福井大学審査
学位論文【博士（工学）】

A Dissertation Submitted to the University of Fukui

for Degree of Doctor of Engineering

A Scheme for Electronic Voting Systems
【電子投票システムの研究】

カ ジ ムハマド ロ キ ブ ル ア ラ ム
Kazi Md. Rokibul Alam

2010年 9月

福井大学審査

学位論文【博士（工学）】

A Dissertation Entitled

A Scheme for Electronic Voting Systems

【電子投票システムの研究】

Submitted for the

Program of System Design Engineering,

Graduate School of Engineering, University of Fukui

for the Degree of Doctor of Engineering

Submitted by

カジ ムハマド ロキブル アラム
Kazi Md. Rokibul Alam

Supervised by

教授 シンスケ タムラ
Prof. Shinsuke Tamura

Acknowledgement

At the outset, I express my heart-felt gratitude to my supervisor Prof. Shinsuke Tamura, Graduate School of Engineering, University of Fukui, for his support, guidance, encouragement and continuous assistance. His inspiration and infinite dedication has enabled me to complete the thesis successfully.

I do acknowledge to Dr. Tatsuro Yanase and Dr. Shuji Taniguchi for their cooperations during my study in this Lab. I am also thankful to all undergraduate and postgraduate students of the laboratory and especially to my tutors Mr. Yusuke Ohashi and Mr. Hiroya Tsurugi for their social, academic and administrative support and friendship. I would also like to thank to all of the staff at University of Fukui.

Finally, I would like to thank my family members specially my wife and my mother for all their support and encouragement throughout my doctoral program.

Affectionately dedicated

To

My Father "Kazi Abdur Rashid"

who was very sick when I was coming for my doctoral program and passed away on 17. 10. 2007 (just 12 days after my coming to Japan)

Abstract

This thesis proposes a new electronic voting (e-voting) scheme that fulfills all the security requirements of e-voting *i.e*. privacy, accuracy, universal verifiability, fairness, receipt-freeness, incoercibility, dispute-freeness, robustness, practicality and scalability; usually some of which are found to be traded. When compared with other existing schemes, this scheme requires much more simple computations and weaker assumptions about trustworthiness of individual election authorities. The key mechanism is the one that uses confirmation numbers involved in individual votes to make votes verifiable while disabling all entities including voters themselves to know the linkages between voters and their votes. Many current e-voting schemes extensively deploy zero-knowledge proof (ZKP) to achieve verifiability. However, ZKP is expensive and complicated. The confirmation numbers attain the verifiability requirement in a much more simple and intuitive way, then the scheme becomes scalable and practical.

Contents

List of Figures

List of Tables

List of Symbols

BB	Bulletin board
CN	Confirmation number
CRV	Coercion-Resistant Voting
C_{Cj}	Confirmation number assigned to voter V_j
DM	Disruption detection manager
DRE	direct/digital-recording electronic
E-voting	Electronic voting
IZKP	Interactive zero knowledge proof
ID_j	Identifier
NIZKP	Non interactive zero knowledge proof
P/W_j	Password
S&V	Scratch & Vote
TM_i	Tallying manager
TRR	Tamper-resistant randomizer
T_{tj}	Token assigned to V_j
V_j	Voter
vj	vote of Vj
VM	Voting manager
x_{xj}	unknown random number assigned to V_j
ZKP	Zero knowledge proof

Chapter 1

Introduction

1.1 Motivation

Voting is the process, in which voters cast their votes while a group of authorities collects the votes and outputs the final tally. Conventional voting systems include papers, punch cards, mechanical levers, optical-scan machines etc. To decide successful candidates a paper ballot voting system records and counts votes of voters cast on paper sheets where paper sheets are produced by voters themselves, by political parties or by election authorities. In a punch card voting system, cards and a small clipboard-sized device are provided to record votes. Voters punch holes in cards at positions corresponding to their selected candidate using punch devices, and then the cards are placed in a ballot box for tabulation. In a mechanical lever voting system, the name of each candidate is assigned to a particular lever in a rectangular array of levers on the front of the machine. A set of printed strips visible to the voters indicate the lever assignment for each candidate and issue the choice. In an optical-scan machine voting system, optical scanners are used to read marked paper ballots and tally the results. Here voters mark their choices in locations corresponding to their choices usually by filling rectangles, circles, ovals, or by completing arrows.

However, none of these conventional schemes can satisfy a truly secure and verifiable election while maintaining privacies of voters because they cannot prove their honest operations without revealing individual votes. Also, these systems are not efficient as they are conducted manually and therefore very often they are not accurate. As a consequence, extensive research is going on in the field of e-voting for last several decades for the purpose of substitution of these systems to establish the true democracy in societies.

Unlike these systems, electronic voting (e-voting) systems based on computers, computer networks and cryptographic protocols, alleviate the limitations of conventional voting systems, and they enable efficient, accurate, verifiable and convenient elections. Also

the resources of e-voting schemes (*e.g*. the computing devices, the software and the communication mechanisms) are reusable, therefore e-voting based elections become inexpensive and economic. Moreover, they do not require any geographical proximity of voters (*e.g*. soldiers or employees working abroad can participate in elections) and they provide better scalability for large scale public elections [1]. The number of people those who usually do not participate in elections because of the inconveniences of conventional voting systems may be encouraged by the above conveniences of e-voting systems and thereby the number of vote castings can be maximized in elections.

However e-voting schemes have potential problems that may degrade their acceptances. For examples, simple issuing of a unique identification number to each voter to smoothly verify the accuracy of elections would enable the authority (or authorities) to identify the linkages between voters and their votes and disclose the privacy of the voters [3]. When election authority issues receipts to voters to prove its honesty, coercers can force voters to follow their intentions more easily. On the other hand, complicated mechanisms that achieve complete anonymity of voters while maintaining verifiability of their votes make e-voting systems non-scalable and non-practical. For example, many election schemes involve zero knowledge proof (ZKP) (either interactive or non-interactive) to prove the correct behavior of entities *e.g*. to confirm that only eligible votes are accepted and all eligible votes are counted, however ZKP requires complicated computations and communications which make e-voting schemes unrealistic [17]. Also in many existing schemes, trustworthiness of authorities is assumed to conduct the election *e.g*. to generate and distribute tokens while registering the legitimate voters for the election, which lead to sacrifice privacy of voters and incoercibility. Likewise the assumption of the existence of trusted or absolutely trusted authority (or authorities) is not practical. Moreover, the vote formats of many existing e-voting schemes are not flexible, *e.g*. some of them can support only yes/no votes or simple one out of two candidate elections while some other schemes can support only pre-specified candidates elections.

To make e-voting schemes acceptable they must satisfy extensive requirements related to privacy, verifiability, implementation, flexibility of vote formats and the assumptions about trustworthiness of involved authorities. E-voting schemes must satisfy even mutually contradictory requirements, and satisfying all of them altogether at the same time is highly challenging.

The objective of this research is to establish an e-voting scheme that fulfills all requirements for e-voting. The proposed e-voting scheme [17] in this research is characterized as follows, namely it

1) satisfies all the security requirements of e-voting systems *i.e*. privacy, accuracy, universal verifiability, fairness, receipt-freeness, incoercibility, dispute-freeness, robustness, practicality and scalability [1, 8, 15]; which are usually found as traded in existing e-voting schemes,
2) the scheme is based on the weaker assumptions about trustworthiness of entities, *i.e*. no one can make the scheme unreliable if at least one authority is honest among multiple authorities, and
3) it enables flexible candidate selection *i.e*. it accepts freely chosen write-in ballots, votes for pre-specified or *t* out of *l* choices as well as yes/no votes.

1.2 Overview of the field

Based on adopted cryptographic techniques, existing e-voting schemes can be classified into three categories: blind signature based schemes [2, 3, 4], homomorphic encryption based schemes [5, 6, 7] and mixnet based schemes [8, 9, 10]. A lot of hybrid of homomorphic encryption and mixnet based schemes [11, 12, 13] are also available. Besides these schemes, paper based cryptographic voting schemes [14, 15, 16] that rely on visual cryptography have been proposed. However, existing schemes are unable to satisfy all the essential requirements of e-voting systems at the same time because there are tradeoffs among the individual requirements and constraints are remarkable. Also to achieve the verifiability of votes or to prove the honest behaviors of voting authorities, almost all of these schemes extensively deploy ZKP, which is expensive, not efficient and not practical enough, because it requires complicated computations and communications. For example, homomorphic encryption based schemes use ZKP to prove the validity of votes and their correct decryptions, and mixnet based voting schemes use ZKP to prove the correctness of operation of each mix-server. Therefore currently available e-voting systems can satisfy only a part of the requirements of voting and also they are non-scalable and non-practical.

1.3 Overview of the proposed e-voting scheme

Key mechanisms of the e-voting scheme proposed in this thesis are confirmation numbers (*CNs*), signature pairs on encrypted votes and those on blinded tokens. Here *CNs*

are publicly disclosed and registered unique numbers and they are attached to votes of individual voters, and a pair of signatures on encrypted votes ensure the authenticity of these encrypted votes. The other component signature pairs on blinded tokens enable voters to act anonymously.

CNs involved in individual votes make votes verifiable while disabling all entities including voters themselves to know the linkages between voters and their votes. *CNs* are unique registered numbers and they are encrypted by multiple entities independently, so that no one knows their exact values. Therefore anyone can convince itself the authenticity of votes when attached *CNs* are the registered ones. Nevertheless any link between voters and their votes is removed because no one knows the decrypted forms of *CNs* attached to voters. Also publicly disclosed encrypted *CNs* ensure that all votes from eligible voters are counted, and thereby maintain the total accuracy of the election while protecting all privacies of voters. Different from ZKP, a mechanism for *CNs* is simple enough, it requires much less computations for individual entities without assuming any absolutely trustworthy election authority. Because of *CNs* this scheme requires much more simple computations for election entities in comparison with other existing schemes. The proposed scheme does not need any extra proof of correctness of votes. Therefore it is possible to develop e-voting systems that satisfy all the requirements including scalability and practicality.

A signature pair of multiple managers on encrypted vote proves the authenticity of vote even when the decryption of encrypted vote reveals a disrupted result. Namely, anyone can convince itself that the vote is meaningless from the beginning when two different signatures on the vote reveal the same value, because no one can forge two different signatures consistently without conspiring with all managers.

Signature pairs on blinded tokens enable voters to act without disclosing their identities *i.e*. anonymously. Although the signatures on token assigned to voter prove its eligibility, token does not reveal voter because managers sign on it blindly. The first signature of the pair is used for vote casting and the second one is used for approving the vote registration. Because the two signatures are generated by different signing keys, voter can prove its eligibility by the second one even after the first one had been publicly disclosed.

1.4 Organization of the thesis

In Chap. 2 the requirements of e-voting schemes are introduced with the related works. Then, the security components are discussed in Chap. 3. The configuration of the proposed e-voting scheme that consists of voters, a single Voting manager, multiple mutually independent Tallying managers and Disruption detection manager is discussed in Chap 4. Chapter 5 provides the precise descriptions of the individual stages of the scheme, and Chapter 6 evaluates the proposed scheme. Namely, behaviors of the scheme are analyzed against various kind of security threats and the computation volumes required for carrying out the scheme are evaluated. The proposed scheme showed substantially better performance than existing schemes that rely on ZKP; which proves that the proposed scheme is scalable and practical enough. Finally Chap. 7 summarizes the work.

Chapter 2

Requirements and Related Works

This chapter discusses the requirements of e-voting systems and the related works while summarizing the contributions of the proposed scheme.

2.1 Requirements of e-voting schemes

E-voting schemes need to satisfy extensive requirements, some requirements are conflicting with others and there are tradeoffs among them. Because of these features of requirements, voting is one of the most challenging applications of information security technologies. Ideal e-voting schemes should satisfy the following requirements [1, 8, 15].

✧ **Eligibility:** As the most primitive requirement for conducting reliable elections, only persons who meet certain pre-determined criteria *e.g.* who have citizenships are allowed to cast permitted number of votes. To achieve this, authority needs to verify the eligibility of voters and record their casting votes.

✧ **Privacy:** Usually voters do not want others including election authorities to know their casting votes. Therefore, anyone must not be able to know votes except its own vote. To achieve this, any traceability between voters and their votes must be removed during the whole election, *i.e.* it is necessary to conceal the identity of voters or votes at every stage of the election.

✧ **Accuracy:** In elections, voters expect that their votes are correctly captured and that all eligible votes are correctly tallied. Accuracy is the degree of satisfactions of voters' this expectation, and can be maintained by the verifiability mentioned below.

✧ **Verifiability:** Verifiability is the ability to determine whether only and all valid votes are counted in final tally or not *i.e.* to determine the accuracy of the election.

Accuracy of the election can be verified in two ways, one is the *individual verifiability* where only voters can verify their own votes in the tally. Therefore accuracy of the election consists of *n* voters is ensured when there are less than or equal to *n* votes and all *n* voters verify their votes. The other is *universal verifiability* which enables any third party to verify the accuracy of the election.

✧ **Fairness:** In order to conduct the impartial election, anyone is not allowed to compute the partial tally before the end of the election which may influence the remaining voters and may affect the voting result. Some voting schemes trust that the authorities will not reveal partial tally *e.g.* [7, 8], but practical solutions must exclude this kind of assumptions.

✧ **Receipt-freeness:** Receipt-freeness disables anyone including voters themselves to link voters to their votes, in order to protect voters from being coerced to follow intentions of other entities. To achieve receipt-freeness, the voting system should not leave any information about votes to voters. Also, votes should not include any information peculiar to the voters. If a vote includes some traceable information regarding the corresponding voter, this information can work as the receipt. When the receipt-freeness is not ensured, e-voting systems enable entities to easily gather data about voters and their votes and link them each other, therefore e-voting schemes cannot be used for real political elections without satisfying receipt-freeness. In some voting schemes, authorities assign random numbers to voters to be put in their votes *e.g.* [5, 6, 7] and cannot achieve receipt-freeness completely because authorities can easily link voters to their votes based on these random numbers. Receipt-freeness shares the same notion with privacy.

✧ **Incoercibility:** Incoercibility protects voters against coercers who can communicate with the voters actively. Incoercibility must cope with *randomization*, *forced-abstention and simulation attacks*.

- *Randomization* attacks force voters to submit invalid votes by manipulating the manner in which votes are cast.
- *Forced-abstention* attacks enable coercers to force voters to abstain from casting their votes, and
- *Simulation* attacks let coercers impersonate valid voters at some stage of the voting scheme and submit votes on their behalf.

Receipt-freeness property does not imply incoercibility but incoercible schemes must be receipt-free.

✧ **Dispute-freeness:** To conduct elections in environments where even dishonest voters are involved, disputes between entities should be solved without involving irrelevant entities. The notion of universal verifiability is similar to dispute-freeness but it is limited to the voting and tallying stages.

✧ **Robustness:** Any entity should not be able to disrupt the voting, *i.e*. the voting system must be able to detect dishonest entities and to complete the voting process without the help of detected dishonest entities.

✧ **Scalability:** In order to enable large scale elections, a scheme has to be extended easily while satisfying computation, communication, and storage requirements of the scheme.

✧ **Practicality**: A scheme should not have assumptions and requirements that are difficult to implement.

Among these requirements, some are usually satisfied and their implementation is not hard, but some others are difficult to satisfy. Especially satisfying several hard requirements altogether at the same time is really difficult because there are tradeoffs among them. For example, achieving incoercibility leads to sacrificing universal verifiability and hence accuracy because incoercible schemes conceals the links between voters and their votes while vote submission. As another example, satisfying dispute-freeness makes schemes complicated [1] because for every stage of the election, dispute-free schemes need to prove the validity of all actions of all involved entities and consequently schemes become impractical or unscalable. Also write-in ballots clash with the properties of receipt-freeness of universally verifiable schemes and randomization attacks (already discussed, which means to force a voter to vote in a certain way). Here write-in ballots are ballots in which a voter can insert a freely chosen message - a right protected in certain legislations and jurisdictions. [13]. Herein, peculiar information inserted within write-in ballots can be used as receipts of their corresponding voters, and thereby coercers can mount randomization attack by manipulating voters to submit invalid votes.

