Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Electronic Voting Using Confirmation Numbers

Kazi Md. Rokibul Alam
Graduate School of Engineering
University of Fukui, Fukui, Japan
h9rokibu@radio.fuis.fukui-u.ac.jp

Shinsuke Tamura
Graduate School of Engineering
University of Fukui, Fukui, Japan
tamura@fuis.fuis.fukui-u.ac.jp

***Abstract*—This paper proposes a new electronic voting (e-voting) scheme that fulfills all the security requirements of e-voting. 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. Unlike complicated zero knowledge proof involved in many e-voting schemes, the confirmation numbers attain the verifiability requirement in a much more simple and intuitive way, then the scheme becomes scalable and practical.**

***Keywords*—encryption/decryption shuffle, incoercibility, public verifiability, confirmation number, signature pairs.**

I. INTRODUCTION

Electronic voting (e-voting) enables efficient and secure elections. Also the reusable resources of e-voting scheme make elections inexpensive. Moreover it does not require any geographical proximity of voters, and it provides better scalability for large elections [1]. However e-voting scheme has potential problems that may degrade its acceptance. Namely, simple vote verification mechanisms enable entities to link voters and their votes, and therefore coercers can force voters to follow their intentions more easily. On the other hand, complicated mechanisms that achieve the anonymity of voters while maintaining verifiability of their votes make e-voting systems non-scalable and non-practical.

To overcome these difficulties, this paper proposes a new e-voting scheme that satisfies all the requirements of e-voting *i.e.* privacy, accuracy, universal verifiability, fairness, receipt-freeness, incoercibility, dispute-freeness, robustness, scalability and practicality, usually some of which are found as traded. This scheme is based on weaker assumptions about trust-worthiness of entities, *i.e.* nothing can corrupt the scheme if at least one entity is honest, and the way of candidate selections is flexible; it accepts freely chosen write-in ballots, votes for pre-specified or t out of l choices as well as yes/no votes.

II. MECHANISMS EXPLOITED IN THE SCHEME

The proposed scheme exploits the following mechanisms.

1. *Bulletin Board* (*BB*)*:* a *BB* is a public broadcast channel with memories. Information sent to a *BB* is readable by anyone and at anytime. Therefore by putting relevant information on several *BB*s, interactions among the entities at every stage of election become publicly verifiable.

2. *Mixnet Like Encryption and Decryption Shuffles:* votes and confirmation numbers (*CNs*) described below are repeatedly encrypted or decrypted by multiple independent entities while interim results are being shuffled. Here individual entities do not know encryption mechanisms of other entities, therefore, no entity can identify links between votes or *CNs* and their encrypted forms unless the multiple entities conspire.

3. *Confirmation Numbers* (*CNs*)*: CNs* are unique numbers in the system that are attached to voters, and voter V_j constructs its vote while attaching confirmation number CN_j assigned to it. Therefore, any entity can confirm that votes are valid, when the votes disclosed in *BB* include different and authorized *CNs*. Nevertheless the privacy of voters are maintained because encryption and decryption shuffles of *CN*s disable entities including voters themselves to know their assigned *CN*s.

4. *Signature Pairs:* they are pairs of signatures of signing authorities, and enable entities to protect them from voters' dishonesties. Namely voters cannot claim that their votes are disrupted by other entities while intentionally submitting invalid votes, when votes with 2 different signatures are equivalent, because entities that do not know the signing keys cannot modify votes with 2 different signatures consistently.

Here among of the above mechanisms, *CN*s play the most important roles. The major approaches of current e-voting schemes are mixnet and homomorphic encryption based [4], and they exploit zero knowledge proof (ZKP) extensively to achieve verifiability, however ZKP is complicated and expensive. Whereas *CNs* make votes verifiable in a simple way. Namely, the proposed scheme does not require any extra proof of votes. From the disclosed data on *BBs* anyone can verify the votes without knowing the links between voters and their votes.

III. RELATED WORKS AND CONTRIBUTIONS

Ideal e-voting schemes satisfy privacy, accuracy, universal verifiability, fairness, receipt-freeness, incoercibility, dispute-freeness, robustness, scalability, practicality etc. However, satisfying these requirements altogether at the same time is really difficult because there are tradeoffs among them, and many existing e-voting schemes cannot satisfy all requirements.

Several voting schemes [2, 3, 4] achieve receipt-freeness by attaching secret random numbers to votes while proving the correctness of votes by using interactive ZKP (IZKP) or non-interactive ZKP (NIZKP) that makes the schemes impractical. Also untappable channels used in them sacrifice practicality

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

and scalability [3, 4], and they cannot achieve the complete receipt-freeness. Namely, authorities can know the random numbers *i.e.* the links between voters and their votes. Although a scheme that uses tamper-resistant randomizer (TRR) [2], a hardware device to generate random numbers for voters, achieves the complete anonymity of voters, TRR and NIZKP that proves the correctness of votes, sacrifice practicality.

