12th International CSI Computer Conference (CSICC'07)
Shahid Beheshti University, Tehran, Iran, 20-22 February 2007

An Electronic Voting Scheme through Blind Signature

Toktam Taghavi, Abbas Ghaemi Bafghi, Mohsen Kahani
Department of Computer Engineering, Ferdowsi university of Mashhad
Mashhad, Iran
to_ta70@stu-mail.um.ac.ir, ghaemib@um.ac.ir, kahani@um.ac.ir

Abstract

Many e-voting schemes have been proposed in the literature. However, none of them is both secure and practical. In this paper, a practical and secure electronic voting protocol for large-scale voting over the Internet is investigated. Blind signature is applied to a voter's ballot making it impossible for anyone to trace the ballot back to the voter. Unlike previous blind signature based schemes, in which the authority directly signs its blind signature on voters' ballots, the authority in the proposed scheme signs blind signature on the voter marks that are generated by voters from ballot serial numbers. Moreover, threshold cryptosystem has been used to guarantee the fairness of the voting process. Using blind signature, this scheme can support all types of election easily and flexibly. Since we haven't use complex cryptographic techniques the proposed scheme is suitable for large scale elections.

Keywords

Blind Signature, Threshold Cryptosystem, Internet, Voter mark, Digital Signature, Public key cryptosystem

1. Introduction

Voting can be time consuming, inconvenient as well as expensive, especially when the voters and administrators are geographically distributed. With the rapid expansion of the Internet, electronic voting appears to be a less expensive alternative to the conventional paper voting. Electronic voting overcomes the problem of geographic distribution of the voters as well as vote administrators. It also reduces the chances of errors in the voting process. However, in order for electronic voting to replace conventional mechanisms, it must provide the whole range of features that conventional voting systems have. Further, due to the inherent lack of security in the Internet, electronic voting systems need to be carefully designed; otherwise these systems become more susceptible to fraud than conventional systems.

Electronic voting has been intensively studied for over the last twenty years. Up to now, many electronic voting schemes have been proposed, and their security as well as their effectiveness has been improved. However, no complete solution has been found in either theoretical or practical domains [1].

The aim of our work has been to review the contemporary state of research in the field of electronic voting, and to introduce a new solution that enhances effectiveness. In this paper, we propose a practical and secure electronic voting protocol that is suitable for large scale voting over the Internet.

In order to be usable in practice, electronic voting scheme has to satisfy some requirements. Generally we can classify the requirements of electronic voting into the following three categories as follows:

1. *Basic Requirements:* unreusability, eligibility, privacy, completeness, soundness and fairness.
2. *Extended Requirements:* individual verifiability, universal verifiability, receipt-freeness and open objection.
3. *Practical Requirements:* flexibility, mobility and scalable.

Basic requirements are satisfied in the most electronic voting systems and their implementation is relatively easy. But extended requirements are hard to implement and in many case, they require large amount of computation and communication.

1.1. Classification of Schemes

Electronic voting schemes found in the literature can be classified by their approaches into the following three categories:

1. Schemes using mix-net; eg. [2]
2. Schemes using homomorphic encryption; eg. [3]
3. Schemes using blind signature; eg. [1]

Voting schemes based on mix-net are generally not efficient because they require huge amount of computation for multiple mixers (mixing and proving correctness of their jobs)

The idea of using homomorphic encryption in electronic voting is to sum the encrypted votes, and then decrypt the sum—without decrypting individual votes. So concealing the voters identity is not required, it can be attached to the ballot at all times. Voting schemes based on homomorphic encryption use zero-knowledge proof techniques to prove the validity of ballot. In this approach Achieving universal verifiability is easy and there have been extensive researches to provide receipt-freeness. The communication complexity in schemes using homomorphic encryption is quite high. Coalition of all authorities usually can decrypt the voter's vote and violate the privacy. In addition, these schemes do not support any other election type except yes-no or 1-out-of-L voting, and can be extended to K-out-of-L voting or 1-L-K voting. More choices can be added by doing several

simultaneous yes/no polls, but each added choice then adds to the complexity of the scheme.

In voting schemes based on blind signature technique each individual vote is decrypted so voter must send his vote anonymously to ensure privacy. These schemes are simple, efficient, and flexible, but providing receipt-freeness in these schemes is almost hard, because the voter's blind factor can be used as a receipt of his vote, therefore a voter can prove his vote to a buyer. Schemes using blind signature, naturally realize the multiple value and since ballots are decrypted, invalid ballots can be detected, and we don't need zero knowledge to construct proofs for verifying correctness of the ballots. This approach is considered to be the most suitable and promising for large scale elections. Since the communication and computation overhead is fairly small even if the number of voters is large. It is very compatible with the framework of existing physical voting systems.