On the other hand sacrificing one requirement sometimes also leads to sacrificing another one or more requirements because they are mutually dependent and interrelated. For example, the maximal level of privacy preservation and fairness has the same notion against corrupt authorities. Because, maximal privacy offers the privacy of a voter to be breached only with a collusion of all remaining entities *e.g*. voters and authorities, and while desirable, requires all the voters to either participate in the post-vote-casting stage or to mandatorily cast their votes (*i.e*. no abstaining). In this situation, breaching the privacy of voters enables corrupted authorities to modify or reveal the partial tally.

Because of these, many existing e-voting schemes can satisfy only a part of the above requirements. For example, voting scheme proposed in [19] can satisfy privacy, accuracy, fairness, universal verifiability, dispute-freeness and practicality, but it cannot satisfy either of robustness, receipt-freeness, incoercibility or scalability. But e-voting systems must cope with intrinsic tradeoffs among these requirements.

2.2 Related works

E-voting schemes proposed and developed up to now, can be classified into three major categories: Schemes in the first category are cryptographic voting schemes *e.g*. [2, 3, 6, 7, 8, 11, 12, 13] and they are based on cryptographic algorithms without any specific hardware devices. Schemes in the second category are based on visual cryptographic algorithms and papers, and they are called paper based cryptographic voting schemes *e.g*. [14, 15, 16]. The third category is the commercial e-voting scheme *e.g*. [28, 29, 30] and schemes in this category are based on cryptographic techniques and machines like optical scan voting machine, direct/digital-recording electronic (DRE) etc.

This thesis discusses a scheme in the first category, and as already discussed in Chap 1, there are three approaches to developing cryptographic voting schemes, they are (i) *blind signature*, (ii) *mixnet* and (iii) *homomorphic encryption* based schemes.

According to how voters submit their votes to the tallying authority (or authorities), [1] has classified e-voting schemes as: *hidden voter*, in which voters anonymously submit

their votes; *hidden vote*, in which voters openly submit their encrypted votes; and *hidden voter with hidden vote*, in which voters anonymously submit their encrypted votes.

2.2.1 Blind signature based schemes

Blind signature is a digital signature scheme that allows an entity to get the signature on a message without revealing the content of the message. Therefore it can be used to authenticate a vote without knowing the content of it. When combined with anonymous channel, blind signature can achieve maximal privacy property [1].

Herein, voters encrypt their votes before presenting them to the election authority for validation, and after the authority validates their votes, voters decrypt the encrypted signed votes in order to reveal signed votes [13]. The protocol proceeds as follows:

Step 1: Voter V_j blinds its vote v_j to $E(a_j, v_j)$ by using its secret encryption key a_j and sends it to the authority (or authorities) *TM*.

Step 2: *TM* verifies the eligibility of V_j and then signs on $E(a_j, v_j)$ *i.e.* generates $S(X_i, E(a_j, v_j))$ by using its signing key X_i and sends it to V_j.

Step 3: Finally, V_j unblinds $S(X_i, E(a_j, v_j))$ and verifies the signature of *TM i.e.* generates $S(X_i, v_j)$, which is the signature on vote v_j.

Schemes based on blind signature are simple, efficient, and flexible. They enable voters to cast any form of votes including freely chosen write-in ballots. Usually they possess the fairness property because votes are blinded and authorities are unable to compute the partial tally. The limitations of these schemes are: usually they cannot satisfy receipt-freeness, because voter's blind factor can be used as a receipt of its vote; and the voter can prove its vote to buyers [8]. Also, they cannot satisfy universal verifiability because votes are encrypted by the corresponding voters and votes can be verified only by their voters. Moreover the involvement of voters in the post-voting *i.e.* tallying stage sacrifices practicality. Besides, these schemes assume the existence of anonymous channels between voters and authorities which is impractical because usually an anonymous channel is implemented using mixnet which is inefficient. From the beginning, if a secure mixnet is available, a blind signature is not required anymore [8].

2.2.2 Mixnet based schemes

Mixnet enables a set of senders to send their messages while concealing their identity *i.e.* anonymously, thus it is a primitive component to provide entities with services while not knowing their identities. It consists of multiple mix-servers and takes a set of encrypted messages as its input and produces a new form of representation of the same messages through either decryption or by encryption operations and indistinguishable shuffling. At stage i, a batch of inputs are received by mix-server M_i and M_i transforms inputs by using either decryption key or by re-encryption, shuffles and transfers to mix-server M_{i+1} to proceed to stage $i + 1$. The original proposal of the mixnet was a decryption mixnet, but many recent works deal with re-encryption mixnet, since it can separate mixing and decryption phases, which provides more flexibility, robustness, and efficiency. Here it is noticed that RSA [24] based mixnet requires the voter to perform n encryptions where n is the number of mix-servers.

Mixnets can also be classified into verifiable mixnet and optimistic mixnet depending on mechanisms to prove the correctness of their behaviors. In verifiable mixnet while votes are disclosed, each mix-server provides proofs of its correct shuffling and thus the correctness of mixing is publicly verifiable. On the other hand, in optimistic mixnet the verification of correct shuffling is not provided by each mix-server. Instead, the correctness of the shuffling of the whole mixnet is verified after the mixnet outputs the shuffling results in plaintexts. Drawbacks of optimistic mixnets include that a cheating server cannot be identified instantly and some outputs are revealed in plaintexts even when the shuffling is incorrect [8].

Schemes based on mixnet are flexible *i.e.* there is no stringent limitations on vote formats, it can support any vote format either pre-specified or unspecified *i.e.* write in ballots. However, schemes based on mixnet are complicated and generally not efficient in practical implementations because it requires a heavy processing load during the tallying process which makes it slow, and it also requires a huge amount of computations for proving the correctness of shuffling and re-encryptions or decryptions *i.e.* the correctness of behaviors of mix-servers. Without the proof of correctness, schemes based on mixnet cannot conduct an accurate election and cannot provide the privacy of voters.

2.2.3 Homomorphic encryption based schemes

An encryption function $E(K, x)$ is said to be homomorphic, if encrypted forms of m_1 and m_2, *i.e.* $E(K, m_1)$ and $E(K, m_2)$ satisfy the relation $E(K, m_1 \circledcirc m_2) = E(K, m_1) \circledcirc E(K, m_2)$ for some operation $\circledcirc$. The operation $\circledcirc$ can be a modular addition ($\oplus$, additive homomorphism) or multiplication ($\otimes$, multiplicative homomorphism). Homomorphic voting schemes apply certain properties of probabilistic cryptosystems where correspondence between a plaintext and a ciphertext exists between a certain group in the plaintext space and the group in the ciphertext space [13]. RSA [24], ElGamal [25] and Paillier [26] etc. well-known asymmetric or public key cryptosystems possesses multiplicative homomorphism. Homomorphic encryption provides a mechanism to directly combine the encrypted votes to get an encrypted tally. The mechanism of homomorphic encryption based voting scheme is as follows:

Voter V_j posts a vote while encrypting it to hide the linkage between the voter and its vote (privacy). Then the tally is obtained by decrypting the sum or the products of them. However, validity of the encrypted votes has to be ensured before combining them. The voter is therefore required to provide an interactive or non-interactive ZKP of validity of its vote. The general form of the vote that V_j posts is: $\{E(K, (v_j \parallel r_j)), proof_j\}$ where K is the public key of a probabilistic homomorphic encryption scheme, v_j is the vote, $proof_j$ is a proof of validity of the vote. After verifying the proofs of votes, the tallying authority computes: $\prod_j E(K, v_j) = E(K, \{\sum_j v_j, \prod_j r_j\})$ or $E(K, \{\sum_j v_j, \sum_j r_j\})$ due to the homomorphism of encryption E. The authority needs to post decrypted tally $\sum_j v_j$ and a proof of correct decryption. Using the posted quantities on the public broadcast channel, anyone can compute and verify tally to be valid, thus achieving universal verifiability.

Importantly, it is very easy for homomorphic voting schemes to satisfy universal verifiability as well as accuracy property. Another advantage is that there is no requirement for any form of mixnet. However the encoding of vote is limited *i.e.* not flexible [1], it cannot support write-in ballots, can support simple binary choices *i.e.* yes/no votes and votes for pre-specified candidates, and the use of intensive ZKP to prove the validity of ballot in the voting stage is costly for the voter [7, 8]. Also, because of the involvement of huge ZKPs, schemes sacrifices scalability and practality.

2.3 Receipt-free schemes

Among various security requirements, many mixnet and homomorphic encryption based schemes emphasize on receipt-freeness and incoercibility, because although they are difficult to satisfy, they are essential for voting. Receipt-freeness disables voters to prove their votes to any entity in order to achieve incoercibility. To achieve receipt-freeness, many voting schemes [5, 6, 7] attach secret random numbers to votes while proving the correctness of votes by using interactive-ZKP (IZKP) or non-interactive-ZKP (NIZKP). In order to generate secret random numbers, these schemes assume some kind of trusted authorities and rely on some physical assumptions about the communication channels between the voter and the authorities *e.g*. one-way untappable channels from voters to the authorities, one-way untappable channels from the authorities to voters and two-way untappable channels (voting booth) between voters and the authorities.

Voting scheme proposed in [5] implemented receipt-freeness in homomorphic encryption based voting scheme. But because of IZKP involved in individual voter's vote while a voter casts its vote, every voter has to wait for all other voters to finish their IZKP phases which make the scheme unscalable. Moreover it was demonstrated that this voting scheme could not satisfy receipt-freeness. A voting scheme based on homomorphic encryption and multiple authorities achieves receipt-freeness [6] while assuming the existence of an untappable channel from each authority to each voter so that authorities can jointly generate random numbers for voters to construct their ballots. Voting scheme proposed in [8] presented another receipt-free e-voting scheme based on re-encryption mixnet protocols with a tamper-resistant randomizer (TRR), a hardware device generates the random numbers for voters to be used to construct their ballots.

All of these schemes exploit ZKP to attain verifiability. However ZKP that requires non negligable computations makes the schemes impractical. Also untappable channels used in [6] make the scheme unrealistic, *i.e*. the scheme sacrifices practicality and scalability. A worse thing is that the scheme cannot achieve the complete receipt-freeness. Namely, authorities can know the random numbers and can know links between voters and their votes. Although TRR, a secure hardwire device such as smart card or Java card to generate random numbers for voters used in [8], achieves the complete anonymity of voters, TRR further worsens its practicality because TRR is not applicable to general re-encryption mixnets where voter needs to provide the proof of knowledge of its secret random number

used to construct its vote.

2.4 Incoercible Schemes

Several incoercible e-voting schemes also had been proposed [11, 12, 13]. In these schemes, voters obtain unique tokens provided by trusted authorities and construct their encrypted votes while combining with the encrypted tokens, to submit multiple votes without being traced by others. Election result is computed while comparing a list of encrypted tokens (prepared by the authorities) with a list of encrypted votes. As a consequence, coercers cannot identify exact votes of voters. However, ZKP to confirm the equivalence of tokens corresponded to multiple votes of same voters, sacrifices practicality and scalability.

Schemes proposed in [12, 13] allow a voter to cast multiple votes with the same token and authority consider only one encrypted vote per token for decryption. Scheme proposed in [12] includes two NIZKP processes; one for the token verifications and another for the vote verifications. First it verifies the correctness of token; therefore a voter can submit the same valid token with its vote multiple times. Voting scheme proposed in [13] improves [12] by accommodating write-in votes and by simplifying the computational burden necessary for voters and in [13] pre-determined policy *e.g.* timestamps removes the duplicate tokens of the same voter. It allows voters to combine their votes with their tokens by applying homomorphic encryption property. Authorities post the token shares of voters on *BB* needed for tallying and also send the same tokens to the registered voters with a designated verifier ZKP to prove the equivalence of these tokens. Like token, authorities also create the shares of permissible ballots which voters can cast in the election. These shares are encrypted with two different public parameters and are posted on *BB* together with NIZKP to prove that each pair of ciphertexts are encryptions of the same underlying share of ballot. Although both [12] and [13] schemes have achieved incoercibility; unfortunately scalability, universal verifiability and accuracy properties are traded for it *i.e.* for ZKP. Also the anonymous broadcast channel with no designated section on the *BB* in [12, 13] is also difficult to implement [1] and it sacrifices universal verifiability property.

A scheme proposed in [11] employs an observer that serves as a convenient and secure transport to facilitate the registration and voting for the benefit of the voter. It also

simplifies the time consuming verification processes at the tallying stage but still involves ZKP to disable voters to transfer encrypted forms of their tokens to others, and NIZKP to prove the correctness of encryption of votes.

2.5 Voting schemes in other categories

2.5.1 Paper based schemes

Regarding paper based cryptographic voting schemes, visual cryptography based schemes had been proposed [14, 15, 16]. Voting scheme proposed in [14] achieved receipt-freeness innovatively and used robust decryption mixnet. Here, voter first fills out its ballot, physically splits it into two pre-determined halves, destroys one, and casts the other while taking a copy of this same half to home with itself as a receipt. But because of mixnet involved in it, inaccuracies can be produced which may lead to an unfair re-election [1] and election officials with the proper secret keys can recover voter's choice during the tallying process. As paper ballot, voter needs to verify its ballot prior voting to ensure that the two halves of the ballot are consistent with one another. Without this verification, a fraudulently created ballot could corrupt the proper recording of the voter's intent. In voting scheme proposed in [15] the ballots are self-contained *i.e*. any one including voter itself, can audit the ballot without interacting with election officials before voter casting its ballot. Also NIZKP generated by election officials prove the correctness of proper ballots.

However, in these systems, voters must delegate their vote computations to the voting booth, therefore the voting booth can know the votes of voters, by which the privacy of voters may be breached. Also paper ballots prepared in advance by either single or multiple authorities do not guarantee privacy against the ballot creators. Although a solution exists for the privacy problems with respect to the voting booth [16], it involves NIZKP to prove the correctness of votes. In it, voter uses computer device only at the preprocessing stage but voting itself is done bare-handedly like [14, 15] etc. Here for every candidate voter itself encrypts two ballots along with a NIZKP from which one ballot is selected and published on *BB*. Now voter casts its vote for a candidate and sent it to the booth which is not published on *BB*. Voting booth re-encrypts the remaining ballot twice and publish on *BB*. The scheme achieves incoercibility, unforgeability, true privacy with bounded candidates and vote is not revealed to even to the booth. But it assumes the existence of recordable

private channel which is an impractical assumption to implement, and it cannot solve disputes between voters and the booth.

2.5.2 Commercial schemes

Commercial e-voting systems have produced many high-profile software and hardware *e.g*. optical scan voting machine, direct/digital-recording electronic (DRE) voting machine etc. However herein, the operational and procedural errors that can occur during elections is quite large. In practice, it has been observed that these hardware machines produce anomalies like under-votes, ambiguous audit, choices "flipping" before the voter's eyes etc. Also, it has been reported that they have their deficiencies in design and implementation, therefore not secure. Security flaws, software bugs, operational errors and mistakes, incorrect configuration, mechanical failure of these hardware and poor human factors of the ballot design etc. make these voting systems inoperable to conduct real world elections. Sometimes, the total arrangement of these hardware for voting, increases mechanical complexity, maintenance burden and failure rates of these machines. Although, NIZKP involved in these systems can prevent a voting machine from grossly stuffing ballots, they cannot prevent a voting machine from flipping votes from one candidate to another [28].

However, some research is going on to develop user interface of these hardware from pre-rendered graphics, reducing runtime code size as well as allowing the voter's exact voting experience to be examined well before the election.

2.6 Contributions of this scheme

To enable e-voting to satisfy all requirements, the proposed scheme uses *CN*s. Namely, *CN*s that are unique and registered in the system are attached to individual votes while being encrypted so that no one knows their exact values. Therefore votes can be verified by checking the attached *CN*s, nevertheless any link between voters and their votes are removed. *CN*s also ensure that all votes from eligible voters are counted, and thereby maintain the total accuracy of the election.

Different from ZKP, a mechanism for *CN*s is simple enough, it requires much less computations for individual entities without assuming any absolutely trustworthy entity.

Therefore it is possible to develop e-voting systems that satisfy all the requirements including scalability and practicality.

Although several voting schemes [5, 6, 7] had already used unique numbers as tokens to make votes verifiable, but they only prove the correctness of individual votes, do not ensure that all votes from eligible voters are counted. Moreover, these schemes require trusted entities that know the values of tokens; therefore complete privacy or incoercibility is not achieved.