In several incoercible voting schemes [5, 6, 7], voters obtain unique tokens and construct encrypted votes while combining with the encrypted tokens, to submit multiple votes without being traced by others. 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. A scheme proposed in [5] employs an observer that simplifies the time consuming verification processes, but still involves ZKP to disable voters to transfer encrypted forms of their tokens to others, and NIZKP to prove the correctness of votes.

Although e-voting schemes based on blind signature do not exploit ZKP, usually these schemes cannot satisfy universal verifiability or receipt-freeness because voters' blind factors can be used as receipts of their votes, and therefore voters can prove their votes to buyers. Besides, these schemes assume the existence of anonymous channels which are impractical [1].

The proposed scheme uses *CNs* that make votes verifiable while maintaining unlinkability between voters and votes. *CNs* also ensure that all votes from eligible voters are counted, and thereby maintain the total accuracy of the election. A mechanism for *CNs* is simple enough. It requires much less computations for individual entities. Although several schemes [5, 6, 7] had already used unique numbers (tokens) to make votes verifiable, they only prove the correctness of individual votes, do not ensue that all votes from eligible voters are counted. Moreover, these schemes require trusted entities that know the tokens; therefore they cannot satisfy complete privacy or incoercibility.

IV. Configuration of the Voting Scheme

This section presents the overview of the proposed voting scheme with descriptions of several newly developed security components.

A. Security Components

1) Confirmation Numbers (CNs): CNs assigned to individual voters make votes verifiable. To conceal the content of CN_j (CN assigned to voter V_j) from any entity including V_j itself, Voting manager VM prepares N different encrypted *CNs* for N voters in advance. First, VM generates N unique numbers as shown in Fig. 1 (a). Then P (at least 2) mutually independent Tallying managers TM_1,---,TM_P repeatedly perform encryption and shuffles of all *CNs* by using their encryption keys, *i.e.* firstly TM_1 encrypts CN_j to CN_j' to be placed in random positions as shown in Fig. 1 (b). Then TM_2, TM_3,--- execute the same operations repeatedly, *i.e.* CN_j is converted to CN_j', CN_j'', CN_j''',--- as shown in Fig. 1 (b), (c) and (d). Here $CN_j' = E_{T1}(CN_j)$, $CN_j'' = E_{T2}(CN_j')$, $CN_j''' = E_{T3}(CN_j'')$,---, and $E_{Ti}(x)$ represents that x is encrypted to $E_{Ti}(x)$ by using the encryption key of TM_i. In the followings repeatedly encrypted CN_j is denoted as $E_{T*}(CN_j)$, *i.e.* $E_{T*}(CN_j) = E_{TP}(E_{T(P\text{-}1)}(\text{---}E_{T1}(CN_j)\text{---}))$. Therefore, no entity can know the linkages between original CN_j and $E_{T*}(CN_j)$ unless all *TMs* conspire. This multiple encryption is carried out based on the probabilistic and commutative re-encryption scheme described later in this subsection.

1	2	...	CN_j	...	N

(a) a set of N unique numbers

i_1	CN_j'	...	i_j	...	i_N

(b) 1st encryption and shuffle

k_1	...	k_j	...	CN_j''	k_N

(c) 2nd encryption and shuffle

m_1	CN_j'''	...	m_j	...	m_N

(d) 3rd encryption and shuffle

Figure 1. Encryption steps of confirmation numbers

2) Signature pairs on encrypted votes: Signature pairs on encrypted votes prove the honesty of all election managers, *i.e.* when managers are honest they can disable anyone to blame them for vote disruptions. In the scheme voter V_j puts its vote v_j on the *BB* while encrypting it into $\underline{E}_{T*}(v_jCN_j)$ with encrypted CN_j *i.e.* $E_{T*}(CN_j)$, and Tallying managers TM_1,---,TM_P repeatedly sign on $\underline{E}_{T*}(v_jCN_j)$ by a pair of their signatures *i.e.* generate $Sig_{T1P}(\text{---}Sig_{T12}(Sig_{T11}(\underline{E}_{T*}(v_jCN_j)))\text{---})$ and $Sig_{T2P}(\text{---}Sig_{T22}(Sig_{T21}(\underline{E}_{T*}(v_jCN_j)))\text{---})$ respectively, and sign on $E_{T*}(CN_j)$ by the first form of their signatures *i.e.* generate $Sig_{T1P}(\text{---}Sig_{T12}(Sig_{T11}(E_{T*}(CN_j)))\text{---})$ to put them on the *BB*. Therefore no single entity can forge two different signed forms consistently and no entity can claim that votes are disrupted by managers, when decrypted votes in two different signed forms on the *BB* are equivalent. In the remainder two forms of repeatedly signed encrypted vote v_j are denoted as $Sig_{T1*}(\underline{E}_{T*}(v_jCN_j))$ and $Sig_{T2*}(\underline{E}_{T*}(v_jCN_j))$, and two forms of signed decrypted vote v_j are denoted as $Sig_{T1*}(v_jCN_j)$ and $Sig_{T2*}(v_jCN_j)$. Also notations $Sig_{T1*}(E_{T*}(CN_j))$ and $Sig_{T1*}(CN_j)$ are used in the same way. These signatures are also generated based on the probabilistic and commutative re-encryption scheme.

