A Practical Verifiable e-Voting Protocol for Large Scale Elections over a Network

Orhan Cetinkaya
Institute of Applied Mathematics,
METU, Ankara, Turkey
e113754@metu.edu.tr

Ali Doganaksoy
Department of Mathematics,
METU, Ankara, Turkey
aldoks@metu.edu.tr

Abstract

We propose a practical verifiable e-voting protocol which guarantees e-voting requirements: privacy, eligibility, uniqueness, uncoercibility, fairness, accuracy, robustness, individual verifiability, and universal verifiability. Unlike existing e-voting protocols we employ dynamic ballot instead of pre-defined usual ballot in order to strengthen accuracy and fairness of the protocol. In dynamic ballots, the ordering of candidates in the ballots is dynamically created and changes for each voter. Therefore the proposed protocol is called as "DynaVote".

DynaVote does not use complex cryptographic algorithms such as homomorphic encryption and does not require anonymous communication channels such as mix-nets since it employs PVID (Pseudo-Voter Identity) scheme which relies on blind signature. Besides it has no physical assumption such as untappable channels. Hence, DynaVote is a practical e-voting protocol for large scale elections. DynaVote is performed over a network such as the Internet. In order to achieve uncoercibility, DynaVote allows recasting without sacrificing uniqueness.

1. Introduction

Due to the rapid growth of computer technologies and advances in cryptographic techniques, electronic voting (e-voting) is now an applicable alternative for paper based voting. Many e-voting protocols have been proposed in the last several decades. Nevertheless, to the best of our knowledge, no practical and complete solution has been found for large scale elections over a network. We propose a practical verifiable e-voting protocol over a network for large scale elections that satisfies all e-voting security requirements.

Design of secure e-voting protocols over a network is not an easy task. It is much more difficult to achieve the e-voting requirements whereas employing the protocol over a network. Especially, satisfying uncoercibility, privacy, and eligibility are major problems of the e-voting over a network. In particular, avoiding from uncoercibility has more importance since voter casts his vote in an uncontrolled and unsupervised environment. In order to overcome this problem, we propose a solution for uncoercibility by allowing recasting without sacrificing uniqueness. We find a solution for privacy and eligibility by applying PVID (Pseudo-Voter Identity) scheme.

In literature, almost all of the proposed protocols try to prevent recasting by introducing some mechanisms. On the other hand, the proposed protocol fully supports recasting which provides a solution for coercibility problem in uncontrolled environments such as the Internet. Even if someone coerces voter, voter casts by that way. Later, he can recast new one and the old one is discarded in counting stage. So, practically it is not possible to coerce the voter or to buy vote from the voter since nobody can know whether the current vote will be the final one or not.

However, the e-voting protocols found in the literature either not fulfill uncoercibility requirement or make some physical assumptions such as voting booths or voting pools to overcome coercibility problem [9]. The proposed protocol provides uncoercibility with no such assumptions due to the vote recasting feature.

The proposed protocol needs an unlinkable pseudo identity mechanism. PVID scheme provides a pseudo identity which is an anonymous identity and it is unlinkable to the voter's real identity [4]. Thus we employ PVID scheme in order to satisfy voter anonymity. In existing e-voting protocols, voter generally uses his real identity while communicating with the authorities. On the other hand, in PVID scheme, voter uses a pseudo identity (PVID) which has no relation with real one. Voter can use it throughout the entire communication and he can easily hide his real identity. PVID scheme provides anonymity

without requiring any complex computational operations and cryptographic mechanisms. It only employs blind signature scheme.

Up until now, e-voting protocols have used either homomorphic encryption or anonymous communication channels mostly based on mix-nets. Anonymous channel implementations need expensive operations and complex calculations. Moreover, anonymous channels are not easy to set up and add substantial complexity to the protocol. For example, in mix-nets, many mix servers are needed. E-voting protocols based on homomorphic encryption have communication complexity. Homomorphic voting protocols are inefficient if there are many candidates or choices. On the other hand, PVID scheme just needs a blind signature and the cost of blind signature is reasonably small and cheap.

The proposed protocol is scalable as it supports small, mid, and large scale elections without any extra effort and the security of the system does not depend on the number of voters. It is suitable for large scale elections since it does not require any complex algorithms, specific hardware, complex computational operations or physical assumptions.

Unlike existing e-voting protocols we employ *"dynamic ballot"* instead of pre-defined usual ballot in order to strengthen accuracy and fairness of the protocol. Therefore the proposed protocol is called as *"DynaVote"*. Dynamic ballot concept is introduced in [3] and described in detail in this paper. In usual ballots, as the ballot is standard, voter's casting displays his actual vote. On the other hand, in dynamic ballots, voter's candidate selection has contextual meaning. It shows voter's actual vote only with the corresponding dynamic ballot. Therefore, any participant or authority including the counter cannot gain any knowledge about the tally before the counting stage. Dynamic ballot mechanism is not a simple user interface implementation; it is a part of the protocol itself and employed in the protocol layer, not only in the user interface layer.