Chapter 3

Security Components

This chapter describes the key security components used in the proposed voting scheme. They are bulletin board (*BB*), blind signature, signature pairs on blinded tokens, mixnet like encryption and decryption shuffles, confirmation numbers (*CNs*), signature pairs on encrypted votes, a probabilistic and commutative re-encryption mechanism and an anonymous authentication mechanism. Here except the *BB*, mixnet, blind signature and anonymous authentication mechanism, other components are newly developed for the proposed voting scheme to satisfy all the requirements of e-voting.

3.1 Newly developed components

3.1.1 Confirmation numbers (*CN*s)

Confirmation numbers (*CNs*) are unique and registered numbers and they are assigned to individual voters to make votes verifiable. Because all *CNs* are publicly disclosed, anyone can convince itself that votes attached by *CNs* are the ones submitted in the authorized way. Also by examining the used *CNs*, anyone can confirm that all submitted votes are counted. On the other hand, *CNs* are attached to votes while being encrypted so that no one can know their decrypted forms. Therefore, anyone including voters themselves cannot link voters to their votes attached by *CNs*.

To conceal the content of C_{Cj} (confirmation number assigned to voter V_j) from any entity including V_j itself, firstly a single entity generates N different encrypted *CNs* for N voters in advance as shown in Fig. 3.1 (a). Then P (at least two) mutually independent election authorities repeatedly perform encryptions and shuffles of all *CNs* by using their encryption keys, *i.e.* firstly the first election authority TM_1 encrypts C_{Cj} to C_{Cj}' to be placed in random positions as shown in Fig. 3.1 (b). Then other mutually independent election

authorities *i.e.* TM_2, TM_3, --- execute the same operations repeatedly, *i.e.* C_{Cj}' is further converted to C_{Cj}'', C_{Cj}''',--- as shown in Fig. 3.1 (c) and (d). Therefore, no entity can identify the linkage between original C_{Cj} and its encrypted form unless multiple election authorities conspire *i.e.* no one including V_j itself can identify V_j from C_{Cj}.

Here $C_{Cj}' = E(K_1, C_{Cj})$, $C_{Cj}'' = E(K_2, C_{Cj}')$, $C_{Cj}''' = E(K_3, C_{Cj}'')$,---, provided that x is encrypted to $E(K_i, x)$ by the encryption key K_i of the i'th election authority TM_i. In the following repeatedly encrypted C_{Cj} is denoted as $E(K_*, C_{Cj})$, *i.e.* $E(K_*, C_{Cj}) = E(K_P, E(K_{P-1}, \text{--- } E(K_1, C_{Cj}) \text{ --- }))$. This multiple encryption is carried out based on the probabilistic and commutative re-encryption scheme described in Sec 3.1.3.

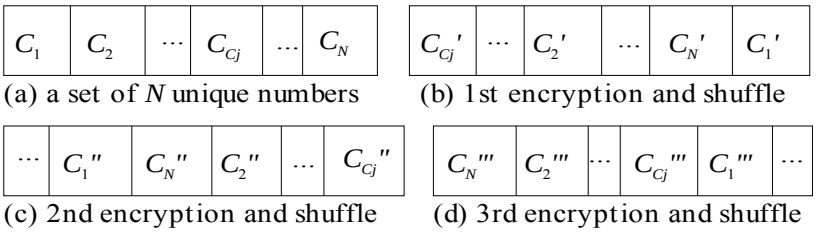


Fig. 3.1 Encryption steps of confirmation numbers

3.1.2 Signature pairs on encrypted votes

Signature pairs on encrypted votes prove the authenticity of votes and the honesty of all election managers, *i.e.* when managers are honest they can disable anyone to blame them for vote disruptions. In the proposed scheme v^*, repeatedly encrypted form of vote v, put and verified by the corresponding voter V_j is decrypted to v by multiple mutually independent election managers TM_1, ---, TM_P, and to protect v^* from unauthorized modifications, multiple election managers repeatedly sign on v^* by their signing keys $\{M_1, M_2, \text{---}, M_P\}$ to generate $S(M_*, v^*) = S(M_P, S(M_{P-1}, \text{---}, S(M_1, v^*) \text{ --- }))$ before the decryptions, where $S(M_i, x)$ is the signature of TM_i on x generated by its signing key M_i. Namely, when encrypted signed form $S(M_*, v^*)$ is successfully decrypted to signed form $S(M_*, v)$, anyone can convince itself that multiple election managers had honestly decrypted $S(M_*, v^*)$.

However, this scheme is effective only when all voters put meaningful votes. When decryption result is meaningless, entities cannot determine whether multiple election managers are dishonest or v is meaningless from the beginning. A signature pair on v^* solves this problem. When each election manager signs on v^* by its two different signing keys $M_{(1)i}$

and $M_{(2)i}$ as shown in Fig. 3.2, it is impossible to consistently generate two different signed forms $S(M_{(1)*}, v) = S(M_{(1)P}, S(M_{(1)P\text{-}1}, \text{---}, S(M_{(1)1}, v) \text{ --- }))$ and $S(M_{(2)*}, v) = S(M_{(2)P}, S(M_{(2)P\text{-}1}, \text{---}, S(M_{(2)1}, v) \text{ --- }))$ in unauthorized ways because no one knows all signing keys. Namely, anyone can convince itself that multiple election managers had decrypted $S(M_{(1)*}, v^*)$ to $S(M_{(1)*}, v)$ honestly, when two forms $S(M_{(1)*}, v^*)$ and $S(M_{(2)*}, v^*)$ reveal the signatures on the same v. These signatures are also generated based on the probabilistic and commutative re-encryption scheme.

$$S(M_{(1)*}, v) = S(M_{(1)P}, S(M_{(1)P\text{-}1}, \text{---}, S(M_{(1)1}, v) \text{ --- }))$$
$$S(M_{(2)*}, v) = S(M_{(2)P}, S(M_{(2)P\text{-}1}, \text{---}, S(M_{(2)1}, v) \text{ --- }))$$

Fig. 3.2 Signature pairs on encrypted votes

3.1.3 Probabilistic and commutative re-encryptions

Existing encryption algorithms are not commutative. Actually RSA is commutative [24], however it is commutative only when all encryptions are carried out based on the same modulo p arithmetic, therefore not applicable to re-encryption schemes. Namely in RSA, data x is encrypted into $E(k_1, x) = x^{k1} \pmod p$ by encryption key k_1, therefore $E(k_2, E(k_1, x)) = (x^{k1})^{k2} \pmod p = x^{k1k2} \pmod p$, the encrypted form of x generated by authorities TM_1 and TM_2, can be decrypted regardless of the key application order. However, when TM_1's encryption key k_1 is disclosed, it is easy for TM_2 to calculate TM_1's decryption key k_1^{-1}, from the relation $k_1 k_1^{-1} \pmod p = k_2 k_2^{-1} \pmod p$. To disable authorities to calculate decryption keys of other authorities, individual encryption algorithms must adopt different modulo arithmetic, and the resulting re-encryption scheme becomes not commutative. The other typical encryption algorithm ElGamal [25] is not commutative from the beginning [22]. Although in re-encryption mixnet based on ElGamal, the commutative property exists, it is not suitable for the proposed e-voting scheme because it requires ZKP for verifying correct encryptions.

A multiple encryption and signature scheme for votes and *CN*s described in Secs. 3.1.1 and 3.1.2 can be implemented based on the probabilistic encryption algorithm with homomorphic and commutative properties, proposed in [22]. In the election, different voters may choose the same candidates, therefore the encryption function must be probabilistic; if

not probabilistic, same candidates are encrypted into same forms, and a voter can know votes of other voters who had chosen the same candidate even they are encrypted. Also to ensure the authenticity of votes, the encryption and signing algorithms must be commutative. When they are not commutative, the signed form of encrypted vote $S(M_*, E(K_*, v))$ cannot be decrypted to $S(M_*, v)$.

To use re-encryption scheme proposed in [22], each election authority TM_i defines its encryption and decryption key pairs $\{K_i, F_i\}$ and $\{H_i, Gi\}$, while selecting two large appropriate integers p_1 and p_2, where for any integer u and w, $u^{KiFi} \pmod{p_1} = u \pmod{p_1}$ and $w^{HiGi} \pmod{p_2} = w \pmod{p_2}$. Then election authority TM_i encrypts x to $E(\{K_i, H_i\}, x) = \{E(K_i, xr) = (xr)^{Ki}, E(H_i, r) = r^{Hi}\}$ while mixing x with random secret number r as shown in Fig. 3.3 *i.e.* the encrypted form consists of a pair of data part $E(K_i, xr)$ and a randomixation part $E(H_i, r)$. Here, the key pairs are kept as TM_i's secrets, in order to enable each TM_i to securely use its key pairs under the environment where multiple election authorities share the same modulo arithmetic. When key K_i of election authority TM_i is disclosed, it is easy for another election authority TM_j to calculate TM_i's decryption key F_i from the relation $K_iF_i \pmod{\varphi(p_1)} = K_jF_j \pmod{\varphi(p_1)}$ where $\varphi(p_1) = p_1 - 1$ when p_1 is a prime number, for example. In the following u^{Ki}, w^{Hi}, $u^{K1---KP}$ and $w^{H1---HP}$ are denoted as $E(K_i, u)$, $E(H_i, w)$, $E(K_*, u)$ and $E(H_*, w)$ respectively.

data part	randomization part
$E(K_i, xr) = (xr)^{Ki} \pmod{p_1}$	$E(H_i, r) = r^{Hi} \pmod{p_2}$

Fig. 3.3 Encrypted form of x

To repeatedly encrypt voter V_j's vote v_j to $E(\{K_i, H_i\}, v_j)$, V_j generates its secret random number r_j and asks election authorities TM_1, ---, TM_P to encrypt v_jr_j and r_j^{Lj}; where $\{L_j, Z_j\}$ is secret encryption and decryption key pair of V_j, and *TMs* cannot calculate v_j from v_jr_j and r_j^{Lj}, because r_j is secret of V_j and the calculation of r_j from r_j^{Lj} is a discrete logarithm problem. Then TM_1, ---, TM_P repeatedly encrypt the pair $\{v_jr_j, r_j^{Lj}\}$, *i.e.* calculate $E(K_*, v_jr_j)$ and $E(H_*, r_j^{Lj})$ by their encryption keys K_1, ---, K_P and H_1, ---, H_P, and finally V_j calculates $E(H_*, r_j^{Lj})^{Zj} = E(H_*, r_j)$, and constructs $E(\{K_*, H_*\}, v_j)$ as a pair $\{E(K_*, v_jr_j), E(H_*, r_j)\}$. $E(\{K_*, H_*\}, v_j)$ can be decrypted into v_j by calculating $E(K_*, v_jr_j)^{F1---FP} = (v_jr_j)^{(K1---KP)(F1---FP)} = v_jr_j$ and $E(H_*, r_j)^{(G1---GP)} = r_j^{(H1---HP)(G1---GP)} = r_j$ by decryption keys F_1, ---, F_P and G_1, ---, G_P, and by dividing v_jr_j by r_j.

For the confirmation of correct encryptions of *TMs*, V_j asks TM_1, ---, TM_P to decrypt $E(K_*, (v_j r_j)^{Aj})$ and $E(H_*, r_j^{Bj})$, where $\{A_j, B_j\}$ are secret random numbers of V_j. Here, TM_1, ---, TM_P cannot decrypt $E(K_*, (v_j r_j)^{Aj})$ and $E(H_*, r_j^{Bj})$ into $(v_j r_j)^{Aj}$ and r_j^{Bj} when they calculate $E(K_*, v_j r_j)$ and $E(H_*, r_j^{Lj})$ dishonestly, because they do not know A_j, B_j, $v_j r_j$ or r_j. Therefore although K_i and H_i of each TM_i are secret, V_j can confirm the correctness of encryptions as same as it is using public keys. It is apparent that this encryption scheme is probabilistic and commutative. Fortunately, it is also homomorphic, *e.g.* $E(K_*, x_1)E(K_*, x_2) = x_1^{K1---KP} x_2^{K1---KP} = E(K_*, x_1 x_2)$ and $E(H_*, y_1)E(H_*, y_2) = y_1^{H1---HP} y_2^{H1---HP} = E(H_*, y_1 y_2)$. The generation process of re-encrypted form of vote v_j as a pair of $E(K_*, v_j r_j)$ and $E(H_*, r_j)$ is shown in Fig. 3.4.

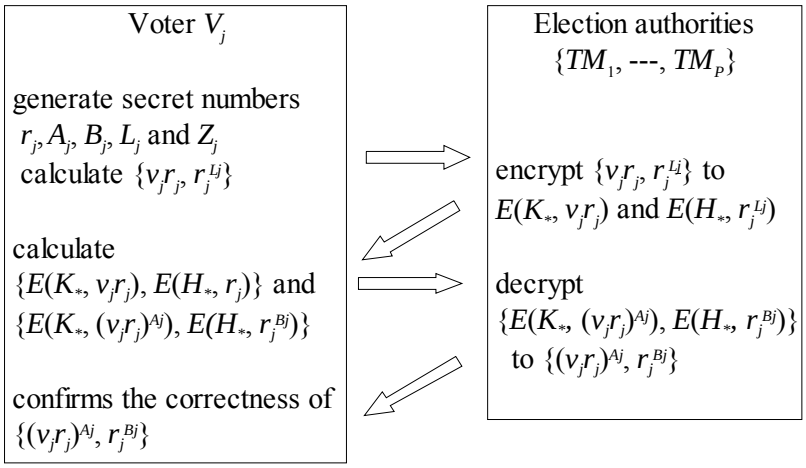


Fig. 3.4 Re-encryption process of vote

In the above procedure, as V_j knows its secret random number r_j, coercers can use this r_j to identify V_j's vote. To disable vote identification, V_j also multiplies its vote v_j by random number x_{xj} that is not known to anyone, where $x_{xj} = x_{xj1} x_{xj2} --- x_{xjP}$. Namely, each TM_i generates its secret random number x_{xji}, and encrypts x_{xji} by its encryption keys K_i and H_i *i.e.* calculates $\{E(K_i, x_{xji}), E(H_i, x_{xji})\}$ and asks other Tallying managers to calculate $\{E(K_*, x_{xji}), E(H_*, x_{xji})\}$. Then by using the homomorphic property, $\{E(K_*, x_{xj}), E(H_*, x_{xj})\}$ is generated by multiplying P different $\{E(K_*, x_{xji}), E(H_*, x_{xji})\}$. Because each TM_i knows only x_{xji}, no one can know the decrypted form of $\{E(K_*, x_{xj}), E(H_*, x_{xj})\}$.

For the confirmation of correct encryptions of x_{xj} *i.e.* to confirm that $E(K_*, x_{xji})$ and $E(H_*, x_{xji})$ are the encrypted forms of the same number, V_j calculates $\{E(K_*, x_{xji}), E(H_*, x_{xji})^{Bj}\}$ to ask TM_1, ---, TM_P to decrypt them *i.e.* to calculate $E(K_*, x_{xji})^{F1---FP} = x_{xji}$ and $E(H_*, x_{xji})^{BjG1---GP} = x_{xji}^{Bj}$, for randomly selected i, and from them V_j checks the consistency

between x_{xji} and x_{xji}^{Bj}. When $E(K_*, x_{xji})$ and $E(H_*, x_{xji})$ are calculated dishonestly, TM_1, ---, TM_P cannot decrypt $E(K_*, x_{xji})$ and $E(H_*, x_{xji})^{Bj}$ into x_{xji} and x_{xji}^{Bj} because they do not know B_j. Because (P-1) remaining x_{xji}s are still unknown to anyone except TM_i, no one can know the decrypted form of x_{xj} unless all TMs conspire, and V_j can calculate $E(\{K_*, H_*\}, v_j) = \{E(K_*, v_j r_j x_{xj}), E(H_*, r_j x_{xj})\}$ while making $r_j x_{xj}$ unknown to anyone. Here, to maintain the equality of two forms of x_{xj}, *i.e.* $x_{xj} \pmod{p_1}$ and $x_{xj} \pmod{p_2}$, each x_{xji} must be defined so that x_{xj} is less than p_1 and p_2. Figure 3.5 describes the construction and attachment process of unknown random number.