1.2. Outline of the Paper

In the next section, some of the related work is discussed. In session 3, our administering agents are presented and we explain our protocol briefly without details. The details of the protocol are explained in session 4, which is followed the analysis of the proposed voting protocol and conclusion.

2. Related Works

The first actual voting protocol employing blind signatures appeared in 1992 [4] and the first implementation in 1997 [5]. A somewhat improved version was offered by He & Su in 1998 [6]. The basic idea underlying all of these schemes is to employ two logical authorities, a registrar and a tallier. The registrar validates eligible voters and provides them with anonymized certified voting tags, using blind signatures. The voters then cast their (blindly signed) ballots over anonymous communication channels to the tallier.

In [4] the encrypted ballot that is cast by the voter contains the voter's signature that allows the voter to identify its ballot in the published list. Thus any entity that can verify the voter's signature is able to link a voter to a cast ballot and anonymity of the voter is not ensured.
Reference [7] proposed a voting scheme which he himself later showed to lack the postulated receipt-freeness; a repaired version by the same author, making use of blind signatures, appears in [8]. Although theoretically sound, but this scheme suffer from the fact that it depend on untappable channels or voting booths. These cannot be implemented over the Internet, making the schemes not practical for a real world remote electronic election.

In [9] the voter trying to vote twice will be traced. The scheme requires existence of the anonymous channel supporting replays (recipient of the anonymous message can send a replay to the anonymous sender).
The eligibility is achieved if the authority is honest. If a voter complains his privacy can compromise (at least the authority will get to know his vote).

In [1] voter cast his or her ballot anonymously, by exchanging untraceable yet authentic messages. It is suitable for large scale voting over the Internet but it can't provide receipt-freeness and fairness.

3. Model of Electronic Voting

In this section we will overview the proposed voting scheme briefly and describe the model of electronic voting.

3.1. Administering agents:

1. A Certificate Keys Authority – *CKA*. He certificates the public keys of voters and authorities.
2. A Voter Registration Authority – *VRA*. He verifies the identities and eligibilities of voters and then issues Registration Certificate (*RC*) to voters in the registration stage.
3. A Voter Certifying Authority – *VCA*. He prepares blank ballots and distributes one to each voter also certifies a ballot that is cast, has been cast by a registered voter and that voter has cast one and only one ballot.
4. A Vote Compiler – *VC*. Each filled ballot cast by a voter is delivered to the vote compiler. After creating decryption key, the vote compiler tallies the votes and announces all the relevant statistics pertaining to this voting process.
5. N talliers –T_j (j = 1... N). They cooperate with each other and create private key of the system that has been shared between them before starting the election. This key will use to decrypt the votes.
6. A *judge* – an independent monitoring authority for handling the objections. Each voter that has complaint about his vote at each stage during the e-voting sends his complaints to this authority.

3.2. Notations

In this session we define a set of notations, and then use them to describe our protocols. The notations are defined as follow:

1. X – The identity of an agent involved in the voting protocol.
2. X_e – agent X's public key
3. X_d – agent X's private key
4. M – a message
5. h (M) – a digest of the message M
6. $[m, X_e]$- an entity m encrypted with X's public key, where X is the recipient of m
7. $[m, X_d]$– an entity m signed with X's private key, where X is the originator of m
8. VS_e – public key of the voting system
9. S–secret key of the voting system that is shared between talliers.

3.3. Trap-door bit commitments

In a trap-door bit commitment scheme, where a voter V has committed to a message M, it is possible for V to open M in many different ways.

We use trap-door bit commitment that [7] used to achieve receipt freeness. In this section we describe the trap-door bit-commitment that we will use in vote casting stage. But there is a weakness in this approach that we are currently working on it to solve efficiently.

Several parameters p, q, g, h are generated and published before the election by the voting system. Where p and q are prime, $q \mid p-1$, g and h are in Z_p^*, and

$$q = order(g) = order(h)$$

(i.e. $g^q \equiv h^q \equiv 1 \,(mod\ p), g \neq h \neq 1$)

Here ω such that $h = g^{\omega} \bmod p$ is not known to any party. These prime numbers p and q are different from prime numbers that are used to generate pair keys of voting system.

Voter randomly generates $\alpha \in Z_q$ and calculates $G = g^{\alpha} \bmod p$. They defined

$$\beta = BC(v,k) = g^v G^k \bmod p$$

Where v is voter's vote and k is a random number
Here $BC(v,k)$ is a trap-door bit-commitment, since voter can open this bit commitment in many ways, (v,k), (v',k'), etc., using α such that

$$v + \alpha k \equiv v' + \alpha k' \ (mod\ q)$$

In [7] the trap door bit-commitment is essential for satisfying receipt freeness. If the value of α is generated by voter as specified, then the scheme satisfies the receipt- freeness. However, if α is generate by a coercer and he forces voter to use $G = g^{\alpha} \bmod p$ for voter's bit-commitment, then voter cannot open $\beta = BC(v,k)$ in more than one way, since voter does not know α. Hence, the voting scheme is not receipt free and coercer can coerce voter.