3) Signature pairs on blinded tokens: Voters can act without disclosing their identities while showing their eligibility by using tokens. Namely, voter V_j blinds its token T_j as $E_{Kj}(T_j)$ by using V_j's secret key, and while confirming identities of voters by usual means, Tallying managers TM_1,---,TM_P blindly sign on $E_{Kj}(T_j)$ [10] by a pair of their signatures in two different forms *i.e.* $\{Sig_{T1}(E_{Kj}(T_j))$,---,$Sig_{TP}(E_{Kj}(T_j))\} = Sig_{T*}(E_{Kj}(T_j))$ and $\{Sig_{T1}'(E_{Kj}(T_j))$,---,$Sig_{TP}'(E_{Kj}(T_j))\} = Sig_{T*}'(E_{Kj}(T_j))$. Then V_j decrypts them into $Sig_{T*}(T_j)$ and $Sig_{T*}'(T_j)$, and can show its eligibility by showing $Sig_{T*}(T_j)$, because T_j is unique and only eligible V_j can get signatures on T_j. Also only V_j can approve the registration of its vote anonymously while proving its eligibility by disclosing $Sig_{T*}'(T_j)$, because only V_j knows $Sig_{T*}'(T_j)$ even after $Sig_{T*}(T_j)$ has been opened.

4) Probabilistic and commutative re-encryptions: A multiple encryption and signature scheme for votes and *CNs* described in 1) and 2) in this subsection can be implemented based on the probabilistic encryption with homomorphic and commutative properties, proposed in [9]. While selecting appropriate 2 large integers p_1 and p_2, it exploits 2 encryption and decryption key pairs of each Tallying manager TM_i and the key pairs are kept as TM_i's secrets, to enable each TM_i to securely use its keys under the same modulo arithmetic shared with other entities. Each V_j also has 2 secret encryption and decryption key pairs to interact with *TMs* to encrypt its vote.

Regarding *CNs*, probabilistic encryption is not necessary, because all *CNs* are unique and their encrypted forms are different. Whereas different voters may submit the same votes, therefore the encrypted form of vote v_j must be probabilistic. This can be constructed as a pair of the vote part $E_{T*}(v_j r_j)$ (mod p_1 arithmetic) and the secret random number part $E_{\underline{T*}}(r_j)$ (mod p_2 arithmetic), thereby v_j becomes $E_{T*\underline{T*}}(v_j) = \{E_{T*}(v_j r_j), E_{\underline{T*}}(r_j)\}$ ($E_{T*\underline{T*}}(v_j)$ can be decrypted by decrypting $E_{T*}(v_j r_j)$ and $E_{\underline{T*}}(r_j)$ and dividing $v_j r_j$ by r_j). Moreover, to hide r_j from V_j itself, they are multiplied by unknown encrypted random numbers $\{E_{T*}(x_j), E_{\underline{T*}}(x_j)\}$. Therefore the final vote $\{\underline{E}_{T*}(v_j CN_j), E_{T*}(CN_j)\}$ is constructed by multiplying $E_{T*}(v_j r_j)$, $E_{T*}(x_j)$ and $E_{T*}(CN_j)$, and $E_{\underline{T*}}(r_j)$, $E_{\underline{T*}}(x_j)$ respectively, based on homomorphic property. Here although encryption keys of all TM_i are secret, V_j can confirm the correct encryptions as same as it is using public keys by asking *TMs* to decrypt test data. In the followings the notation $\underline{E}_{T*}(v_j CN_j)$ is used to represent $\{E_{T*}(v_j CN_j r_j x_j), E_{\underline{T}*}(r_j x_j)\}$.

A signing mechanism on re-encrypted forms can be implemented by only adding signing and verification key pairs of each TM_i to the above re-encryption mechanism. However, verification keys are disclosed after all votes in the *BB* are decrypted to protect signing keys. Because data on the *BB* are readable by anyone at any time, no one can forge signatures on votes in *BB* after the tallying starts, even signing keys are calculated from the disclosed verification keys.