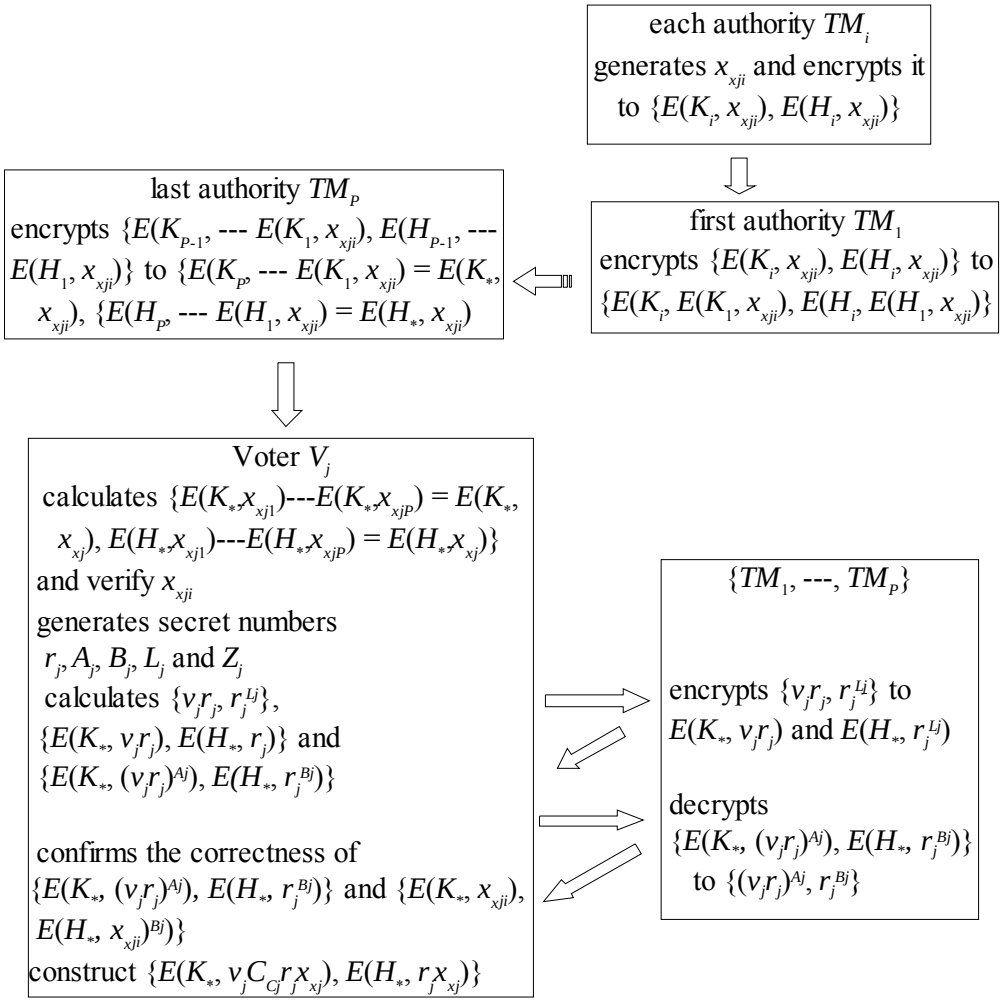


Fig. 3.5 Secret number $r_j x_j$ unknown to anyone

Repeatedly signing mechanisms on re-encrypted forms can be implemented in the same way. However, each TM_i can calculate signing keys $\{M_{(1)k}, M_{(2)k}\}$ of other TM_k when its verification keys $\{U_{(1)k}, U_{(2)k}\}$ are disclosed from the relation $M_{(1)k} = M_{(1)i} U_{(1)i} / U_{(1)k}$ and $M_{(2)k} = M_{(2)i} U_{(2)i} / U_{(2)k}$. Therefore, verification keys must be disclosed only after all votes are decrypted. In the proposed scheme all votes are put in bulletin board (*BB*), where a *BB* is a

public broadcast channel with memories, and information sent to a *BB* is readable by anyone and at anytime. Then, no one can forge signatures on votes in *BB* even the signing keys are revealed *i.e.* before the disclosure of verification keys, no one knows all signing keys; and after the disclosure of verification keys, votes are already decrypted and cannot be modified. Here, V_j can confirm the correctness of signatures without knowing the verification keys in the same way as in the encryption processes.

In Sec. 3.1.1 *CN*s are encrypted without being mixed with random numbers and P multiple independent election authorities repeatedly encrypt CN_j simply into $(CN_j)^{K1\text{---}KP}$ by using their secret encryption keys K_1, ---, K_P. Here, probabilistic encryption is not necessary, because all *CN*s are unique and all of their encrypted forms are different.

Probabilistic and commutative re-encryption schemes also can be constructed based on ElGamal or threshold ElGamal encryption. However, to identify dishonest managers without disclosing privacies of voters, ElGamal based schemes require complicated ZKP processes even if *CN*s are exploited.

3.1.4 Signature pairs on blinded tokens

Voters can act without disclosing their identities while showing their eligibility by using tokens. Namely, voter V_j encrypts its token T_j to $E(a_j, T_j)$ by using its secret key a_j, and while confirming the identity of V_j by usual means *e.g.* through an *ID* and a password of V_j, multiple mutually independent election managers TM_1 --- TM_P blindly sign on $E(a_j, T_j)$ [23] to generate two different sets *i.e.* $\{S(X_{(1)1}, E(a_j, T_{tj})), \text{---}, S(X_{(1)P}, E(a_j, T_{tj}))\} = S(X_{(1)*}, E(a_j, T_{tj}))$ and $\{S(X_{(2)1}, E(a_j, T_{tj})), \text{---}, S(X_{(2)P}, E(a_j, T_{tj}))\} = S(X_{(2)*}, E(a_j, T_{tj}))$ as shown in Fig. 3.6 by using their signing keys $\{X_{(1)1}, X_{(1)2}, \text{---}, X_{(1)P}\}$ and $\{X_{(2)1}, X_{(2)2}, \text{---}, X_{(2)P}\}$, and V_j decrypts them into two unblinded sets of signed tokens *i.e.* $\{S(X_{(1)1}, T_{tj}), \text{---}, S(X_{(1)P}, T_{tj})\} = S(X_{(1)*}, T_{tj})$ and $\{S(X_{(2)1}, T_{tj}), \text{---}, S(X_{(2)P}, T_{tj})\} = S(X_{(2)*}, T_{tj})$. Then, because *TM*s had signed without knowing T_{tj}, V_j can show its eligibility for putting its vote without disclosing itself, by showing $S(X_{(1)*}, T_{tj})$. Also only V_j can approve the registration of its vote while proving its eligibility by $S(X_{(2)*}, T_{tj})$, because no one except V_j knows $S(X_{(2)*}, T_{tj})$ even after $S(X_{(1)*}, T_{tj})$ had been opened.

$$\{S(X_{(1)1}, E(a_j, T_{tj})), \text{---}, S(X_{(1)P}, E(a_j, T_{tj}))\} = S(X_{(1)*}, E(a_j, T_{tj}))$$
$$\{S(X_{(2)1}, E(a_j, T_{tj})), \text{---}, S(X_{(2)P}, E(a_j, T_{tj}))\} = S(X_{(2)*}, E(a_j, T_{tj}))$$

Fig. 3.6 Signature pairs on blinded tokens

3.2 Existing components

3.2.1 Mixnet like encryptions/decryptions and shuffles

The construction of encrypted confirmation numbers *CNs* by the technique of encryptions and shuffles by multiple mutually independent authorities was discussed in Sec. 3.1.1. Whereas in order to decrypt all encrypted votes, mutually independent election authorities repeatedly perform decryptions and shuffles of votes by using their secret decryption keys. As like decryption mixnet, it also consists of a set of *mix-severs* (already which is written as mutually independent election authorities). Each *mix-server* processes a batch of incoming messages in a way that the outgoing messages are unlinkable to the incoming ones, and forwards the whole batch to the next *mix-server* in the mixnet. Incoming messages are anonymized by removing the messages' encryption layers and by shuffling the order of each batch of messages. Here, multiple decryptions are executed in the order different from encryptions. Figure 3.7 presents a schematic diagram of mixnet like decryptions and shuffles.

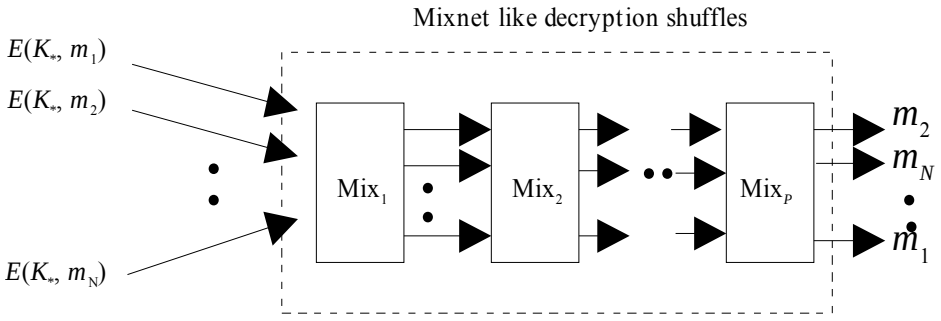


Fig. 3.7 Mixnet like decryption shuffles

As discussed in Sec. 3.1, K_i is the secret encryption key of the *i*-th stage of *i*-th mix-server (*i.e.* election authority). Voter V_j encrypts its message (*i.e.* vote) into $E(K_P, E(K_{P\text{-}1}, \text{---} E(K_1, m_j) \text{---}))$ by the encryption keys of P mutually independent authorities *i.e.* mix-servers and send it to the authorities. Mix_i with its secret decryption key F_i receives inputs as, $E(K_i, E(K_{i\text{-}1}, \text{---} E(K_1, m_j) \text{---}))$ from $\text{Mix}_{i\text{-}1}$. The decryption algorithm can be described as follows.

Input $E(K_P, E(K_{P\text{-}1}, \text{---} E(K_1, m_j) \text{---}))$; $j = 1,\ldots, N$.

For $i = 1,\ldots, P$; {where P is the no. of mix-servers involved in mixnet}

For j = 1,…,N; {where N is the no. of voters participated in voting}

Now $E(K_P, E(K_{P-1}, \text{---} E(K_1, m_j) \text{---}))$ is decrypted to $E(K_{P-1}, \text{---} E(K_1, m_j) \text{---})$. In lexicographically order, all quantities are decrypted in this way to obtain their decrypted form. Output: $\{m_j\}_R$, a batch of mixed messages that cannot be traced back to senders. Decryptions and shuffles according to the proposed cryptosystem of Sec. 3.1.3, requires the voter to perform P encryptions here (source modified from [1]).

3.2.2 Bulletin board (*BB*)

To achieve the vote and voter verifiability requirements, the data of interactions among the entities are publicly disclosed in *BB* at every stage of the scheme. *BB* is a publicly accessible broadcast communication channel with memory. Any information that is exchanged among entities is stored in memory and readable by anyone. Consequently anyone can check its integrity *i.e*. data is only added and old data can not be changed or modified. In short, reliability of a *BB* stems from the fact that it is under constant public inspection [15]. *BB* can be implemented by multiple servers in order to protect it from attackers who may want to modify information in it. These servers may run a distributed Byzantine agreement protocol [33] to ensure the consistency of content, or observers may simply rely on cryptographic signatures of the content and the redundancy of the servers, using, for example hash trees [16].

Existence of such a public *BB* is assumed in many previous works. Many e-voting protocols consider that the designated sections of a *BB* has the correspondence with authenticated voters *i.e*. only authenticated voters can access their designated sections to write data. Some voting schemes consider a special form of *BB* that does not contain any designated sections, in order to remove the traceability between voters and their votes to achieve incoercibility [1]. The voting scheme proposed in this thesis uses both forms of *BB*s *i.e*. with designated sections and without any designated sections.

3.2.3 Blind signature

Signature pairs on blinded tokens discussed in Sec. 3.1.4 are constructed based on the blind signature mechanism. The blind signature protocol proposed in [23] is based on the RSA digital signature algorithm [24]. It is assumed that to get a signature on its token T_{tj},

voter V_j encrypts T_{tj} to $E(a_j, T_{tj})$ by using its secret encryption key a_j. Here the public key is denoted as a pair (E, W) and the secret signing key is denoted as a number X.

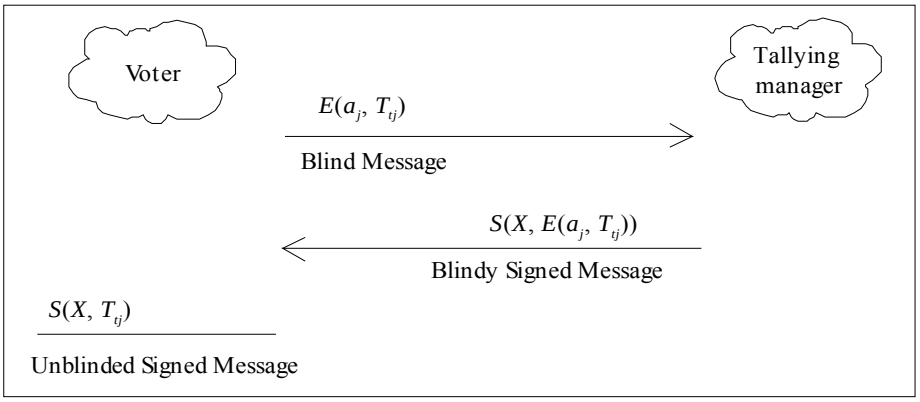


Fig. 3.8 Steps of RSA blind signature mechanism proposed in [23]

Blind signature protocol proposed in [23] then consists of the following three steps as shown in Fig. 3.8:

Blinding:

V_j picks its secret encryption key a_j, which is a random integer between 0 and n, and computes the value: $E(a_j, T_{tj})$ mod W. V_j sends $E(a_j, T_{tj})$ to the Tallying manager TM_i. The $E(a_j, T_{tj})$ is the blinded token to be signed by TM_i and not the original message *i.e.* T_{tj}.

Signing:

TM_i uses its private key X to compute the value: $S(X, E(a_j, m))$ mod W. Then TM_i returns $S(X, E(a_j, T_{tj}))$ to V_j.

Unblinding:

V_j extracts the signature: $S(X, T_{tj})$ mod W by using its secret decryption key b_j. So, V_j ends up with a pair (T_{tj}, S) satisfying the equation $S = T_{tj}^{\ X}$ mod n. This is exactly the verification relation for standard RSA signatures. It should be noted that TM_i doesn't know on which message T_{tj} it had actually signed [27].

3.2.4 Anonymous authentication mechanism

Anonymous authentication mechanism proposed in [20] enables voter V_j to be authenticated as authorized one without disclosing its identity, based on *password selection* strategy. According to this mechanism a single entity (or entities) determines whether V_j, who is characterized by its identifier ID_j and password P/W_j pair, is an authorized one or not. According to password selection strategy, V_j does not show its ID_j and P/W_j pair to the

entity. Instead of showing an ID_j and P/W_j pair, V_j finds its P/W_j in the password list, which is generated by the entity, and declares whether its P/W_j is included in the list or not. For details, see [20] and the behavior of this mechanism is shown in Fig. 3.9.