Both individual verifiability and universal verifiability are the guarantors of accuracy and robustness respectively. We employ bulletin boards and hash functions to achieve verifiability of the protocol. The protocol is verifiable in each stage, and voter can object to any corruption without revealing his real identity. DynaVote has strong individual verifiability and universal verifiability.

In this paper, we propose a practical verifiable large scale e-voting protocol over a network which is a complete protocol since it guarantees the wide variety of e-voting requirements: privacy, eligibility, uniqueness, uncoercibility, fairness, accuracy, robustness, individual verifiability, and universal verifiability.

The remainder of the paper is organized as follows. In the next section related work is summarized. In Section 3 the proposed protocol DynaVote is illustrated. Then it is explained how DynaVote fulfills the e-voting requirements in Section 5. Finally, conclusions are drawn and future work is suggested.

2. Related Work

We propose a practical secure e-voting protocol which assures all aforementioned e-voting requirements for large scale elections over a network. Many e-voting protocols have been proposed in the last decades. Nevertheless, to the best of our knowledge, no complete solution has been found for large scale elections over a network.

Chaum [5] pioneered the notion of e-voting and then many protocols were proposed. The first practical e-voting protocol for large scale elections ensuring both privacy and fairness is of Fujioka *et al.* [1]. However, accuracy can be violated that the malicious authority can add votes if any voter abstains from voting in counting stage. The e-voting protocol proposed by Baraani *et al.* [6] extends [1]. The model of the original protocol has been further modified with the addition of a trusted third party. Later, Okamoto [7] proposed a solution for large scale elections based on untappable channel and even stronger physical assumptions, whereas the protocol suffers from practicality.

In general, the e-voting protocols, stating that they satisfy practicality and privacy, have strong assumptions such as anonymous communication channels and mix-nets. They suffer from computational costs to prove that their anonymizing is correct. Moreover, their implementations are actually not practical [1], [2], [6], [7], [10].

Another commonly proposed way of achieving privacy in e-voting protocols is to use homomorphic encryption [8], [11], [12], [13]. In e-voting protocols based on homomorphic encryption, a combination of encrypted votes yields accumulation of votes. The voting result is then obtained from the accumulation of votes whereas no individual ballot is opened and the corresponding individual vote remains secret. In these protocols, voting results are obtained easily so ballot tabulations are more efficient. However, homomorphic voting has a drawback where each vote must be verified to be valid, since without validation, correctness of the tallying cannot be guaranteed. When the number of candidates or choices is large, computational and communicational cost for the proof

and verification of vote validity is quite large that homomorphic voting actually becomes inefficient for large scale elections.

DynaVote neither requires anonymous communication channels and any other physical assumption nor uses homomorphic encryption and complex computational operations. It only uses RSA and PVID scheme [4] which is based on blind signature; and voting can occur entirely over existing networks such as the Internet.

There are some e-voting protocols in the literature which use neither anonymous channels nor homomorphic encryption in order to perform e-voting over a network [15], [16], [17]. These protocols suffer from accuracy as corrupted participants can make fraud without being detected. Besides they have no solution for uncoercibility.

Uncoercibility as an extension to receipt freeness was introduced by Benaloh *et al.* [8]. Recently some e-voting protocols have been proposed in order to satisfy uncoercibility in e-voting [12], [14]. Moreover, they use mix-nets and homomorphic encryption as others.

In most of the e-voting protocols, voter is allowed to vote only once since the uniqueness requirement is accepted as unreusability. However, DynaVote allows recasting to overcome uncoercibility problem. Besides, uniqueness is also assured with PVID scheme.

Our protocol contributes to the literature mainly by presenting a practical verifiable e-voting protocol which has the following properties: (i) DynaVote has no computational complexity in all stages of the protocol. Furthermore it has no physical assumption. Hence it is a practical protocol. (ii) DynaVote employs PVID scheme in order to achieve anonymous communication and guarantees privacy, eligibility, and uniqueness. It allows recasting and assures uncoercibility. It employs bulletin boards in all stages of the protocol and provides direct individual verifiability as well as universal verifiability. It uses dynamic ballots to strengthen accuracy and fairness. (iii) It is a complete protocol for large scale elections over an existing network such as the Internet.

3. The Proposed Protocol: DynaVote

The proposed protocol DynaVote has the following actors: Voter, Ballot Generator, Key Generator, Counter, and PVID Authority. The protocol consists of 3 distinct stages: Authentication & Authorization, Voting, and Counting. Authentication & authorization are performed before the election day. Voting is carried out during the election period. Later counting is performed.