B. Entities and Their Roles

Entities involved in the scheme are *N* voters V_j (j = 1,---, N), Voting manager *VM*, *P* (at least 2) mutually independent Tallying managers TM_i (i = 1,---,P), Disruption detection manager *DM* and 6 public *BB*s that maintain authenticated communication transcripts *i.e.* *VoterList*, *TokenList*, *ConfNoList*, *ActiveTokenList*, *VotingPanel* and *TallyingPanel*. Fig. 2 depicts the configurations of individual *BBs*. The roles of entities are as follows:

Voter V_j: Each V_j generates its encrypted vote while combining it with its assigned CN_j *i.e.* $E_{T*}(CN_j)$ 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 three secret encryption and decryption key pairs $\{K_j, K_j^{-1}\}$, $\{EVj, DVj\}$ and $\{\underline{EVj}, \underline{DVj}\}$. ID_j and P/W_j pair proves the eligibility of V_j. Key pair $\{K_j, K_j^{-1}\}$ is used to acquire two different forms of signatures of all *TMs* on its token T_j blindly *i.e.* $Sig_{T*}(T_j)$ and $Sig_{T*}'(T_j)$, and key pairs $\{EVj, DVj\}$ and $\{\underline{EVj}, \underline{DVj}\}$ are used to ask *TMs* to encrypt vote v_j into $\underline{E}_{T*}(v_j CN_j)$.

Voting manager VM: VM is responsible to authenticate voters, to assign *CNs* to voters, and to put data about voters and votes in the corresponding *BBs*.

Tallying managers TMs: Mutually independent *P TMs* sign on blinded tokens, perform encryption shuffles of *CNs*, repeatedly sign on encrypted votes and encrypted *CNs* in *VotingPanel*, and perform decryption shuffles of votes in *VotingPanel* to compute the tally and put results on *TallyingPanel*. For encryption and decryption of votes and *CNs*, each TM_i has 2 encryption and decryption key pairs $\{ETi, DTi\}$ and $\{\underline{ETi}, \underline{DTi}\}$. Also to sign on $E_{Kj}(T_j)$, TM_i has 2 signing and verification key pairs *i.e.* $\{T_i^{-1}, T_i\}$ and $\{T_i'^{-1}, T_i'\}$, and to repeatedly sign on encrypted votes and encrypted *CNs*, TM_i has 4 secret signing and verification key pairs $\{\{T_{1i}^{-1}, T_{1i}\}, \{\underline{T_{1i}}^{-1}, \underline{T_{1i}}\}\}$ and $\{\{T_{2i}^{-1}, T_{2i}\}, \{\underline{T_{2i}}^{-1}, \underline{T_{2i}}\}\}$.

Disruption detection manager DM: DM detects inconsistent votes in *TallyingPanel* and the liable entities that cause them.

ID_1	$E_{Kj}(T_j)$
ID_2	
...	$E_{Kj}(T_9)$
ID_j	
...	$E_{Kj}(T_6)$
. . .	
ID_N	$E_{Kj}(T_1)$

(a) VoterList

T_1	√
T_{77}	
T_{21}	√
. . .	
T_i	
. . .	√
T_N	

(b) TokenList

$E_{T*}(CN_1)$	$E_{T*}(x_1)$, $E_{\underline{T*}}(x_1)$
. . .	
$E_{T*}(CN_j)$	$E_{T*}(x_9)$, $E_{\underline{T*}}(x_9)$
. . .	
$E_{T*}(CN_N)$	$E_{T*}(x_{11})$,$E_{\underline{T*}}(x_{11})$
. . .	
$E_{T*}(CN_{11})$	$E_{T*}(x_N)$,$E_{\underline{T*}}(x_N)$

(c) ConfNoList

$Sig_{T*}(T_j)$	$E_{T*}(CN_{j-1})$
	. . .
$Sig_{T*}(T_9)$	$E_{T*}(CN_1)$
	. . .
$Sig_{T*}(T_6)$	$E_{T*}(CN_{j+1})$
	. . .
$Sig_{T*}(T_1)$	$E_{T*}(CN_N)$

(d) ActiveTokenList

$Sig_{T1*}(\underline{E}_{T*}(v_j CN_{j-1}))$, $Sig_{T1*}(E_{T*}(CN_{j-1}))$, $Sig_{T2*}(\underline{E}_{T*}(v_j CN_{j-1}))$	$Sig_{T*}'(T_j)$
...	
$Sig_{T1*}(\underline{E}_{T*}(v_j CN_1))$, $Sig_{T1*}(E_{T*}(CN_1))$, $Sig_{T2*}(\underline{E}_{T*}(v_j CN_1))$	$Sig_{T*}'(T_9)$
. . .	
$Sig_{T1*}(\underline{E}_{T*}(v_j CN_{j+1}))$, $Sig_{T1*}(E_{T*}(CN_{j+1}))$, $Sig_{T2*}(\underline{E}_{T*}(v_j CN_{j+1}))$	$Sig_{T*}'(T_6)$
...	
$Sig_{T1*}(\underline{E}_T^*(v_j CN_N))$,$Sig_{T1*}(E_T^*(CN_N))$, $Sig_{T2*}(\underline{E}_T^*(v_j CN_N))$	$Sig_{T*}'(T_1)$