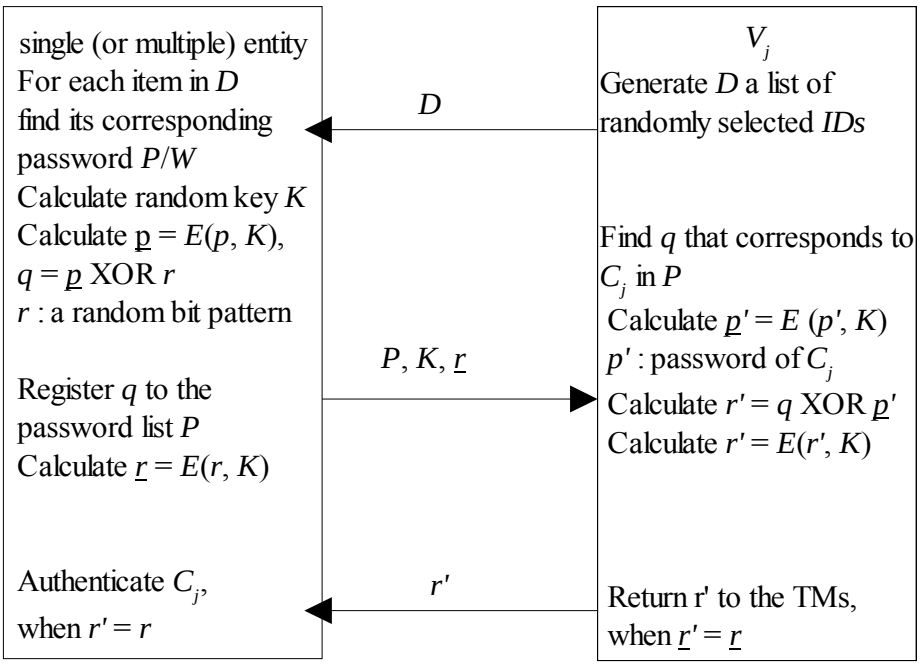


Fig. 3.9 Behavior of anonymous authentication mechanism

Chapter 4

Configuration of the Voting Scheme

This chapter describes the configurations of the voting scheme *i.e.* the involved entities, their roles and the overview of the scheme are presented here.

4.1 Entities and their roles

Entities involved in the scheme are N voters V_j (j = 1,---,N), election authorities *i.e.* Voting manager *VM*, P (at least two) mutually independent Tallying managers TM_i (i = 1,---,P), Disruption detection manager *DM* and six public *BB*s that maintain authorized communication transcripts *i.e.* *VoterList*, *TokenList*, *ConfNoList*, *ActiveTokenList*, *VotingPanel* and *TallyingPanel*. Figure 4.1 depicts the configurations of individual *BB*s. By putting relevant information on several *BB*s, interactions among the entities at every stage of the election become publicly verifiable. The relationships among the entities are shown in Fig. 4.2. In the followings V_j is the j-th voter, v_j is the vote of V_j, and C_{Cj}, T_{tj} and x_{xj} are the *CN*, token and unknown random number assigned to V_j. ID_j and P/W_j are the identifier and password of V_j. The roles of each entity and the descriptions of *BB*s are as follows:

Voter V_j: Each V_j generates its encrypted vote $E(\{K_*, H_*\}, v_jC_{Cj})$ while combining it with its assigned confirmation number CN_j *i.e.* $E(K_*, CN_j)$, and then puts and approves it in *VotingPanel*. It has its own identifier (ID_j) and password (P/W_j) that characterize V_j as unique, and two secret encryption and decryption key pairs $\{a_j, g_j\}$, and $\{Lj, Zj\}$. ID_j and P/W_j pair proves the eligibility of V_j. Key pair $\{a_j, g_j\}$ is used to acquire two different forms of signatures of all Tallying managers *TM*s on its token T_{tj} blindly *i.e.* $S(X_{(1)*}, T_{tj})$ and $S(X_{(2)*}, T_{tj})$, and key pairs $\{L_j, Z_j\}$ is used to ask *TM*s to encrypt vote v_j without disclosing v_j itself.

Voting manager *VM:* *VM* is responsible for authenticating voters, for assigning confirmation numbers *CN*s to voters, and for putting encrypted votes of voters in

VotingPanel. It also puts other data about voters and votes in *VoterList*, *TokenList*, *ConfNoList*, *ActiveTokenList*, and *VotingPanel*. *VM* can be constructed by multiple independent entities to distribute its responsibility if necessary.

Tallying managers *TMs*: Mutually independent P ($P \geq$ two) *TMs* sign on blinded tokens, perform encryptions and shuffles of confirmation numbers *CNs* and votes, repeatedly sign on encrypted votes and encrypted *CNs* in *VotingPanel*, and perform decryptions and shuffles of votes in *VotingPanel* to compute the tally and to put results on *TallyingPanel*. For encryption and decryption of votes and *CNs*, each TM_i has two encryption and decryption key pairs $\{K_i, F_i\}$ and $\{H_i, G_i\}$. Also to sign on blinded token $E(a_j, T_{tj})$, TM_i has two signing and verification key pairs *i.e.* $\{X_{(1)i}, B_{(1)i}\}$ and $\{X_{(2)i}, B_{(2)i}\}$, and to repeatedly sign on encrypted votes and encrypted *CNs* in two different forms, it has four secret signing and verification key pairs $\{\{M_{(1)i}, U_{(1)i}\}, \{Q_{(1)i}, W_{(1)i}\}\}$ and $\{\{M_{(2)i}, U_{(2)i}\}, \{Q_{(2)i}, W_{(2)i}\}\}$.

Disruption detection manager *DM*: *DM* detects inconsistent votes in *TallyingPanel*, and when inconsistencies are detected it identifies the entities that cause the inconsistencies.

ID	token
ID_1	$E(a_1,T_{t1})$
⋮	⋮
ID_j	$E(a_j,T_{tj})$
⋮	⋮
ID_N	$E(a_N,T_{tN})$

(a) VoterList

token	flag
T_1	unused
⋮	⋮
T_{tj}	used
⋮	⋮
T_N	used

(b) TokenList

CN	random number
$E(K_*,C_k)$	$E(K_*,x_k),E(H_*,x_k)$
⋮	⋮
$E(K_*,C_{Cj})$	$E(K_*,x_{xj}),E(H_*,x_{xj})$
⋮	⋮
$E(K_*,C_u)$	$E(K_*,x_u),E(H_*,x_u)$

(c) ConfNoList

token	*CN*
$S(X_{(1)*},T_1)$	$E(K_*,C_v)$
⋮	⋮
$S(X_{(1)*},T_{tj})$	$E(K_*,C_{Cj})$
⋮	⋮
$S(X_{(1)*},T_N)$	$E(K_*,C_s)$

(d) ActiveTokenList

vote	approval
$S(\{M_{(1)*},Q_{(1)*}\},E(\{K_*,H_*\},v_qC_q)),S(M_{(1)*},E(K_*,C_q)),S(\{M_{(2)*},Q_{(2)*}\},E(\{K_*,H_*\},v_qC_q))$	$S(X_{(2)*},T_1)$
⋮	
$S(\{M_{(1)*},Q_{(1)*}\},E(\{K_*,H_*\},v_jC_{Cj})),S(M_{(1)*},E(K_*,C_{Cj})),S(\{M_{(2)*},Q_{(2)*}\},E(\{K_*,H_*\},v_jC_{Cj}))$	$S(X_{(2)*},T_{tj})$
⋮	
$S(\{M_{(1)*},Q_{(1)*}\},E(\{K_*,H_*\},v_hC_h)),S(M_{(1)*},E(K_*,C_h)),S(\{M_{(2)*},Q_{(2)*}\},E(\{K_*,H_*\},v_hC_h))$	$S(X_{(2)*},T_N)$

(e) VotingPanel

vote	*CN*
$S(\{M_{(1)*},Q_{(1)*}\},v_gC_n)$ $S(\{M_{(2)*},Q_{(2)*}\},v_gC_n)$	$S(M_{(1)*},C_n)$
⋮	
$S(\{M_{(1)*},Q_{(1)*}\},v_jC_{Cj})$ $S(\{M_{(2)*},Q_{(2)*}\},v_jC_{Cj})$	$S(M_{(1)*},C_{Cj})$
⋮	
$S(\{M_{(1)*},Q_{(1)*}\},v_eC_r)$ $S(\{M_{(2)*},Q_{(2)*}\},v_eC_r)$	$S(M_{(1)*},C_r)$

(f) TallyingPanel

Fig. 4.1 Configurations of bulletin boards

VoterList: *VoterList* enables audiences including voters themselves to know eligible voters and voters who have been registered *i.e.* who acquired signatures of Tallying managers *TMs* on their tokens. It consists of *ID* and token parts. *ID* part maintains *IDs* of eligible voters, and when V_j registers itself, Voting manager *VM* puts $E(a_j, T_{tj})$, a token encrypted by voter

V_j, at the token part corresponding to V_j's *ID* when V_j shows it to obtain *TMs*' signatures on it as shown in Fig. 4.1 (a). However no one except voters themselves can know tokens on which *TMs* had signed.

TokenList: *TokenList* consists of the token and used flag parts, and enables voters to acquire tokens without collision. The token part maintains tokens *i.e.* unique numbers prepared by Voting manager *VM*. When voter V_j picks token T_{tj} from *TokenList* anonymously, *VM* gives it to V_j while making the corresponding flag part used as shown in Fig. 4.1 (b).

ConfNoList: It consists of *CN* and random number parts, and for *N* voters, *N* different confirmation number *CN*s and unknown random numbers are generated and each *CN* and random number pair $\{C_{Cj}, x_{xj}\}$ is encrypted to $E(K_*, C_{Cj})$ and $\{E(K_*, x_{xj}), E(H_*, x_{xj})\}$ to be posted here at random by Voting manager *VM* as shown in Fig. 4.1 (c).

ActiveTokenList: It consists of the token and *CN* parts, and enables anyone to know anonymous V_j who had been assigned CN_j in its encrypted form. The t_j-th position of the token part maintains the first signed form of the t_j-th token T_{tj} *i.e.* $S(X_{(1)*}, T_{tj})$ of V_j who had acquired CN_j. The corresponding *CN* part maintains encrypted C_{Cj}, *CN* assigned to the voter who obtains T_{tj} *i.e.* $E(K_*, CN_j)$ as shown in Fig. 4.1 (d). Here, by comparing the items in *ActiveTokenList*, *ConfNoList* and *VoterList*, anyone can verify that only voters with their signed tokens acquire *CN*s, and *VM* is not misusing or adding any extra *CN* illegally.

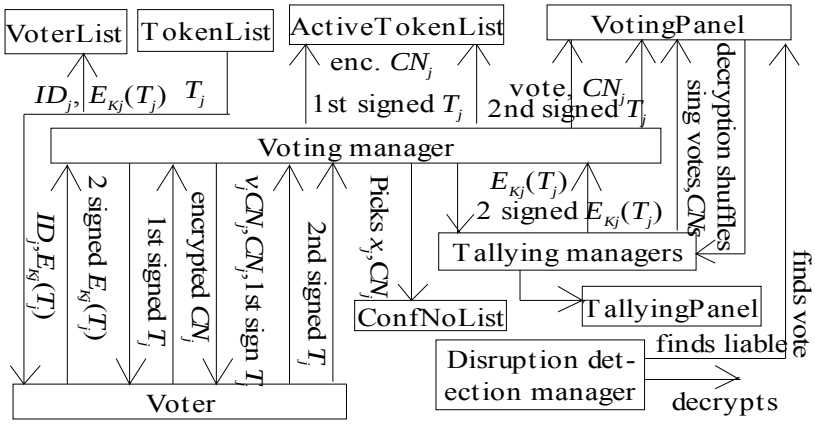


Fig. 4.2 Relationships among the entities of the scheme

VotingPanel: *VotingPanel* consists of the vote and the approval parts, and enables anyone to know encrypted votes approved by their voters. The vote part corresponding to the *tj*-th

position maintains two different signed forms of encrypted vote of the voter to whom the t_j-th token T_{tj} is assigned. Namely it maintains anonymous V_j's encrypted vote v_j repeatedly signed by two secret signing key pairs $\{M_{(1)i}, Q_{(1)i}\}$ and $\{M_{(2)i}, Q_{(2)i}\}$ of all TM_i and encrypted C_{Cj} in the first signed form *i.e.* $S(\{M_{(1)*}, Q_{(1)*}\}, E(\{K_*, H_*\}, v_jC_{Cj}))$, $S(\{M_{(2)*}, Q_{(2)*}\}, E(\{K_*, H_*\}, v_jC_{Cj}))$ and $S(M_{(1)*}, E(K_*, C_{Cj}))$, and the approval part maintains the second signed form of T_{tj} *i.e.* $S(X_{(2)*}, T_{tj})$ as shown in Fig. 4.1 (e). In the above, $S(\{M_{(h)*}, Q_{(h)*}\}, E(\{K_*, H_*\}, x))$ represents pair $\{S(M_{(h)*}, E(K_*, xr)), S(Q_{(h)*}, E(H_*, r))\}$ for h = 1 and 2, provided that r is a secret random number used for encrypting x to $E(\{K_*, H_*\}, x))$.

TallyingPanel: *TallyingPanel* consists of the vote part and *CN* part and enables anyone to know the election results. It maintains decrypted data of *VotingPanel i.e.* the vote part maintains $\{S(\{M_{(1)*}, Q_{(1)*}\}, v_jC_{Cj}), S(\{M_{(2)*}, Q_{(2)*}\}, v_jC_{Cj})\}$ and the *CN* part maintains $S(M_{(1)*}, C_{Cj})$ as shown in Fig. 4.1 (f). Based on *CNs*, anyone can verify that only and all votes from eligible voters are included in *TallyingPanel*. However, because votes on *VotingPanel* are decrypted while being shuffled to be put on *TallyingPanel*, no one can identify linkages between voters and their votes.

4.2 Overview of the scheme

The proposed voting scheme consists of five major stages as follows. The relationships and the data flows among the modules of the scheme are shown in Fig. 4.3.

4.2.1 Token acquisition: Anonymously authenticated voter V_j picks unique token T_{tj} while maintaining tokens collision free.

4.2.2 Registration: Voter V_j whose eligibility is checked by its identifier ID_j and password P/W_j pair obtains two kinds of blind signatures of Tallying managers on T_{tj} *i.e.* $S(X_{(1)*}, E(a_j, T_{tj}))$ and $S(X_{(2)*}, E(a_j, T_{tj}))$. Therefore later on V_j can prove its eligibility by showing decrypted signatures $S(X_{(1)*}, T_{tj})$ and $S(X_{(2)*}, T_{tj})$, without disclosing its identity. Here a_j is a secret encryption key of V_j and $X_{(1)i}$ and $X_{(2)i}$ are the signing keys of Tallying manager TM_i.

4.2.3 Voting: This stage consists of two sub-stages.

> ***CN* Assignment:** V_j proves its eligibility by showing $S(X_{(1)*}, T_{tj})$ and obtains repeatedly encrypted confirmation number C_{Cj} *i.e.* $E(K_*, C_{Cj})$ and encrypted unknown random number $\{E(K_*, x_{xj}), E(H_*, x_j)\}$ from Voting manager *VM*. Also, V_j

verifies the correctness of encryption of $\{E(K_*, x_{xj}), E(H_*, x_j)\}$.