To prevent this attack, voting authority must be sure that voter knows α that is used to create bit-commitment. On the other hand the authority must not know α because he can trace the voter through α. This is still an open area in this research and we will work on it in our future works

3.4. Overview of the Proposed Voting Protocol

Before the voting period, voters have to register with Voter Registration Authority (*VRA*) to be an eligible voter. The authority then issues a certificate for each such registered voter. Then during the election voter sends his certificate to the Voter Certifying Authority (*VCA*) to obtain a blank ballot. If he is an eligible voter, *VCA* sends a blank ballot to him. Each blank ballot has a unique serial number. Voter uses this serial number to create voter mark, and then he blinds and signs it. After that voter sends blinded voter mark to the *VCA* to obtain *VCA*'s signature on his blinded voter mark. If *VCA* hasn't already received a blinded voter mark from that voter, he signs blinded voter mark for him and then sends signed blinded voter mark to voter. Voter un-blinds it and obtain *VCA*'s signature on his voter mark, then he generate secret random number α and calculates $G = g^{\alpha} mod$ then using G, his vote and a random number, he generates trap door bit-commitment. In vote casting stage voter sends his signed voter mark, his vote, G, trap door bit-commitment and the random number is used to create it, to the Vote Compiler (*VC*) through an anonymous channel. Vote casting in our protocol is a process similar to uploading to a site-that is the identity of the voter is not provided in the message.

At the deadline of the voting talliers cooperate with each other to create secret key of the voting system and then send this secret key to the *VC* to decrypt the votes.

4. Proposed Electronic Voting Scheme

In this session we have described our protocol completely. We have explained each stage of e-voting with its details.

4.1. Pre-System Set up

1. Each entity goes to the election site and download key generation applet. After that he generates a pair of keys (such as RSA public key and private key).
2. Each entity (voters and authorities) must go to the Certificate Keys Authority (*CKA*) and registers his public key and receives his public key certificate (*CKA*'s signature on his public key) and *CKA*'s public key. So each entity has his own public key that is certified by *CKA*.

Note: this stage is performed in one's presence and it isn't possible through Internet. As a general rule each person must first register his public key and then he can use it in exchanging message. Therefore this stage isn't dependent on the voting protocol but this is dependent on electronic society.

4.2. System Set up

- **Sharing secret key of voting system:** *N* talliers ($T_1 \ldots T_N$) execute the key generation protocol of (*t; N*)-threshold encryption scheme and as a result each tallier T_i possesses his share S_i of a secret *S*. Any cooperation of more than *t* talliers can decrypt an encrypted ballot. [10]
- **Publishing authorities' public keys:** public key of the *VRA*, *VCA* and *VC* that contains *CKA*'s signature, are published on the site.
  Since all voters have to get public key of the *VCA* and *VC* from the site during the voting process, bottleneck may be occurred. To prevent this problem we can send the public key of the *VCA* and *VC* together with the message that *VRA* send to voters (session 4.3).
- **Publishing the list of candidates:** The *VRA* publishes the list of *L* candidates, their certificates and their advertisement on the public site.
  Candidate's certificate has *VRA*'s signature and ensures that this candidate has already registered as candidate and he is eligible for it.

4.3. Voter Registration

Voter Registration Authority ⇒ *VRA*

$Voter\ identification = request, [request, V_a], voter's\ public\ key\ certificate$

1. $V \longrightarrow VRA$: $voter\ identification$
2. $VRA \longrightarrow V$:
$[\{request, name\ of\ the\ e-voting\}, VRA_d]$
$= voter\ Register\ Certificate\ (RC)$

In any election, an individual must register to be an eligible voter. This is done before the voting period. The voter must register with Voter Registration Authority. This authority prepares a list of registered voters from all people who have expressed a willingness to vote by verifying the identity of such people.

Voter registration is done as follows:

1. Voter sends his request, signed request with his private key and his public key certificate to the Voter Registration Authority. (Request at least consists of voter's *ID*). *VRA* makes sure that the voter himself sends this request because of the voter's signature on it.
2. Registration Authority checks the users with the National Registration Database* to determine the eligibility of a voter and his precinct. If a voter is eligible and hasn't registered before, The authority issues a certificate for each such registered voter that contains the voter's request and name of the e-voting that are signed with *VRA*'s private key. If voter doesn't have the right, *VRA* gives him an error message.