(e) VotingPanel

$Sig_{T1*}(v_j CN_N)$ $Sig_{T2*}(v_j CN_N)$	$Sig_{T1*}(CN_N)$
$Sig_{T1*}(v_j CN_1)$ $Sig_{T2*}(v_j CN_1)$	$Sig_{T1*}(CN_1)$
$Sig_{T1*}(v_j CN_{26})$ $Sig_{T2*}(v_j CN_{26})$	$Sig_{T1*}(CN_{26})$
. . .	
$Sig_{T1*}(v_j CN_V)$ $Sig_{T2*}(v_j'' CN_V)$	$Sig_{T1*}(CN_V)$
$Sig_{T1*}(v_j CN_{N+1})$ $Sig_{T2*}(v_j CN_{N+1})$	$Sig_{T1*}(CN_{N+1})$

(f) TallyingPanel

Figure 2. Configurations of bulletin boards.

VoterList: VoterList enables audiences including voters to know eligible voters and voters who have registered. It consists of *ID* and token parts. *ID* part maintains *IDs* of voters, and when V_j registers itself, *VM* puts V_j's blind token *i.e.* $E_{Kj}(T_j)$ in token part corresponding to ID_j as shown in Fig. 2 (a).

TokenList: It consists of the token and used flag parts, and enables voters to acquire tokens without collision. The token part maintains unique tokens prepared by *VM*. When V_j picks T_j from *TokenList* anonymously, *VM* signs on T_j (it is different from $Sig_{T*}(T_j)$ and $Sig_{T*}'(T_j)$, and ensures that T_j is picked from

TokenList) and gives it to V_j while putting a check mark in used flag part corresponding to T_j as shown in Fig. 2 (b).

ConfNoList: It consists of *CN* and random number parts, and for *N* voters, *N* different CN_j and unknown random number x_j are generated and encrypted to $E_{T^*}(CN_j)$ and $\{E_{T^*}(x_j), \underline{E}_{T^*}(x_j)\}$ to be posted here at random by *VM* as shown in Fig. 2 (c).

ActiveTokenList: It consists of the token and *CN* parts, and enables anyone to know anonymous V_j who had been assigned $E_{T^*}(CN_j)$. The token part maintains the first signed form of T_j *i.e.* $Sig_{T^*}(T_j)$ of V_j who had acquired CN_j. The *CN* part maintains an encrypted CN_j assigned to V_j *i.e.* $E_{T^*}(CN_j)$ as shown in Fig. 2 (d). By comparing the number of items in *ActiveTokenList*, *ConfNoList* and *VoterList*, anyone can verify that only registered voters acquire *CNs* and *VM* is not misusing any *CN*.

VotingPanel: It consists of the vote and the approval parts, and enables anyone to know encrypted votes approved by their voters. The vote part corresponding to anonymous voter V_j (*i.e.* T_j) maintains encrypted v_jCN_j in two different forms *i.e.* $Sig_{T1^*}(\underline{E}_{T^*}(v_jCN_j))$ and $Sig_{T2^*}(\underline{E}_{T^*}(v_jCN_j))$, and $E_{T^*}(CN_j)$ in a single form *i.e.* $Sig_{T1^*}(E_{T^*}(CN_j))$, and the approval part maintains the second form of signed T_j *i.e.* $Sig_{T^*}'(T_j)$ as shown in Fig. 2 (e).

TallyingPanel: It 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 holds $\{Sig_{T1^*}(v_jCN_j), Sig_{T2^*}(v_jCN_j)\}$ and the *CN* part holds $Sig_{T1^*}(CN_j)$ as shown in Fig. 2 (f). Based on *CNs*, anyone can verify that only and all votes from eligible voters are included in *TallyingPanel*, but no one can identify linkages between voters and their votes.

V. INDIVIDUAL STAGES OF THE SCHEME

The scheme consists of 5 stages and proceeds as follows.

A. Token Acquisition Stage

An objective of this stage is to assign voter V_j a token T_j which is unique in the system while maintaining anonymity of V_j. For this, anonymously authenticated V_j picks T_j from *TokenList*. To enforce V_j to pick T_j from *TokenList*, every T_j has the signature of *VM*. However tokens in *TokenList* are open to the public only in non-signed forms to disable entities to use them in unauthorized ways. Theoretically, V_j authentication is not necessary for this step. But by disabling unauthorized entities to pick tokens, it becomes possible to make the *TokenList* as small as possible. V_j and *VM* interact as follows:

1. *VM* authenticates V_j anonymously through anonymous authentication mechanism *e.g.* [8].