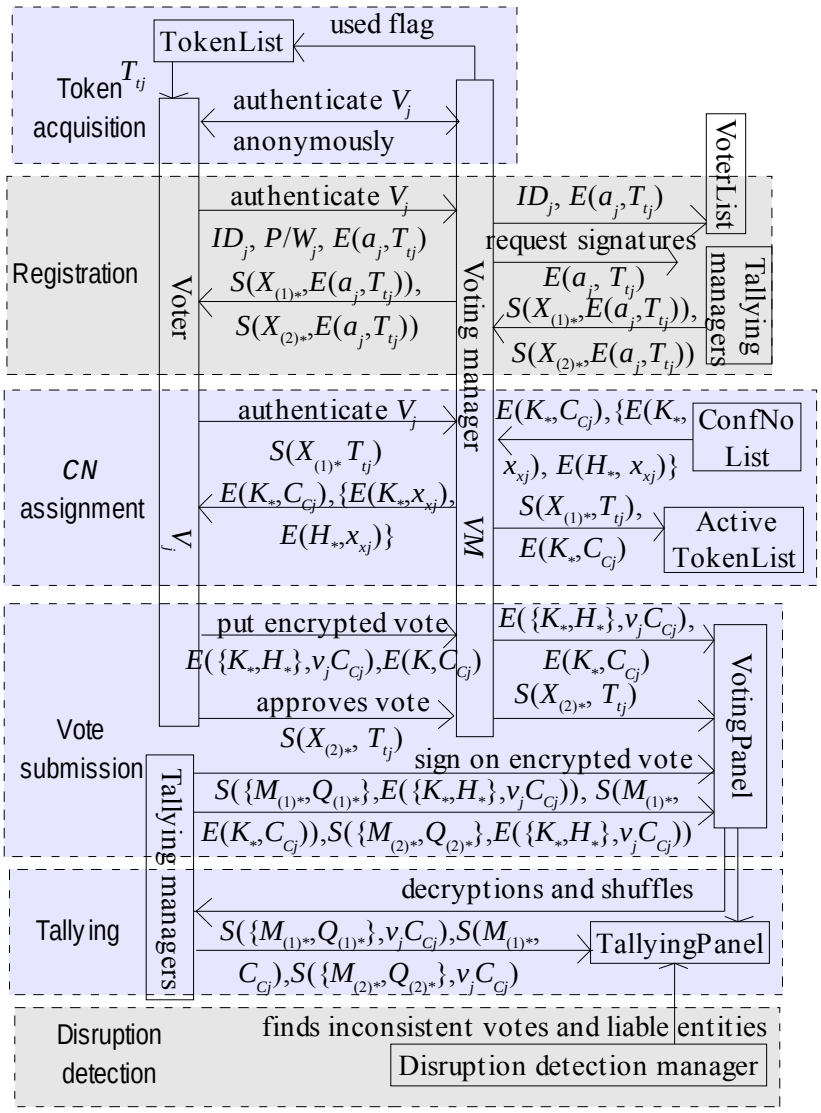


Fig. 4.3 Relationships and data flow among the modules of the scheme

Vote Submission: V_j asks Tallying managers to repeatedly encrypt its vote v_j to $E(\{K_*, H_*\}, v_j)$ while randomizing it by secret numbers r_j and x_{xj} and verify its correctness. Then V_j calculates $\{E(\{K_*, H_*\}, v_jC_{Cj}), E(K_*, C_{Cj})\}$ while combining $E(\{K_*, H_*\}, v_j)$ with its assigned $E(K_*, C_{Cj})$ and submits it as its vote, and TM_1,---, TM_P sign on them by the first form of their signatures *i.e.* calculate $S(\{M_{(1)*}, Q_{(1)*}\}, E(\{K_*, H_*\}, v_jC_{Cj}))$ and $S(M_{(1)*}, E(K_*, C_{Cj}))$. After checking its vote on *VotingPanel*, V_j approves the registration of its vote by $S(X_{(2)*}, T_{tj})$, and finally TM_1,---,TM_P sign on $E(\{K_*, H_*\}, v_jC_{Cj})$ by the second form of their signatures *i.e.* calculate $S(\{M_{(2)*}, Q_{(2)*}\}, E(\{K_*, H_*\}, v_jC_{Cj}))$.

4.2.4 Tallying: Multiple decryptions and shuffles of votes in *VotingPanel* by Tallying managers compute the election results and they are disclosed on *TallyingPanel* while concealing links between encrypted and disclosed votes in *VotingPanel* and *TallyingPanel*. However *CNs* attached to individual votes ensure that all and only eligible votes are counted.

4.2.5 Disruption detection: If inconsistency is found for any disclosed vote, the responsible entity is detected while maintaining the privacy of voters.

Chapter 5

Proposed Voting Scheme

This Chapter describes the individual stages of the proposed voting scheme in detail. Here individual stages proceed as follows:

5.1 Token acquisition stage

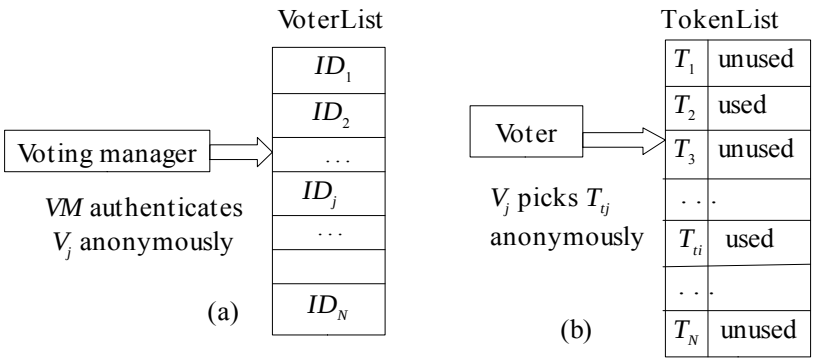


Fig. 5.1 For eligible voter: (a) anonymous authentication, and (b) anonymously token acquisition procedures.

An objective of this stage is to assign voter V_j a token T_{tj} which is unique in the system while maintaining anonymity of V_j. To achieve this objective, anonymously authenticated V_j picks T_{tj} from *TokenList* prepared by Voting manager *VM*. Because voters can pick tokens that are not picked by other voters, the uniqueness of tokens are maintained. Figure 5.1 (a) shows that *VM* anonymously authenticates voters in *VoterList*. Theoretically, V_j authentication is not necessary for this step. But by protecting T_{tj} from being picked by unauthorized entities, it becomes possible to make *TokenList* as small as possible *i.e.* unauthorized entities cannot request tokens. However, more than N different numbers are generated as tokens in advance and they are put in *TokenList* to be picked by voters; where N is the number of eligible voters. Fig. 5.1 (b) shows that V_j anonymously picks its T_{tj} from

TokenList.

To enforce V_j to pick T_{tj} from *TokenList*, every T_{tj} has the signature of Voting manager *VM* (this signature is different from $S(X_{(1)*}, T_{tj})$ and $S(X_{(2)*}, T_{tj})$, and ensure that T_{tj} is picked from *TokenList*). Because of this signature, entities cannot misuse tokens in unauthorized ways. This signature is accomplished by the digital signature mechanism [24, 25]. However tokens in *TokenList* are open to the public only in non-signed forms to disable entities to use them in unauthorized ways. It is possible to maintain privacies of voters even when they are involved in dishonest events as shown at the end of Sec. 5.5.

In this stage, V_j and *VM* interact as follows:

1. *VM* authenticates eligible V_j anonymously *e.g.* through anonymous authentication mechanism [20].
2. Authenticated V_j picks unused token T_{tj} from *TokenList* and asks *VM* to sign on T_{tj} (this signature is omitted in the following notations).
3. In order to avoid collision, *VM* makes T_{tj} in *TokenList* used as shown in Fig. 5.1 (b).

Security problems of this stage are token collision, firstly voters may get multiple tokens and *VM* may not issue tokens to eligible voters.
These problems are solved as bellow:

- *Voters may get already picked tokens:*
  Here the signature of *VM* on T_{tj} ensures that the token is picked from *TokenList*. Also the used mark on flag part of *TokenList* disable voters to get already picked tokens.

- *Voters may get multiple tokens:*
  Because only tokens with signatures of Tallying managers, which are given at the registration stage while confirming the eligibility of individual voters, are effective, voters can use only single tokens even they get multiple tokens.

- *Voters may not get tokens*:
  As multiple tokens cause no inconvenience, V_j can request its token assignment repeatedly.

5.2 Registration stage

Objectives of this stage are: (1) to let Tallying managers (*TMs*) sign on token T_{tj} that is shown by eligible voter V_j without knowing T_{tj} itself [23], so that V_j can show its eligibility anonymously by it at the later stages, and (2) to let all entities know signed T_{tj} that is assigned to V_j. Here the pair of signatures *i.e.* $S(X_{(1)*}, T_{tj})$ and $S(X_{(2)*}, T_{tj})$ are generated by *TMs* through the blind signature scheme [23], as described in Sec. 3.1.4. To make voters that obtain signed tokens publicly visible, *VM* maintains *VoterList*, as shown in Fig. 4.1 (a), but at this stage V_j shows T_{tj} in its blinded form, *i.e. VM* puts $E(a_j, T_{tj})$ in *VoterList*. Therefore anyone can monitor V_j who is registered, however, only V_j knows its token T_{tj}. As a consequence, V_j can abstain from vote submission without being noticed even it is registered in *VoterList* for example. Figure 5.2 shows the interactions of this stage.

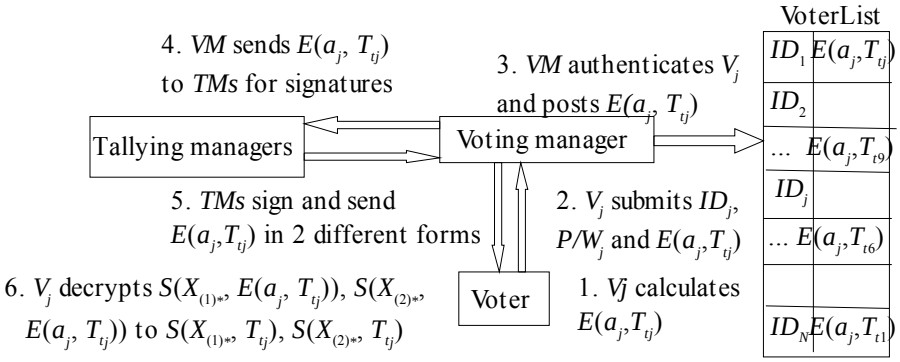


Fig. 5.2 Voter registration procedures

The interactions between V_j and *VM* in this stage are as follows:

1. V_j encrypts T_{tj} by using its secret encryption key K_j *i.e.* V_j calculates $E(a_j, T_{tj})$.
2. V_j shows its ID_j, P/W_j and its blinded token $E(a_j, T_{tj})$ to *VM*.
3. *VM* authenticates V_j based on ID_j and P/W_j and after the successful authentication, it posts $E(a_j, T_{tj})$ in *VoterList* so that anyone can know the anonymous V_j who has registered.
4. *VM* sends $E_{Kj}(T_{tj})$ to Tallying managers for their signatures, and mutually independent TM_1,---,TM_P sign on $E(a_j, T_{tj})$ with their two different signatures *i.e.* calculate $S(X_{(1)*}, E(a_j, T_{tj}))$ and $S(X_{(2)*}, E(a_j, T_{tj}))$.
5. TM_1,---,TM_P send them to *VM* to be sent to V_j.
6. V_j checks the validity of signatures on T_{tj} and decrypts $S(X_{(1)*}, E(a_j, T_{tj}))$ and $S(X_{(2)*}, E(a_j, T_{tj}))$ to $S(X_{(1)*}, T_{tj})$ and $S(X_{(2)*}, T_{tj})$.

Here the third step ensures that ineligible V_j cannot obtain signed T_{tj} and eligible V_j cannot get multiple signed tokens. Also the fourth step ensures that anyone cannot forge signed tokens unless all Tallying managers conspire. Security problems of this stage are as follows:

- *Multiple entities request signatures on T_{tj} picked by voter V_j:*
  By this threat, V_j's vote will be rejected. There are two possibilities, the first one occurs when signed T_{tj} is stolen, however V_j is responsible for that. The other possibility is a case where *VM* uses signed T_{tj}. This possibility can be excluded, if necessary, by duplicating *VM*, *i.e*. no entity can obtain signatures of all *TMs* on T_{tj} in unauthorized ways unless all *VMs* conspire.

- *Voters cannot get correct signed tokens:*
  V_j can prove *VM*'s dishonesty by showing $E(a_j, T_{tj})$ and the incorrect signed token. However these situations do not happen, because dishonesty of authorities are revealed.

- *Voting manager puts its forging $E(a_j, T_{tj})$:*
  Already discussed in the first security problem of this section, this dishonesty also can be protected by duplicating *VM*, *i.e*. forged blinded tokens cannot be accepted if at least one *VM* is honest.

5.3 Voting stage

Objectives of this stage are: (1) to assign confirmation number C_{Cj} to anonymous voter V_j without knowing C_{Cj} itself, (2) to disclose used T_{tj} to the public, (3) to enable V_j to put its vote on *VotingPanel* while concealing its identity and vote, (4) to enable V_j to validate its vote on *VotingPanel*, and (5) to let Tallying managers TM_1, ---, TM_P sign on encrypted votes and encrypted *CNs* on *VotingPanel*. This stage consists of two sub-stages, which are: i) *CN* assignment and ii) Vote submission.

5.3.1 *CN* assignment sub-stage

In this sub-stage: (1) Voting manager *VM* authenticates voter V_j anonymously by signed token $S(X_{(1)*}, T_{tj})$, and (2) V_j receives encrypted C_{Cj} *i.e.* $E(K_*, C_{Cj})$ and encrypted

unknown random number $\{E(K_*, x_{xj}), E(H_*, x_{xj})\}$. While *VM* sends $E(K_*, C_{Cj})$ to V_j, it also discloses $E(K_*, C_{Cj})$ and $S(X_{(1)*}, T_{tj})$ in *ActiveTokenList*. Here as shown in Sec 3.1.2, anyone even V_j itself cannot identify the correspondence between original C_{Cj} and $E(K_*, C_{Cj})$, and hence between V_j and C_{Cj}. However, because *CN*s are unique and registered, and no one can forge signatures of all Tallying managers on them, any entity can confirm the accuracy of votes by *CN*s disclosed in *TallyingPanel*. Figure 5.3 shows the interactions of this sub-stage.

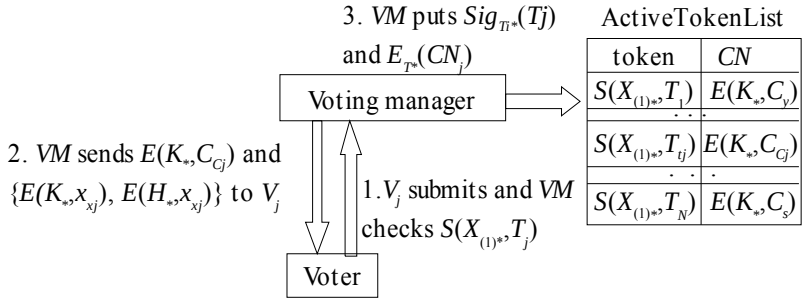


Fig. 5.3 *CN* assignment procedures

The interactions between V_j and *VM* in this sub-stage are as follows:

1. V_j submits $S(X_{(1)*}, T_{tj})$ to *VM* and *VM* checks the validity of $S(X_{(1)*}, T_{tj})$. Here *VM* can verify the authenticity of V_j by checking only the signatures on T_{tj} that is not used repeatedly.
2. *VM* sends $E(K_*, C_{Cj})$ and $\{E(K_*, x_{xj}), E(H_*, x_{xj})\}$ to V_j, and V_j checks the correctness of encryption of $\{E(K_*, x_{xj}), E(H_*, x_{xj})\}$.
3. *VM* also puts $S(X_{(1)*}, T_{tj})$ and $E(K_*, C_{Cj})$ in *ActiveTokenList* as shown in Fig. 4.1 (d).

Security problems of this sub-stage are as follows:

- *Voting manager may put signed tokens in ActiveTokenList before voters*:
  VM knows neither of V_j's secret key nor the signing keys of all Tallying managers, therefore it cannot generate $S(X_{(1)*}, T_{tj})$ from $S(X_{(1)*}, E(a_j, T_{tj}))$ or T_{tj} to put it before V_j *i.e.* it is impossible for *VM* to generate and put $S(X_{(1)*}, T_{tj})$ before V_j.

- *Voting manager may not put signed token in ActiveTokenList*:
  VM cannot deny putting of $S(X_{(1)*}, T_{tj})$ on *ActiveTokenList* because $S(X_{(1)*}, T_{tj})$ has the signatures of Tallying managers.

- *Voting manager may not give C_{Cj}, or give incorrect C_{Cj} to V_j :*
  As $S(X_{(1)*}, T_{tj})$ is open to the public, *VM* cannot deny giving of C_{Cj}. Also as $E(K_*,$

C_{Cj}) is open on *ConfNoList*, *VM* cannot give non-registered C_{Cj}. Although it is possible that *TMs* encrypt C_{Cj} incorrectly, this dishonesty and the responsible entities are detected at the disruption detection stage, therefore *TMs* cannot encrypt *CNs* incorrectly.