Note: *VRA* makes sure that the voter himself has sent the request because of the voter's signature on it. If voter doesn't sign his request, anyone that knows his public key certificate and his *ID* can send this request and gives register certificate. So when original voter wants to register, *VRA* sends an error message to him because his *ID* has been already existed in *VRA*'s database. However counterfeit voter won't be able to cast vote instead of original voter because in voter certification stage he must sign voter mark. But he can prevent original voter to vote.

Objection: When a voter receives an error message from *VRA* that means he isn't eligible to vote, he can complain to *judge*. He sends following message to the *judge*.
$V \longrightarrow judge$:
$request, [request, V_d], voter's\ public\ key\ certificate$

4.4. Voter Certification

Voter Certifying Authority ⇒ *VCA*

1. $V \longrightarrow VCA$:
$voter\ Register\ Certificate, voter's\ public\ key\ certificate$
2. $VCA \longrightarrow V$: $[\{y, [h(y), VCA_d], N\}, V_e]$
N is a random nonce; *y* is a ballot serial number
3. $V \longrightarrow VCA$:
$[\{m \times [r, VCA_e], [h(m \times [r, VCA_e]), V_d], V_{ID}, N\}, VCA_e]$
m is a voter mark generated by the voter
r is a random number that is blind factor
4. $VCA \longrightarrow V$: $[[\{m \times [r, VCA_e]\}, VCA_d], V_e]$

1. The voter sends his Register Certificate and his public key certificate to *VCA*.
2. *VCA* first makes sure that the voter is a registered voter by verifying the *VRA*'s signature on the voter register certificate, and then checks whether he or she has received blank ballot before. if he hasn't received blank ballot before *VCA* register voter's Register Certificate and voter's public key certificate and voter's *ID* in his database also he verifies voter's public key certificate if it is *OK* saves voter's public key in his database then sends a blank ballot encrypted with the voter's public key.

   The blank ballot is a message of two fields (i) the ballot serial number field, *y* and (ii) *VCA* signed digest of the ballot serial number, [h(*y*), VCA_d]. *VCA* generates a unique serial number, *y*, for every voter and saves it in his data base then creates a list of ballot serial numbers and voter register certificates. This is the list of the blank ballots issued and will be published by *VCA* at the end of the voting. So everyone can check that blank ballots issued to the registered voters.

   In this stage *VCA* sends a random nonce with the blank ballot to the voter to ensure that the voter himself responds the message. In the next stage that voter sends his blinded voter mark to the *VCA*, he adds this nonce to his message. We use nonce to prevent replay attacks. if we don't use nonce in this step, somebody can create a message and wants voter to sign it in another process (not voting process) so he gain the voter's signature on the message then he can send that message with voter's *ID* to the *VCA*.
3. When the voter receives the message, he makes sure that *VCA* sends this blank ballot because of *VCA*'s signature on the blank ballot also he makes sure that the blank ballot has not been tampered with during transit, including that nobody has put an identifying mark within the blank ballot .The voter then retrieves the serial number, *y*, from the received message. Using the serial number, *y*, the voter creates a voter mark, *m*, as follows: The voter pads *y* with a fixed length random number to obtain a number *x*. The voter then computes a hard to invert permutation, *m*, of *x*. The value, *m*, is the voter mark [1]. Note that since the serial number *y* is unique for every voter the voter mark is unique to every voter. However from the voter mark it is not possible to obtain the serial number *y* and hence impossible to identify the voter.

   The voter then blinds the voter mark with a random number, *r*, to get the blinded voter mark *m*× [*r*, VCA_e]. The voter also computes a digest of the blinded voter mark and signs the digest.
   The voter encrypts the blinded voter mark, the signed digest of the blinded voter mark, his *ID* and the random nonce that *VCA* had sent to him in the previous stage, with *VCA*'s public key.

   Note: We need one to one relation between x and voter mark and we use hard to invert permutation to achieve this property.

* This database contains information about all people who lived in the country such as name, family, father's name, date of birth, user ID, etc.

4. *VCA* first makes sure that the voter is a registered voter and has get a blank ballot from *VCA,* by searching voter's *ID* in his database. (All registered voters that have given blank ballot are in *VCA*'s database). *VCA* also makes sure that the voter has not submitted earlier, another blinded voter mark to sign. Since the vote cast by the voter later on will be accompanied by the voter mark, this step effectively ensures that the voter casts one and only one vote. By verifying the voter's signature on the digest of blinded voter mark and verifies random nonce, *VCA* makes sure that the voter himself sends blinded voter mark also *VCA* makes sure that the voter mark has not been tampered with in transit. *VCA* saves blinded voter mark and signed digest of the blinded voter mark with voter's private key in his data base.

   VCA verifies voter's signature on his blinded voter mark then signs his blinded voter mark. *VCA* saves signed blinded voter mark with his private key in his database. *VCA* encrypts signed blinded voter mark, with voter's public key and sends it to the voter.

