted Ballot	Write-in ballot bit	Left Node	Right Node

Figure 1. BST node structure