5.3.2 Vote submission sub-stage

In this sub-stage: (1) anonymous voter V_j submits its verifiable secret vote, (2) Tallying managers TM_1,---,TM_P repeatedly sign on the vote, (3) after confirming the successful registration of the vote on *VotingPanel*, V_j approves its vote by putting the second signed form of T_{tj} *i.e.* $S(X_{(2)*}, T_{tj})$ in *VotingPanel* as shown in Fig. 4.1 (e), and (4) finally *TMs* repeatedly sign on the vote by the second form of their signatures. Here, $E(K_*, v_j r_j x_j)$ and $E(H_*, r_j x_{xj})$ are computed as the product of $E(K_*, v_j r_j)$ and $E(K_*, x_{xj})$, and $E(H_*, r_j)$ and $E(H_*, x_{xj})$ respectively, and vote $E(\{K_*, H_*\}, v_j C_{Cj})$ is constructed as the product of $E(K_*, v_j r_j x_{xj})$ and $E(K_*, C_{Cj})$. As V_j asks Tallying managers to encrypt $v_j r_j$ instead of v_j while generating secret random number r_j, TM_1,---,TM_P cannot know v_j. Also encrypted $v_j r_j$ is further multiplied by encrypted x_{xj}, of which decrypted value is not known to anyone; therefore even V_j cannot identify its vote at the tallying stage. About the approval of votes, because no one except V_j knows $S(X_{(2)*}, T_{tj})$ even after $S(X_{(1)*}, T_{tj})$ had been disclosed, only V_j can approve its vote, consequently V_j cannot claim any dishonesty about its vote after its approval. Figure 5.4 depicts the steps of vote constructions; they proceed as follows:

1. V_j generates its secret random number r_j to calculate $v_j r_j$ and r_j^{Lj}, and asks TM_1, ---, TM_P to encrypt them into $E(K_*, v_j r_j)$ and $E(H_*, r_j^{Lj})$. By using these results, V_j calculates $E(\{K_*, H_*\}, v_j) = \{E(K_*, v_j r_j), E(H_*, r_j)\}$ as described in Sec 3.1.4.
2. V_j verifies the correctness of encryption of $E(\{K_*, H_*\}, v_j)$, and calculates $E(K_*, v_j r_j)E(K_*, x_{xj}) = E(K_*, v_j r_j x_{xj})$ and $E(H_*, r_j)E(H_*, x_{xj}) = E(H_*, r_j x_{xj})$. Then it multiplies $E(K_*, v_j r_j x_{xj})$ by its $E(K_*, C_{Cj})$, *i.e.* calculates $E(K_*, v_j r_j x_{xj})E(K_*, C_{Cj}) = E(K_*, v_j C_{Cj} r_j x_{xj})$, and constructs its vote as $E(\{K_*, H_*\}, v_j C_{Cj}) = \{E(K_*, v_j C_{Cj} r_j x_{xj}), E(H_*, r_j x_{xj})\}$.
3. V_j submits $E(\{K_*, H_*\}, v_j C_{Cj})$ and $E(K_*, C_{Cj})$ as its vote and puts them on the position corresponding to T_{tj} in *VotingPanel*.
4. TM_1, ---, TM_P repeatedly sign on $E(\{K_*, H_*\}, v_j C_{Cj})$ and $E(K_*, C_{Cj})$ in *VotingPanel* by the first form of their signatures *i.e.* calculate $S(\{M_{(1)*}, Q_{(1)*}\}, E(\{K_*, H_*\}, v_j C_{Cj}))$ and $S(M_{(1)*}, E(K_*, C_{Cj}))$.
5. After confirming the correctness of signatures on its vote in *VotingPanel*, V_j submits

$S(X_{(2)*}, T_{tj})$ to *VM* as its approval.

6. TM_1, ---, TM_P repeatedly sign on $E(\{K_*, H_*\}, v_jC_{Cj})$) by the second form of their signatures *i.e.* calculate $S(\{M_{(2)*}, Q_{(2)*}\}, E(\{K_*, H_*\}, v_jC_{Cj}))$. Finally V_j verifies the signatures.

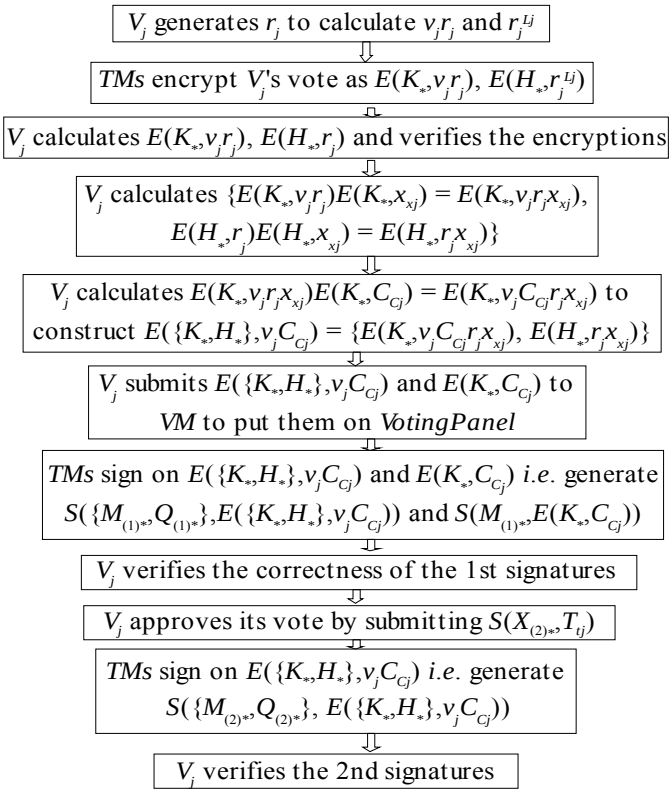


Fig. 5.4 Vote construction procedures

For this sub-stage security problems are as follows:

- *Voters may submit invalid votes to disrupt the voting*:

  V_j cannot claim that its vote is disrupted even its vote is meaningless when disclosed C_{Cj} is valid and signatures *i.e.* $S(\{M_{(1)*}, Q_{(1)*}\}, v_jC_{Cj})$ and $S(\{M_{(2)*}, Q_{(2)*}\}, v_jC_{Cj})\}$ are consistent.

- *Voting manager may not put vote or put incorrect vote on VotingPanel:*

  As $S(X_{(1)*}, T_{tj})$ is open to the public, V_j can repeatedly submit its vote before its approval, therefore *VM* cannot deny putting. If *VM* puts incorrect vote, V_j can disapprove it.

- *Someone may disrupt votes in VotingPanel:*

As *VotingPanel* is open to the public, no one can modify or delete votes without being detected.

- *Tallying managers may sign both forms of their signatures incorrectly:*
  Voter verifies the correctness of both forms of signatures of Tallying managers, therefore impossible.

5.4 Tallying stage

Objectives of this stage are to decrypt all encrypted votes in *VotingPanel* and to disclose the results on *TallyingPanel* while concealing links between voters and their votes. When the deadline of vote submission comes, mutually independent Tallying managers repeatedly perform decryptions and shuffles of votes by using their secret decryption keys to post the results on *TallyingPanel*, as shown in Fig. 5.5. In the figure, 3 Tallying managers TM_2, TM_1 and TM_3 execute decryptions and shuffles. In this example, multiple decryptions are executed in the order different from encryptions.

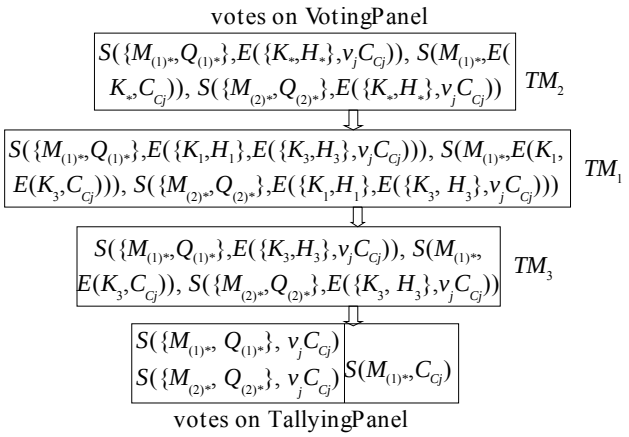


Fig. 5.5 Procedures in Tallying stage

For this stage security problems are as follows:

- *Tallying managers may change votes*:
  No one can generate two different forms of votes consistently unless all *TMs* conspire, and when votes are changed, responsible *TMs* are detected at the disruption detection stage based on this inconsistency. For example, although TM_i can forge $S(\{M_{(1)*}, Q_{(1)*}\}, v_kC_{Cj})$ from $S(\{M_{(1)*}, Q_{(1)*}\}, v_jC_{Cj})$, $S(M_{(1)*}, C_{Cj})$, $S(\{M_{(1)*}, Q_{(1)*}\}, v_kC_{Ck})$ and $S(M_{(1)*}, C_{Ck})$, and replace $S(\{M_{(1)*}, Q_{(1)*}\}, v_jC_{Cj})$ by it based on the homomorphic property, TM_i cannot forge $S(\{M_{(2)*}, Q_{(2)*}\}, v_kC_{Cj})$ consistently

because it does not know $S(M_{(2)*}, C_{Cj})$.

- *Tallying managers may add votes*:
  Anyone can detect the added votes by duplicated or by non registered *CNs*.

- *Tallying managers may delete votes*:
  By this the numbers of votes on *VotingPanel* and *TallyingPanel* become different which is detectable by anyone.

5.5 Disruption detection stage

$S(\{M_{(1)*}, Q_{(1)*}\}, v_2C_{18}), S(\{M_{(2)*}, Q_{(2)*}\}, v_2C_{18})$	$S(M_{(1)*}, C_{18})$	√
$S(\{M_{(1)*}, Q_{(1)*}\}, v_bC_2), S(\{M_{(2)*}, Q_{(2)*}\}, v_aC_2)$	$S(M_{(1)*}, C_2)$	x
$S(\{M_{(1)*}, Q_{(1)*}\}, v_hC_{-10}), S(\{M_{(2)*}, Q_{(2)*}\}, v_hC_{-10})$	$S(M_{(1)*}, C_{-10})$	x
$S(\{M_{(1)*}, Q_{(1)*}\}, v_bC_{25}), S(\{M_{(2)*}, Q_{(2)*}\}, v_bC_{25})$	$S(M_{(1)*}, C_{25})$	x
$S(\{M_{(1)*}, Q_{(1)*}\}, v_aC_{25}), S(\{M_{(2)*}, Q_{(2)*}\}, v_aC_{25})$	$S(M_{(1)*}, C_{25})$	x

"√" and "x" imply consistent and inconsistent votes, respectively

Fig. 5.6 Possible vote disruptions

If any inconsistency is found in *TallyingPanel*, Disruption detection manager *DM* detects liable entities. Figure 5.6 shows examples of consistent and inconsistent votes on *TallyingPanel*. The first vote (first row) is accepted because two different forms of vote v_2CN_{18} are same and also CN_{18} is registered. The second vote (second row) is not consistent because the candidates within the two signed forms are different. The third vote (third row) is inconsistent because CN_{-10} is not registered. The fourth and fifth votes (fourth and fifth rows) are also inconsistent because of duplicated *CNs*.

DM identifies the liable entities as follows. When an inconsistent vote $\underline{v_i}$ is found, *DM* asks *TMs* to encrypt $\underline{v_i}$ again in the reverse order of the tallying stage, namely each TM_i encrypts $\underline{v_i}$ and discloses the result with its input vote in the tallying stage that matches with $\underline{v_i}$. When this matching chain fails, the dishonest TM_i is found. Here TM_i cannot encrypt votes dishonestly because anyone can check the correctness of its encryption in the same way as in Sec. 3.1.4. Also when $\underline{v}$ had been submitted in the authorized way, dishonest managers are identified before the chain reaches *VotingPanel*. Therefore privacies of voters are maintained.

When *DM* detects the inconsistent votes in *VotingPanel*, the corresponding tokens are revealed, but as tokens are anonymous, voters are still anonymous. However, although this case does not occur as long as authorities are honest, coercers may know the links between inconsistent votes and their voters because they can know the tokens from the voters.

Chapter 6

Evaluation of the Scheme

This chapter evaluates security performance of the proposed scheme for satisfying the requirements of e-voting systems. Also the computation volume is discussed based on simulation result of prototype system.

6.1 Security analysis

The proposed scheme satisfies the following requirements of e-voting.

6.1.1 Eligibility

The eligibility of voters are checked by their *ID* and *P*/*W* pairs, and no one can forge the signatures on the tokens of voters as tokens possess the signatures of multiple mutually independent Tallying managers, therefore anyone cannot pretend authorized voters. Also as voters put their votes in the positions of *VotingPanel* corresponded to their tokens, multiple voting by a single voter is prevented.

6.1.2 Privacy

Voters submit their votes without disclosing their identities *i.e*. anonymously while showing their signed tokens, therefore no one except voters themselves can know votes of individual voters. No one including the authorities can know voters who did not submit their votes either, although they are already registered in *VoterList*, because voters submit their votes anonymously by using tokens.

6.1.3 Accuracy and universal verifiability

For obtaining tokens the eligibility of voters are checked by their *ID* and *P*/*W* pairs, and no one can forge signatures of multiple Tallying managers on their tokens, therefore any

unauthorized entity cannot put its vote. Also as voters put their votes in the positions of *VotingPanel* corresponded to their tokens, multiple voting is prevented. Moreover, uniqueness of registered *CNs* and signatures on votes and *CNs* ensure that all and only votes approved by individual voters are counted.

6.1.4 Fairness

No single authority or entity can decrypt the interim voting results *i.e*. encrypted votes submitted by voters because votes on *VotingPanel* are repeatedly encrypted by multiple Tallying managers.

6.1.5 Receipt-freeness

Voters know only their tokens, encrypted votes and encrypted *CNs*, and all of them cannot be linked to their votes in *TallyingPanel*. Also no one knows the correspondences between encrypted votes on *VotingPanel* and decrypted votes on *TallyingPanel* either, because of decryptions and shuffles of votes and *CNs* by multiple independent entities. Therefore the scheme is *receipt-free.*

6.1.6 Incoercibility

When decrypted votes in two different signed forms are equivalent, no one can claim that votes are disrupted, it means that the vote is meaningless from the beginning of its submission. Therefore coercers cannot invalidate elections by claiming vote disruptions while forcing voters to submit disrupted votes, and thereby the scheme is secure against *randomization attacks*. Although coercers can bias or influence voters to submit invalid votes with specific patterns through write-in ballots which will reduce effective votes if there are pre-specified candidates, this issue is out of the scope of this research and merits significant research. However at least it is not a so serious problem for this scheme as it is so serious in paper based schemes (see Sec. 2.5.1)

Although in *VoterList*, *IDs* of voters are open to the public, the abstention of registered voters cannot be identified because while vote submission, voters appear by using their anonymous signed tokens. Also receipt-freeness of the scheme disables coercers to identify voters who had put meaningless votes, therefore voters can abstain from elections

without being noticed by internal coercers *e.g*. authorities, therefore the scheme is free from *forced abstention attacks*.

Also, the uniqueness of signed tokens that enable registered voters to prove their eligibilities, disable coercers to submit votes on behalf of voters, and protects the scheme from *simulation attacks*.

6.1.7 Dispute-freeness

Publicly-verifiable data about interactions among entities on the *BBs*, signature pairs on votes and the disruption detection processes enable entities to resolve disputes.

6.1.8 Robustness

Voters can disrupt only their votes by submitting invalid *i.e*. meaningless votes. Either Voting manager or multiple Tallying managers cannot disrupt votes. Because the correctness of votes in *VotingPanel* is ensured by individual voters' approvals, and inconsistent votes and the liable entities are identified at the disruption detection stage. Also inconsistencies can be recovered by simply decrypting inconsistent votes again.

6.1.9 Scalability

CNs simplify the computational requirements of individual entities *e.g*. voters, Tallying managers etc. while maintaining the total accuracy and the incoercibility of the election as demonstrated in Sec 6.2.

6.1.10 Practicality

The scheme is based on weaker assumptions about trustworthiness regarding entities *i.e*. nothing can make the scheme unreliable unless multiple entities conspire. The scheme is consistent if at least one authority (Tallying manager) is honest among multiple authorities. The scheme does not assume any absolutely trustworthy authority.

6.2 Performance evaluation

A prototype system of the proposed scheme consists of three Tallying managers has been developed, and the computation times required for registration, voting and tallying are measured. Then the performances are compared with those of Scratch & Vote (S&V) [16] and Coercion-Resistant Voting (CRV) [31] which are available for comparisons. The environment consists of a 1.60 GHz CPU with 504 MBytes of RAM, and GMP [32] 1024 bit modulus running on Windows XP is used for encryptions. The time required for registering a voter is 0.0471 secs, for generating a vote is 0.308 secs, and for tallying a vote is 0.171 secs. Regarding the tallying, 1000 votes can be counted within 171 secs (*i.e.* 0.171 * 1,000 = 171), and this shows that the scheme is scalable and practical enough.