Objection: Voter verifies *VCA*'s signature on his blinded voter mark if it isn't *OK*, he can complain to the *judge*. In this case voter must send the signed blinded voter mark that receives from *VCA* and there are *VCA*'s signatures on it, to the *judge,* on the other hand *VCA* must send blinded voter mark that receives from voter and there is voter's signature on it, to the *judge*. The *judge* verifies the *VCA*'s signatures on them. If the *VCA*'s signature isn't *OK*, *judge* forces *VCA* that re-sings the blinded voter mark for that voter.

$V \longrightarrow judge$: $[[\{m \times [r, VCA_e]\}, VCA_d], judge_e]$

VCA $\longrightarrow$*judge*:

$[\{m \times [r, VCA_e], [h(m \times [r, VCA_e]), V_d], V_{ID}\}, judge_e]$

Note that here blinded voter mark is sent to the *judge* therefore *judge* doesn't know actual voter mark and so the privacy of voter remains secret.

4.5. Vote Casting

Vote Compiler ⇒ *VC*

1. $V \longrightarrow Public\ site$:

$[\{[\{vote, \beta, k, G, [m, VCA_d]\}, VS_e], \beta, [m, VCA_d]\}, VC_e]$

Reminder: $\beta = BC(vote, k)$ is trap door bit-commitment and $G = g^{\alpha} \bmod p$

2. $Public\ site \longrightarrow VC$:

$[\{[\{vote, \beta, k, G, [m, VCA_d]\}, VS_e], \beta, [m, VCA_d]\}, VC_e]$

3. $VC \longrightarrow Public\ site$:

$[h([\{vote, \beta, k, G, [m, VCA_d]\}, VS_e]), VC_d]$,

$[h(\beta, [m, VCA_d]), VC_d]$

4. $Public\ site \longrightarrow V$:

$[h([\{vote, \beta, k, G, [m, VCA_d]\}, VS_e]), VC_d]$,

$[h(\beta, [m, VCA_d]), VC_d]$

Vote casting in our protocol is a process similar to uploading to a site – that is the identity of the voter is not provided in the message – unlike the other steps.

At best, an IP address be traced back but cannot be linked with a voter. That way we ensure the anonymity of the voter.

The voter "un-blinds" the signed blinded voter mark and obtains *VCA's* signature on it.

1. The voter now prepares a fixed length message of a pre-determined format (the format is announced to all voters prior to voting initiation) and indicates his or her vote as the message's content. The voter appends his signed voter mark, to this message (signed by *VCA*). Recall that it is not possible to recover the serial number from the voter mark, so it is not possible to identify the voter by the voter mark. Also voter appends his G, his trap door bit-commitment and the random number is used in the trap door to his vote and then encrypts them with public key of the voting system afterwards, encrypts this message, signed voter mark and trap door bit-commitment (β) with *VC*'s public key and, after waiting for a random amount of time, uploads the same onto a publicly up loadable site announced to voters before. The mechanism used to upload does not associate, in any manner, the voter's identity with the uploaded material – for example an anonymous/guest ftp mechanism or proxy mechanism.
2. Periodically, *VC* downloads cast votes from this site. *VC* checks that it has not received this voter mark before.
3. If *VC* has not received this voter mark before, he saves this voter mark, β and casted vote in his database then signs the digest of the casted vote and the digest of the voter mark and β then stores them in his database and then uploads them to the public place.

   Casted vote means: $[\{vote, \beta, k, G, [m, VCA_d]\}, VS_e]$
4. Sometime later the voter retrieves these signed from the public place. This guarantees that *VC* has received the voter's vote. Voter verifies the *VC*'s signature on the hash value of his casted vote and makes sure that his vote hasn't been tampered.

Note: $[h(\beta, [m, VCA_d]), VC_d]$ will use for universal verifying. For each β and signed voter mark that published at the end of e-voting, corresponding signed of them that have *VC*'s signature should be existed on the public site. So everyone makes sure that all counted votes are confirmed with *VC* and send through public site. Also this message ensures that *VC* cannot claim later that it did not receive a vote from that voter mark and voter can use this message for objection if his vote is deleted or changed. We discuss more about this objection in next stage.

Note: When one message is sent to the site nobody can erase or alter it later except administrator. But if administrator erases or alters some votes, everyone can detect this attack.

Objection: If casted vote has been tampered, voter can complain. The voter sends his casted vote and signed digest of it with *VC*'s private key to the *judge* through anonymous channel (public sit in our protocol).