2. Authenticated V_j picks its unused token T_j from *TokenList*. Here the signature of *VM* on T_j ensures that the token is picked from *TokenList*.

3. In order to avoid collision, *VM* puts a check mark on its T_j in *TokenList* as shown in Fig. 2 (b).

Security problems of this stage are solved as bellow:

- *Voters may get multiple tokens*: As only tokens with signatures of *TMs*, which are given at the registration stage while confirming the eligibility of 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 T_j assignment repeatedly.

B. Registration Stage

Objectives of this stage are: (1) to let Tallying managers sign on token T_j that is shown by eligible voter V_j without knowing T_j itself [10], and (2) to enable all entities to know V_j who is assigned signed T_j. To make voters that obtain signed tokens publicly visible, *VM* maintains *VoterList*, as shown in Fig. 2 (a), and at this stage *VM* puts $E_{Kj}(T_j)$ in the position of *VoterList* corresponding to V_j. Therefore anyone can monitor V_j who is registered, however only V_j knows its token T_j. As a consequence, V_j can abstain from vote submission without being noticed even it is registered in *VoterList* for example. The interactions between V_j and *VM* in this stage are as follows:

1. V_j encrypts T_j by using its secret encryption key K_j *i.e.* V_j calculates $E_{Kj}(T_j)$.

2. V_j submits its ID_j, P/W_j and $E_{Kj}(T_j)$ to *VM*.

3. *VM* authenticates V_j and post $E_{Kj}(T_j)$ in *VoterList* so that anyone can know the registered V_j. *VM* also sends $E_{Kj}(T_j)$ to Tallying managers for their signatures.

4. Mutually independent TM_1,---,TM_P sign on $E_{Kj}(T_j)$ with their two different signatures *i.e.* calculate $Sig_{T^*}(E_{Kj}(T_j))$ and $Sig_{T^*}'(E_{Kj}(T_j))$ and send them to *VM* to be sent to V_j.

5. V_j checks the validity of signatures on T_j.

Here the 3rd step ensures that ineligible V_j cannot obtain signed T_j and eligible V_j cannot get multiple signed tokens. Also because of the 4th step, anyone cannot forge signed tokens unless all *TMs* conspire. Security problems of this stage are as follows:

- *Multiple entities request signatures on T_j picked by V_j:* By this threat, V_j's vote will be rejected. There are 2 possibilities, the first one occurs when T_j is stolen, however V_j is responsible for that. The other possibility is a case where *VM* uses T_j. This possibility can be excluded, if necessary, by duplicating *VM*, *i.e.* no entity can obtain signatures of all *TMs* on T_j in unauthorized ways unless all *VMs* conspire.
- *Voters cannot get correct signed tokens:* V_j can prove *VM*'s dishonesty by showing $E_{Kj}(T_j)$ and $Sig(E_{Kj}(T_j))$.

C. Voting Stage

This stage consists of two sub-stages, which are: i) *CN* assignment and ii) Vote submission.

1) CN assignment sub-stage: In this sub-stage: (1) Voting manager *VM* authenticates voter V_j anonymously by signed token $Sig_{T^*}(T_j)$, and (2) V_j receives encrypted CN_j *i.e.* $E_{T^*}(CN_j)$. While *VM* sends $E_{T^*}(CN_j)$ to V_j, it also discloses $E_{T^*}(CN_j)$ and

$Sig_{T*}(T_j)$ in *ActiveTokenList*. Here, anyone even V_j itself cannot identify the correspondance between original CN_j and $E_{T*}(CN_j)$, and hence between V_j and CN_j. However, because *CNs* are unique and no one can forge signatures of all Tallying managers on them, any entity can confirm the accuracy of votes by *CNs* disclosed in *TallyingPanel*. The interactions between V_j and *VM* in this sub-stage are as follows:

1. V_j submits $Sig_{T*}(T_j)$ to *VM*. Then *VM* checks the validity of $Sig_{T*}(T_j)$. Here *VM* can verify the authenticity of V_j by checking only the signatures on T_j that is not used repeatedly.

2. *VM* sends $E_{T*}(CN_j)$ to V_j. *VM* also puts $Sig_{T*}(T_j)$ and $E_{T*}(CN_j)$ pair in *ActiveTokenList* as shown in Fig. 2 (d).