Table 6.1. Computation time required by the proposed scheme

Registration (m. secs)		Voting (m. secs)		Tallying (m. secs)	
Blinding	0.3	V_j's encryption	3.0	Decryption	133.0
Signing	45.0	*TMs* encryption	17.0	Verification	38.0
Unblinding	1.8	V_j's decryption	9.0		
		Verification	108.0		
		Signing	135.0		
		Verification	36.0		
Total	47.1	Total	308.0	Total	171.0

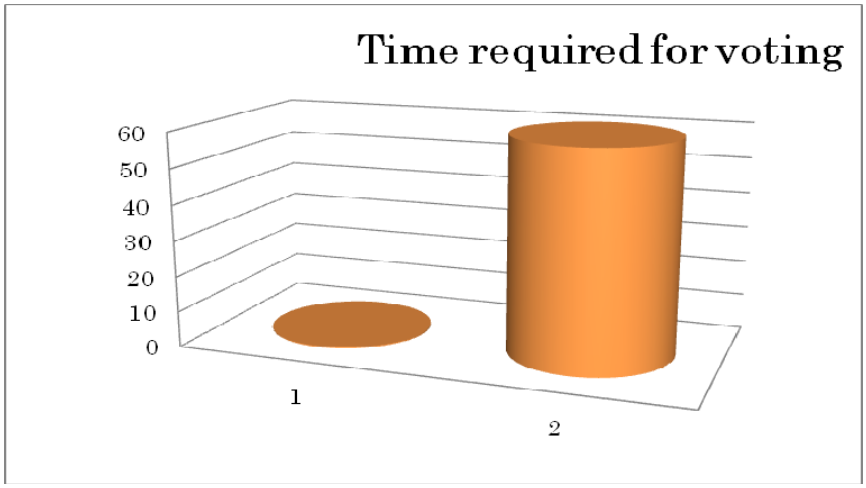
The registration of voter V_j is comprised of token T_{tj} blinding, signnature pair generations of three *TMs i.e.* generating six different signatures on blinded T_{tj}, and unblinding of six signed blinded T_{tj}. Blinding a token takes 0.3 m. secs, signature pair generations takes 45.0 m. secs and unblinding takes 1.8 m. secs. Here it is assumed that encrypted *CNs* and encrypted unknown random numbers are prepared in advance, therefore their computation time is not considered. The construction of vote v_j is comprised of the encryption of v_j by V_j itself, three *TMs*' triple encryptions of v_j, V_j's decryption of it, V_j's verification of *TMs*' encryptions of v_j and x_{xji}, *TMs*' repeatedly signing on encrypted vote and *CN* and V_j's verification of both forms of *TMs*' signatures. The time required for V_j's encryption of v_j is 3.0 m. secs, *TMs*' triple encryption is 17.0 m. secs, V_j's decryption is 9.0 m. secs, V_j's verification of *TMs*' encryptions of v_j and x_{xji} is 108.0 m. secs, *TMs*' repeatedly signing on encrypted vote and *CN* is 135.0 m. secs, and V_j's verification of both forms of

TMs' signatures is 36.0 m. secs. The time for tallying is comprised of decryptions and shuffles and verifications of two signed forms of votes and single signed form of *CNs*. Time required for decryptions and shuffles is 133.0 m. secs and verifications is 38.0 m. secs. Table 6.1 shows the computation time of each stage.

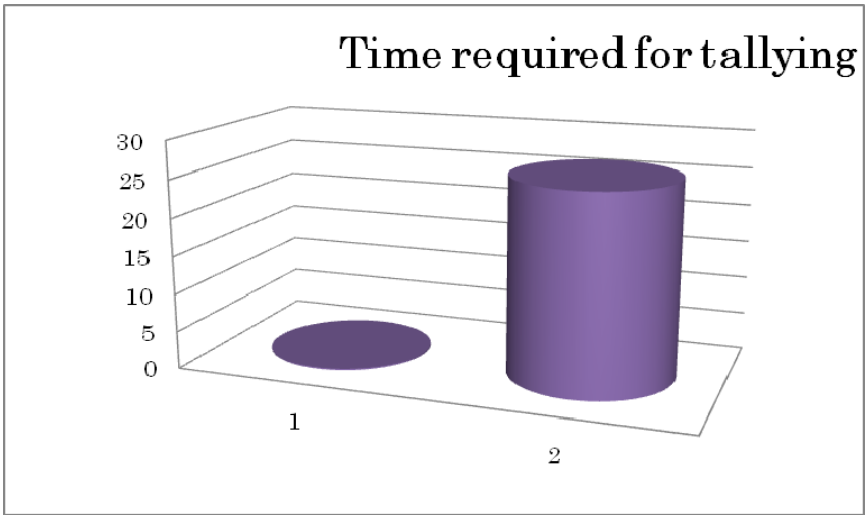
Table 6.2. Computation time comparisons with other schemes

	CPU	Registration	Voting	Tallying
Proposed scheme	1.6 GHz	0.0471 secs	0.308 secs	0.171 secs
CRV	2.0 GHz	-	-	26 ~ 62 secs
S&V	2.8 GHz	-	1 ~ 2 min	-

As Table 6.2 shows, compared with CRV that rely on ZKP, *CNs* of the proposed scheme substantially reduced the computation times *i.e*. the time required for the tallying is reduced at least more than 100 times. In the table all computation times of all schemes do not include the communication time. Figure 6.1 shows the voting and tallying stage computation time comparisons with other schemes where 1 refers the proposed scheme and 2 refers the other schemes *i.e*. in (a) 2 is S&V and in (b) 2 is CRV. Here CRV adopts threshold (*n*, *t*) ElGamal as the base encryption algorithm while using five and three as *n* and *t* values, where *n* is the total number of authorities and *t* is the threshold. The tallying process of CRV is comprised of verification of votes by NIZKPs, shuffling of verified votes, elimination of duplicated votes, shuffling of votes with unique credentials, shuffling of encrypted credentials of registered voters, collision detections of registered credentials, separations of votes with invalid credentials, decryptions and tallying.



(a)



(b)

Fig. 6.1 Computation time comparison of voting and tallying stages where 1 is for the proposed scheme. In (a) 2 is for S&V and in (b) 2 is for CRV.

Among the above operations all shufflings are carried out by verifiable mixnets each consists of multiple Tallying managers. Although the computation volumes of individual encryptions/decryptions and shuffles included in these mixnets are the same as those in the proposed scheme, *i.e*. they are propotional to key lengths, verifiable features of the mixnets supportrd by NIZKP make the whole computations complicated, when compared with the proposed scheme supported by *CNs*. Different from the proposed scheme, in which each TM_i executes multiplications corresponding to six decryptions for each vote, CRV requires huge number of multiplications for each vote to verify the correct behaviors of mixnets, *i.e*. to conduct each NIZKP process reliably usually about 80 times of challenges and responses are necessary each of which requires the same numbers of multiplications as encryptions and decryptions do. Also, the computation time required for tallying in the proposed scheme is strictly proportional to N, the number of voters, on the other hand that in CRV is the order of N^2 because it must eliminate duplicated votes, although it is suppressed to the linear order by using hashtables. Because voters carry out voting processes interactively, time required for voting is not so serious as tallying, therefore CRV did not mention the time of voting. Regarding the proposed scheme, 0.308 secs can be considered practical and scalable enough also. Moreover, many processes can be carried out in parallel by multiple managers if required. S&V is a paper based cryptographic voting system that offers entirely paper- and pen-based ballot casting, therefore the voting procedure is comparatively time consuming.

Chapter 7

Conclusions

6.1 Conclusions

The proposed e-voting scheme achieves verifiability by involving registered and unique confirmation numbers *CNs* while disabling all entities including voters themselves to know the linkages between voters and their votes. Therefore, the scheme can satisfy all the essential requirements of e-voting *e.g.* privacy, accuracy, universal verifiability, fairness, receipt-freeness, incoercibility, dispute-freeness, robustness, scalability and practicality. Most importantly, while being supported by *CNs*, these are achieved in a simple and efficient way, *e.g.* no extra proof is required at any stage of the election to prove its verifiability. Also a pair of signatures on encrypted votes proves the authenticity of votes and the honesty of election authorities even when the decryption of encrypted votes reveal a disrupted or meaningless results. Moreover, a pair of signature on blinded tokens enable voters to act anonymously *e.g.* the participation of voters to submit their encrypted votes and to approve the correct registration of their encrypted votes on *BB* can be achieved without disclosing their usual identities. As a conclusion, unlike existing e-voting schemes with complicated ZKP, the simplified computational requirements of individual election entities make the scheme practical and scalable.

6.2 Future works

The proposed e-voting scheme relies on the cryptosystem proposed in [22] in which all the encryption and decryption keys of all entities are secret. Also as discussed in Sec. 3.1.3, for encryptions of votes, unknown random numbers etc. and signing on votes, *CNs* etc., voters need to interact with authorities in order to confirm their correctness. It increases the time required and computations and communication overheads of the scheme. A further investigation can be made in making probabilistic public key cryptosystem *e.g.* ElGamal [25] or Paillier [26] available. Thereby voters do not need to interact with authorities, they

can encrypt or verify the correctness of signatures on their data directly.

Regarding trustworthiness, the assumption of the scheme is that “among multiple authorities, at least single authority is trustworthy.” The scheme can be enhanced when all trusworthy entities are excluded.

The proposed scheme must be evaluated in more realistic environments where multiple authorities are distributed over different places, and many voters are involved.

Bibliography

[1] K. Sampigethaya and R. Poovendran, “A framework and taxonomy for comparison of electronic voting schemes,” Elsevier Computers and Security, Vol. 25, pp. 137-153, 2006.

[2] A. Fujioka, T. Okamoto and K. Ohta, “A practical secret voting scheme for large scale elections,” AUSCRYPT ’92. LNCS, Vol. 718. Springer-Verlag, pp. 248–59, 1993.

[3] J. Wen-Shenq, L. Chin-Laung and L. Horng-Twu, “A verifiable multi-authority secret election allowing abstention from voting,” The Computer Journal, Vol. 45(6), pp. 672–82, 2002.

[4] D. Chaum, “Elections with unconditionally- secret ballots and disruption equivalent to breaking RSA”, Advances in Cryptology – Eurocrypt’88, LNCS 330, Springer-Verlag, pp. 177–182, 1988.

[5] J. Benaloh and D. Tuinstra, “Receipt-free secret-ballot elections,” Proceedings of 26th Symp. on Theory of Computing, pp. 544–553, 1994.

[6] M. Hirt and K. Sako, “Efficient Receipt-Free Voting Based on Homomorphic Encryption,” Proceedings of EUROCRYPT, LNCS, Vol. 1807, pp. 539-556. Springer, 2000.

[7] B. Lee, and K. Kim, “Receipt-free electronic voting scheme with a tamperresistant randomizer”, ICISC 2002, LNCS 2587, Springer-Verlag, pp. 389–406, 2002.

[8] B. Lee, C. Boyd, E. Dawson, K. Kim, J. Yang and S. Yoo, “Providing receipt-freeness in Mixnet-based voting protocols,” Proceedings of the ICISC ’03, pp. 261–74, 2003.

[9] A. Neff, “A verifiable secret shuffle and its application to E-voting”, *ACM CCS 2001*, ACM Press, pp. 116–125, 2001.

[10] P. Golle, S. Zhong, D. Boneh, M. Jakobsson, and A. Juels, “Optimistic mixing for exit-polls”, Asiacrypt2002, LNCS 2501, Springer-Verlag, pp. 451–465, 2002.

Bibliography

[11] J. Schweisgut, “Coercion-resistant electronic elections with observer,” 2nd International Workshop on Electronic Voting, Bregenz, August 2006.

[12] A. Juels and M. Jakobsson, “Coercion-resistant electronic elections,” Cryptology ePrint Archive, Report 2002/165, <http://eprint.iacr.org/>; 2002.

[13] A. Acquisti, “Receipt-free homomorphic elections and write-in voter verified ballots,”. Cryptology ePrint Archive, Report 2004/105, <http://eprint.iacr.org/>; 2004.

[14] D. Chaum, “Secret-ballot receipts: true voter-verifiable elections,” IEEE Security & Privacy Magazine, Feb 2004.

[15] B. Riva and A. Ta-Shma, “Bare-Handed Electronic Voting with Pre-processing,” Proceedings of the USENIX/Accurate Electronic Voting Technology Workshop, Boston, MA, 2007.

[16] B. Adida and R. Rivest, “Scratch and vote: Self-Contained paper-based cryptographic voting,” Workshop on Privacy in Electronic Society 2006.

[17] A. K. Md. Rokibul and S. Tamura, “Electronic Voting Using Confirmation Numbers,” Proceedings of the 2009 IEEE International Conference on Systems, Man and Cybernetics, San Antonio, TX, USA, pp. 4672–77, 2009.

[18] Taisya Krivoruchko “Robust Coercion-Resistant Registration for Remote E-voting” 7th IAVoSS Workshop On Trustworthy Elections (WOTE 2007), University of Ottawa, Ottawa, Canada, June 20-22, 2007.

[19] Kiayias Aggelos and Yung Moti. “Self-tallying elections and perfect ballot secrecy,” Proceedings of public key cryptography, 5th international workshop on practice and theory in public key cryptosystems, LNCS, vol. 2274. Springer-Verlag, pp.141–58, 2002.

[20] S. Tamura and T. Yanase, “Information Sharing among untrustworthy entities”, IEEJ Trans. EIS, Vol.125, No.11, pp.1767-1772 (2005) .

Bibliography

[21] Ben Adida and Douglas Wikstrom, "How To Shuffle in Public", In Theory of Cryptography, LNCS, vol 4392, Springer Berlin, pages 555-574 (2007).

[22] S. Tamura, A. K Md. Rokibul and H. A. Haddad, "A Probabilistic and Commutative Re-Encryption Scheme," published in Asia Simulation Conference 2009, ID032, Ritsumeikan University, Shiga, Japan, October 7-9, 2009.

[23] D. Chaum, A. Fiat and M. Naor, "Untraceable electronic cash," in CRYPTO 88, Springer-Verlag, pp. 319 –327, 1988.

[24] R. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," Communications of the ACM, Vol 21, no. 2, pp. 120–126, 1978.

[25] Taher ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Transactions on Information Theory, vol. IT-31, no. 4, pp. 469-472 or CRYPTO, vol. 84, pp. 10-18, 1985.

[26] P. Paillier, "Public-key cryptosystems based on composite degree residue classes," in Advances in cryptology – EUROCRYPT '99. LNCS, Vol. 1592, pp. 223–38, 1999.

[27] E. Mohammed, A. E. Emarah, and K. El-Shennawy, "A blind signatures scheme based on ElGamal signature," in IEEE/AFCEA EUROCOMM 2000 Information Systems for Enhanced Public Safety and Security, pp. 51–53, 2000.

[28] D. Sandler, K. Derr and D. S. Wallach, "VoteBox: a tamper-evident, verifiable electronic voting system," in *p*roceedings of the 17th USENIX Security symposium, pp 349-364, July 28- August 1, San Jose, California, 2008.

[29] K.-P. Yee, "Extending prerendered-interface voting software to support accessibility and other ballot features," in Proceedings of the 2nd USENIX/ACCURATE Electronic Voting Technology Workshop (EVT'07), Boston, MA, Aug. 2007.

Bibliography

[30] K.-P. Yee, D. Wagner, M. Hearst, and S. M. Bellovin, "Prerendered user interfaces for higher-assurance electronic voting," in USENIX/ACCURATE Electronic Voting Technology Workshop, Vancouver, B.C., Canada, 2006.

[31] S. Weber, "A Coercion-Resistant Cryptographic Voting Protocol -Evaluation and Prototype Implementation," Diploma thesis, Darmstadt University of Technology; 2006.

[32] T. Granlund. GNU Multiple Precision Arithmetic Library (GMP). Software available at http://gmplib.org/ April 2009.

[33] Yehuda Lindell, Anna Lysyanskaya, and Tal Rabin, "On the composition of authenticated byzantine agreement," in STOC, pp. 514–523, ACM, 2002.