$V \longrightarrow judge$: $[\{[\{vote, \beta, k, G, [m, VCA_d]\}, VS_e],$

$[h([\{vote, \beta, k, G, [m, VCA_d]\}, VS_e]), VCd]\}, judge_e]$

Note that since casted vote exists on the public site the voter can't change it before sending to the judge

4.6. Vote Counting

At the deadline of voting, *N* talliers jointly execute the (*t; N*)-threshold decryption protocol to obtain secret key of the voting system (for more details see Appendix B). A threshold *t* denotes the lower bound of the number of talliers that is guaranteed to remain honest during the protocol. Because the secret key of the voting system, *S*, is shared among *N* talliers, any subset of *t* talliers can construct the secret key. Then talliers send secret key to the Vote Compiler to decrypt the votes. There are two scenarios to send secret key to the Vote Compiler:

First scenario:

1. Talliers cooperate with each other and construct the secret key.
2. Alternatively each tallier that participates in constructing the secret key, signs it with his private key.
3. Signed secret key is encrypted with Vote Compiler's public key.
4. Sequence of talliers that sing the secret key is encrypted with Vote Compiler's public key.
5. Encrypted secret key and encrypted sequence send to the Vote Compiler.

$T \longrightarrow VC$:

$[[\dots [\dots [[secret\ key, T_{d_1}], T_{d_2}] \dots, T_{d_t}] \dots, T_{d_n}], VC_e]$

$[\{d_1, d_2, d_3 \dots d_t \dots d_n\}, VC_e]$

T_{d_1}: First tallier's private key

T_{d_n}: Nth tallier's private key

When *VC* receives the message, he first decrypts it and then verifies tallier's signature on it. To verify tallier's signature, *VC* must know the sequence of the talliers that sign the secret key. This is the reason that we send sequence of talliers that sing the secret key together with signed secret key.

VC makes sure that at least *t* talliers participate in decryption process. When *VC* obtains secret key, he decrypts the votes that he saves in his database at vote casting stage.

Second scenario:

Through the key generation protocol, each tallier T_j will possess a share S_j of a secret *S*

1. Each tallier signs its S_j and send to the *VC*.
2. *VC* can construct the secret key with at least *t* signed S_j.

$$T_j \longrightarrow VC: [S_j, T_{d_j}]$$

When *VC* receives at least *t* such message from talliers he can construct the secret key and then he decrypts the votes.

4.7. Publishments

At the end of voting each entity publishes the following data:

a. VRA publishments:

i. list of registered voters (voter certificate)

b. VCA publishments:

i. number of blank ballots (N_{bb})

ii. serial numbers of the blank ballots together with the Register Certificate of the voters (*y*, *RC*)

Note: The number of blank ballots must be greater than or equal to the number of votes received but less than or equal to the number of registered voters.

$$(N_{bb} \geq N_v\ ,\ N_{bb} \leq N_{rv})$$

Indexed by the identity of the voters:

iii. blinded voter marks received from the voter, $m\times [r, VCA_e]$

iv. their digests signed by the voters $[h(m\times [r, VCA_e]), V_d]$

v. the corresponding blinded voter marks signed by *VCA* $[\{m\times[r, VCA_e]\}, VCA_d]$

Note: N (above items) ≥ N_v (number of received vote), N (above items) ≤ N_{bb}

Note: although it cannot be established by any entity other than the voter, for every vote that is cast there should be one and exactly one signed blinded voter mark with V*CA*.

c. VC publishments:

i. Signed voter marks and their corresponding β in random order. $[m, VCA_d], \beta, G$

ii. List of voter marks that haven't valid vote and corresponding error message.

iii. List of correct votes in random order.

iv. Non-interactive modification of zero-knowledge proof, σ to prove that the list of valid votes contains only correct open values of the list of β (bit-commitment votes) without revealing the linkage between β and vote [7]. In other words, *VC* publishes $(v'_1, \dots, v'_l)$, which is a random order list of votes. That is $v'_i = v_{\pi(i)}$ (i = 1,..., l), where π is a random permutation of l elements. Given $(\beta_1, \dots, \beta_l)$ and $(v'_1, \dots, v'_l)$, VC proves that knows (π, k_i) such that

$$\beta_i = BC(v_i, k_i)\ ,\ v'_i = v_{\pi(i)}$$

Without revealing (π, k_i)

4.8. Objections

After publishing the results, some voters may have a objection about their published vote. Here we describe that in each condition the voter how can send his objection to the judge that his privacy remain secret.

***Condition 1*:** when *VC* decrypts a vote he check that if β is the correct bit-commitment of the casted vote or not. If it's not *OK* he doesn't count that vote and at the end of counting he publishes corresponding voter mark with error message E_1 instead of β. E_1 means that β isn't correct bit-commitment of the vote. Also a voter may cast empty ballot, in this case *VC* publishes corresponding voter mark with error message E_2 instead of β and that vote isn't counted. Also a voter may cast a vote that has invalid content, for example the name that is in the vote, isn't the name of candidates that are published before. In this case *VC* publishes corresponding voter mark with error message E_3 instead of β and that vote isn't counted.