Security problems of this sub-stage are as follows:

- *VM may put signed tokens in ActiveTokenList before voters*: *VM* knows neither of V_j's secret key and the signing keys of *TMs*, therefore it cannot generate $Sig_{T*}(T_j)$ from $Sig_{T*}(E_{Kj}(T_j))$ to put it before V_j.
- *VM may not put signed token in ActiveTokenList*: *VM* cannot deny putting of $Sig_{T*}(T_j)$ on *ActiveTokenList* because $Sig_{T*}(T_j)$ has the signatures of all *TMs*.
- *VM* may not give CN_j, or give incorrect CN_j to V_j*:* As $Sig_{T*}(T_j)$ is already open to the public, *VM* cannot deny giving of CN_j. Also as $E_{T*}(CN_j)$ is open on *ConfNoList*, *VM* cannot give incorrect CN_j to V_j. Although it is possible that *TMs* encrypt CN_j incorrectly, this dishonesty and the responsible entities are detected at the disruption detection stage, therefore *TMs* cannot encrypt *CNs* incorrectly.

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 and (3) finally after confirming the successful registration of the vote on *VotingPanel*, V_j approves its vote by putting $Sig_{T*}'(T_j)$ in *VotingPanel* as shown in Fig. 2 (e). Here as V_j asks *TMs* to encrypt v_jr_j instead of v_j while generating secret random number r_j, TM_1,---,TM_P cannot know v_j. Also encrypted v_jr_j is further multiplied by encrypted x_j 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 $Sig_{T*}'(T_j)$ even after $Sig_{T*}(T_j)$ is disclosed, only V_j can approve its vote, consequently V_j cannot claim any dishonesty about its vote after its approval. This sub-stage proceeds as follows:

1. V_j generates its secret random number r_j and asks TM_1,---,TM_P to encrypt its randomized vote v_jr_j to $\{E_{T*}(v_jr_j), E_{\underline{T*}}(r_j)\}$ while encrypting them by 2 secret encryption keys.

2. Then *VM* sends encrypted unknown random number $\{E_{T*}(x_j), E_{\underline{T*}}(x_j)\}$ to V_j that is prepared in advance.

3. V_j calculates $E_{T*}(v_jr_j)E_{T*}(x_j)E_{T*}(CN_j) = E_{T*}(v_jCN_jr_jx_j)$ and $E_{\underline{T*}}(r_j)E_{\underline{T*}}(x_j) = E_{\underline{T*}}(r_jx_j)$.

4. V_j submits $\underline{E}_{T*}(v_jCN_j) = \{E_{T*}(v_jCN_jr_jx_j), E_{\underline{T*}}(r_jx_j)\}$ and $E_{T*}(CN_j)$, and puts them in the position corresponding to T_j in *VotingPanel*.

5. TM_1,---,TM_P repeatedly sign on $\underline{E}_{T*}(v_jCN_j)$ and $E_{T*}(CN_j)$ in *VotingPanel* by the first form of their signatures *i.e.* calculate $Sig_{T1*}(\underline{E}_{T*}(v_jCN_j))$ and $Sig_{T1*}(E_{T*}(CN_j))$.

6. After confirming the correctness of signatures of its vote on *VotingPanel*, V_j submits $Sig_{T*}'(T_j)$ to *VM* as its approval.

7. TM_1,---,TM_P repeatedly sign on $\underline{E}_{T*}(v_jCN_j)$ by the second form of their signatures *i.e.* calculate $Sig_{T2*}(\underline{E}_{T*}(v_jCN_j))$.

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 CN_j is valid and signatures *i.e.* $Sig_{T1*}(v_jCN_j)$ and $Sig_{T2*}(v_jCN_j)$ are consistent.
- *VM may not put vote or put incorrect vote on VotingPanel:* As $Sig_{T*}(T_j)$ is already 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 corrupt votes in VotingPanel:* As *VotingPanel* is open to the public, no one can modify or delete votes before they are moved to *TallyingPanel*.
- *TMs may sign their 2nd signatures incorrectly:* Because a vote with the 1st signatures of *TMs* has been approved as correct, the voter can claim that the signatures are inconsistent.

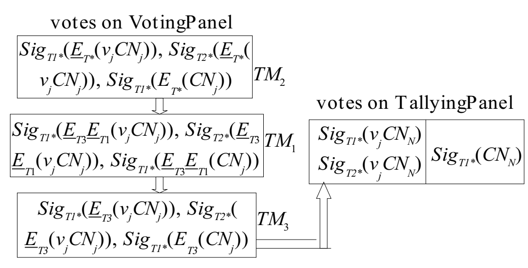

Figure 3. Procedures in Tallying stage.

D. Tallying Stage

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

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, inconsistencies are detected at the disruption detection stage.

- *Tallying managers may add votes*: Anyone can detect the addition 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 also detectable by anyone.