At the end of the voting when the results are published if a voter mark of the voter contains an error message

instead of β he can complain to the *judge*. He sends following message to the *judge* through an anonymous channel (in our protocol through public sit):

$V \longrightarrow judge$:

$$\left[\left[h\left(\left[\{vote, \beta, k, G, [m, VCA_d]\}, VS_e\right]\right), VC_d\right], judge_e\right]$$

Note that from the voter mark it is not possible to identify the voter.

Condition 2: After publishing the result if the voter's vote has been changed, the voter can complain. He sends his signed voter mark and β that there is *VC*'s signature on them and the signed voter mark and β that are published, to the *judge* through public sit.

$V \longrightarrow judge$:

$$\left[\{[h(\beta, [m, VCA_d]), VC_d], published\beta, [m, VCA_d]\}, judge_e\right]$$

Condition 3: After publishing the result if the voter's vote has been deleted, the voter can complain. He sends his signed voter mark and β that there is *VC*'s signature on them to the *judge* through public sit.

$V \longrightarrow judge$: $\left[\left[h\left(\beta, [m, VCA_d]\right), VC_d\right], judge_e\right]$

5. Analysis of the Voting Protocol

5.1. Basic Requirements

Unreusability: Each cast ballot is linked to a signed voter mark. When *VCA* signs a blinded voter mark it makes sure that it does not sign two blinded voter mark from the same voter. Also each voter mark corresponds to one and only one voter and each voter mark is unique. Because the seed that is used to generate the voter mark is the unique serial number, *y* the voter mark guarantees that two cast votes are not erroneously attributed to the same voter.

Eligibility: *VCA* makes sure that ineligible voters do not get a blank ballot because *VCA* first makes sure that the voter is a registered voter by verifying the *VRA* signature on the voter certificate. Also when *VCA* wants to sign a blinded voter mark he checks that voter's *ID* has already existed in his database as a valid voter.

Completeness: When the vote compiler, *VC* receives a cast vote with the signed voter mark; it signs the digest of it. *VC's* signature on the digest ensures that *VC* cannot claim later that it did not receive a valid vote. When the votes are officially published, a voter will be able to identify his or her vote by the voter mark. These two together guarantees every vote that cast to the *VC* is counted in the final tally.

Soundness: A vote can be invalid for three reasons (i) it has been cast by an ineligible voter or (ii) an eligible voter has voted more than once or (iii) the content of the vote is invalid. For example the ballot is empty or voter writes name of person that isn't a candidate, etc. The first case is explained in Eligibility and the second case is explained in Unreusability. Since all votes are decrypted at the end of e-voting so invalid votes are detected and they won't be counted in the final tally.

Fairness: In this protocol we use threshold encryption to encrypt the votes. The secret key is distributed between *N* talliers and cooperation of at least *t* talliers can decrypt the vote. At the deadline of vote casting talliers cooperate with each other and construct the secret key so before vote counting stage *VC* doesn't know the secret key. So during the voting *VC* won't be able to decrypt the votes.

Privacy: Is provided to the extent that a cast ballot is not traceable back to a voter without the voter's cooperation and that before a vote is cast, nobody other than the voter knows what the vote is.

At every stage, till the vote is cast, the voter makes sure that none of the agents has put an identification mark on his vote. When a voter casts a vote the only thing that can possibly be identified with the ballot is the IP address of the server that the voter used to cast the vote (the server can be thought to be like an open work station accessible everyone that wants to vote). This does not reveal the voter's identity. Although the voter mark generated by the voter contains the unique serial number, *y,* it is computationally infeasible to compute *y* from the voter mark. Also when the filled ballot is transferred to *VC*, it is encrypted (by *VC's* public key) in transit. Thus only the voter knows about his vote till such time as the vote is cast.

5.2. Extended Requirements

Individual verifiability: When the votes are published at the end of the voting, a voter can identify his/her vote by the voter mark. If the identified vote does not match the vote that the voter actually cast, voter can send a complaint to the judge without revealing his privacy. We discuss about this situation in session 4.8.

Universal verifiability: Voter Registration Authority publishes certificates of registered voters. Voter Certifying Authority publishes a list of ballot serial numbers and voter register certificates. So everyone can check that blank ballots issued to the registered voters and everyone can check that if The number of blank ballots is greater than or equal to the number of votes received but less than or equal to the number of registered voters.

VCA publishes blinded voter marks that there is voter's signature on them. So everyone makes sure that voter marks are created by eligible voters also *VCA* publishes signed blinded voter marks with his private key so everyone can check that *VCA* signs all valid voter marks. When voters send their vote through public sit *VC* signs their voter mark and their bit-commitment and put it on the site. $[h(\beta, [m, VCA_d]), VC_d]$use for universal verifying. For each β and signed voter mark that published at the end of e-voting, corresponding signed of them that have *VC*'s signature should be existed on the public site. So everyone makes sure that all counted votes are confirmed with *VC* and send through public site. Also this message ensures that *VC* cannot claim later that it did not receive a vote from that voter mark and voter can use this message for objection if his vote is deleted or

changed. We discussed more about this objection in session 4.8.