E. Disruption Detection Stage

If any inconsistency is found in *TallyingPanel*, Disruption detection manager *DM* detects liable entities. Fig. 4 are examples of votes on *TallyingPanel*. The 1st vote (1st row) is accepted because two different forms of vote v_jCN_{18} are same and also CN_{18} is registered. The 2nd vote (2nd row) is not consistent because the candidates within the two signed forms are different. The 3rd vote (3rd row) is inconsistent because CN_{-10} is not registered. The 4th and 5th votes (4th and 5th rows) are also inconsistent because of duplicated *CNs*. *DM* detects the liable entities as follows. When an inconsistent vote $\underline{v}_j$ is found, *DM* asks *TMs* to encrypt $\underline{v}_j$ in the reverse order of the tallying stage to find the encrypted form of $\underline{v}_j$ in *VotingPanel*, namely each TM_i encrypts $\underline{v}_j$ and discloses the result with its input vote in the tallying stage that matches with $\underline{v}_j$. When this matching chain fails, the dishonest TM_i is found. Here 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.

$Sig_{T1*}(v_jCN_{18}), Sig_{T2*}(v_jCN_{18})$	$Sig_{T1*}(CN_{18})$	√
$Sig_{T1*}(v_j\text{''}CN_2), Sig_{T2*}(v_jCN_2)$	$Sig_{T1*}(CN_2)$	x
$Sig_{TP1*}(v_jCN_{-10}), Sig_{TP2*}(v_jCN_{-10})$	$Sig_{TP1*}(CN_{-10})$	x
$Sig_{T1*}(v_iCN_{25}), Sig_{T2*}(v_iCN_{25})$	$Sig_{T1*}(CN_{25})$	x
$Sig_{T1*}(v_jCN_{25}), Sig_{T2*}(v_jCN_{25})$	$Sig_{T1*}(CN_{25})$	x

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

Figure 4. Possible vote disruptions.

VI. Requirements Analysis

Proposed scheme satisfies the requirements of e-voting as follows.

Privacy: No one knows links between encrypted votes on *VotingPanel* and decrypted votes on *TallyingPanel* because of encryption and decryption shuffles of votes and *CNs*.

Accuracy and Universal-verifiability: Signatures on tokens and uniqueness of *CNs* ensure that all and only eligible votes are counted.

Fairness: No single entity can decrypt interim voting results because votes on *VotingPanel* are repeatedly encrypted by multiple *TMs*.

Receipt-freeness and incoercibility: Voters know only their tokens, encrypted votes and encrypted *CNs*, and all of them cannot be linked to their votes. Therefore the scheme is *receipt-free*. In addition, when decrypted votes in two different signed forms are equivalent, no one can claim that votes are disrupted, therefore the scheme is secure against *randomization attacks*. Although in *VoterList*, *IDs* of voters are open to the public, the abstention of registered voters cannot be confirmed because tokens are anonymous, therefore voting 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*.

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

Robustness: Voters can disrupt only their votes, and either of *VM* and *TMs* cannot disrupt votes because inconsistent votes are detected at the disruption detection stage.

Scalability: CNs simplify the computations required by individual entities *e.g.* voters, tallying authorities etc. while maintaining the total accuracy of the election.

Practicality: The scheme is based on weaker assumptions about trustworthiness regarding entities *i.e.* nothing can corrupt the scheme unless multiple entities conspire.

VII. Conclusions

The proposed e-voting scheme makes votes verifiable in a simple way, and no entity even voters knows the links between voters and their votes. Also the simplified computational requirements of individual election entities makes the scheme scalable and practical. Namely, the scheme satisfies all the essential requirements of e-voting. According to simulations, the computation time required for the proposed scheme is substantially small compared with already proposed schemes

References

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

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

[3] J. Benaloh and D. Tuinstra, "Receipt-free secret-ballot elections," in Proceedings of the 26th Symp. on Theory of Computing (STOC'94), New York, pp. 544- 553, 1994.

[4] M. Hirt and K. Sako, "Efficient Receipt-Free Voting Based on Homomorphic Encryption," in EUROCRYPT '00, LNCS 1807, pp. 539-556. Springer, 2000.

[5] J. Schweisgut, "Coercion-resistant electronic elections with observer," 2nd Int. Workshop on Electronic Voting, Bregenz, August 2006.

[6] A. Juels and M. Jakobsson, "Coercion-resistant electronic elections," Cryptology ePrint Archive, Report 2002/165, 2002.

[7] A. Acquisti, "Receipt-free homomorphic elections and write-in ballots," Cryptology ePrint Archive, Report 2004/105, 2004.

[8] S. Tamura and T. Yanase, "Information Sharing among untrustworthy entities", IEEJ Trans. EIS, Vol. 125, No.11, pp. 1767-1772, 2005.

[9] S. Tamura, A. K. Md. Rokibul and H.A. Haddad, "A probabilistic and commutative Re-Encryption Scheme," unpublished.

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