Receipt-freeness: The proposed election scheme provides receipt-freeness by using of trap door bit-commitment. So voter can open this bit commitment in many ways, (v, k), (v', k'), etc., using α such that

$$v + \alpha k \equiv v' + \alpha k' \ (mod\ q).$$

The most important thing in this way is that voter knows α to be able to open his bit-commitment in many ways. When VC publishes the result, he publishes signed voter marks and corresponding bit-commitment index by the voter marks. on the other hand *VC* publishes List of correct votes in random order. And then by using of Non-interactive modification of zero-knowledge proof, σ he proves that the list of valid votes contains only correct open values of the list of β (bit-commitment votes) without revealing the linkage between β and vote.

If voter himself creates α and he knows it our protocol is receipt-free but if voter doesn't know α he can't open his vote in many ways and our protocol isn't receipt-free. We are working to solve this weakness.

Open Objection: We discussed about this property before. In each stage we explained that how voter can complain while his privacy remains secret. We discussed this property in *objection* sessions.

5.3. Practical requirements

Flexibility: In this protocol we use blind signature and anonymous channel (public site) so we can use different type of voting because all individual votes are decrypted unlike homomorphic schemes that only sum of the votes is decrypted. Homomorphic schemes were designed to yes-no voting. More choices can be added by doing several simultaneous yes/no polls, but each added choice then adds to the complexity of the scheme. But blind signature schemes Support any type of the voting because all individual votes are decrypted.

Mobility: In this protocol voters use internet to cast their votes so anywhere that a voter can access to the internet he can cast his vote. Nowadays access to the internet is very easy so this protocol is practical.

Scalable: our scheme is good for large scale election with a lot of voters and candidates because we don't use complex cryptographic techniques. Since we use blind signature and anonymous channel (public sit) we don't need to use complex zero-knowledge techniques to achieve anonymity of the voter. Our scheme doesn't involve complex and high computational overheads that they may not be readily available.

6. Conclusion

In this paper we have proposed an efficient electronic voting scheme that is suitable for large scale voting over the Internet.

The protocol satisfies the core properties of secure voting systems – namely Unreusability, Eligibility, Privacy, Completeness, Soundness, Fairness Further the protocol ensures extended requirement such as Individual verifiability, Universal verifiability, Open Objection, Flexibility, Mobility and Scalable. Receipt-freeness in our protocol isn't achieved completely and we want to satisfy it in our future works.

Unlike homomorphic scheme we haven't used zero knowledge proofs to prove validity of votes because each vote is decrypted so the computation complexity is lower than homomorphic schemes. Proposed scheme support all type of voting without increasing computation or communication complexity.

We use the Internet for electronic voting so voters can participate in voting in any place they like over the Internet. Then electronic voting system can play an important role to increase the participation rate in voting and realize participatory democracy.

Acknowledgement

The authors gratefully acknowledge the Computer and Communication Research Center, Ministry of Information Technology and Telecommunication, Mashhad, Iran for their support.

References

[1] Indrajit Ray and Indrakshi Ray. "An Anonymous Electronic Voting Protocol for Voting Over The Internet".2001

[2] M. Abe, "Universally verifiable mix-net with verification work independent of the number of mix-servers", Advances in Cryptology – Eurocrypt'98, LNCSVol.1403, pages 437–447, Springer-Verlag,

[3] B. Lee and K. Kim. "Receipt-free Voting Scheme with a Tamper-Resistant Randomizer".2002

[4] A. Fujioka, T. Okamoto, and K. Ohta. "A practical secret voting scheme for large scale elections." Advaced in Cryptology - AUSCRYPT'92, 1992.

[5] L. F. Cranor and R. K. Cytron, "Sensus: A security-conscious electronic polling system for the Internet". Proceedings of the Hawaii International Conference on System Science, 1997, Wailea, Hawaii

[6] Q. He and Z. Su, "A new practical secret voting scheme for large scale election". Information Security Conference, 1998

[7] T. Okamoto, "An Electronic Voting Scheme", proc. Of IFIP'96, Advanced IT Tools, Chapman & Hall, pp.21- 30 (1996)

[8] T. Okamoto. "Receipt-free electronic voting scheme for large scale election", Proc. of Workshop on Security Protocols'97, LNCS (1361), 1997.

[9] Michael J. Radwin. An untraceable, universally verifiable voting scheme. 1995.

[10] Adi Shamir, "How to Share a Secret", Massachusetts Institute of Technology, 1979