Instead of using one election day, we employ an election period which can be several days. Our purpose is to gain more flexibility and more voter involvement. However, depending on the election policy, the voting duration can be one day as well.

In the authentication & authorization stage, we employ PVID scheme. The voting stage consists of 2 phases: Ballot obtaining phase and vote casting phase. In the ballot obtaining phase Ballot Generator provides dynamic ballot to the voter. In this phase, Key Generator provides vote encryption key to the voter over Ballot Generator as well. In the vote casting phase, voter selects his vote from the dynamic ballot and then encrypts his candidate selection by using vote encryption key. Lastly, voter casts his encrypted candidate selection by using his PVID. In the counting stage, votes are decrypted and counted.

In all stages bulletin boards are employed in order to increase security and trust in the protocol. Voter checks and verifies intermediate outcomes against bulletin boards. Chaum [5] introduced the concept of the bulletin board, a public broadcast channel with universally accessible memory where authorities may write information in the designated areas via secure communication that any party may read. All communications with the bulletin board are public and therefore can be monitored. Generally, data already written to a bulletin board cannot be altered or erased anymore, but it can be appended in case of need.

Before explaining each stage in detail, we provide the following notation and abbreviations:

(e_p, d_p): Voter's permanent public-private key pair used to communicate with PVID Authority.

(e_s, d_s): Voter's session public-private key pair used to communicate with Ballot Generator.

(e_v, d_v): Voter's session public-private key pair used to communicate with Key Generator and Counter.

(e_a, d_a): PVID Authority's public-private key pair.

(e_b, d_b): Ballot Generator's public-private key pair.

(e_k, d_k): Key Generator's public-private key pair.

(e_c, d_c): Counter's public-private key pair.

(e_z, d_z): Voting public-private key pair generated for Voter to cast his candidate selection.

$\check{E}_x(m)$: Encryption of message *m* with the public key e_x.

$\check{D}_x(m)$: Decryption/Sign of message *m* with the private key d_x.

H(m): One way hash function on message *m* used by the voter and authorities.

B: Dynamic ballot.

V': Voter's candidate selection depending on the dynamic ballot.

V: Voter's actual vote.

PVID-list: {$PVID_1$, $PVID_2$}, a list of approved anonymous pseudo identities which are unlinkable to the voter's real identity.

3.1. Authentication & Authorization Stage

This stage is performed prior to the election period. Voter applies PVID authority to obtain a PVID-list by using his real registration identity. Registration identity can be any widely used identity such as social security number. PVID-list is nothing but a list of approved anonymous pseudo identities which are unlinkable to voter's registration identity.

After completing this stage, voter obtains a PVID-list and he can use PVIDs at any time and place during the election period. Voter's real registration identity is hidden to the voting authorities. Thus, voter becomes anonymous while he is using the PVIDs in his communications with the voting authorities. Voting authorities can easily check the validity of any PVID by applying PVID Authority's public key on it. This stage is carried out as voter authentication and authorization. PVID Authority checks voter eligibility and issues voter's PVID-list.

PVID-list is a list of blindly signed identities. In PVID scheme, voter performs blind signature with PVID Authority in order to obtain PVID-list [4]. DynaVote employs PVID scheme for two identities. Voter creates an ID list {ID_1, ID_2} where each ID contains a random number as well as some meaningful keywords such as *ID = (Election Data, Authority Data, Random Number)*. Voter blinds the IDs separately with different random blinding factors r, and obtains message M_b which is the combination of blinded IDs.

Then the voter sends $\check{E}_a$(Registration ID, $\check{D}_p(M_b)$) to PVID Authority. PVID Authority checks voter's eligibility. If the voter is eligible and has not made any request yet, the PVID Authority signs blinded IDs in message M_b and obtains M_{bs}. which is the combination of blindly signed IDs.

Then PVID Authority sends $\check{E}_p(\check{D}_a(M_{bs}))$ back to the voter. PVID Authority employs threshold cryptography in signing process to prevent single authority corruption. Voter checks PVID Authority's signature on M_{bs} and then unblinds each blindly signed ID in message M_{bs} and obtains PVID-list = {$PVID_1$, $PVID_2$}. Actually, PVID-list is the list of signed IDs.

3.2. Voting Stage

In voting stage voter obtains a dynamic ballot and casts his candidate selection. Dynamic ballot mechanism is the main building block of the protocol and before going into detail we explain it in brief. In usual ballots, the order of candidates in ballot is pre-determined, so everyone at least authorities know the order of candidates. In dynamic ballots, the ordering of candidates changes randomly for each ballot.

In usual ballots, as the ballot is standard, voter's casting displays his actual vote. On the other hand, in dynamic ballots, voter's candidate selection has contextual meaning. It shows voter's actual vote only with the corresponding dynamic ballot. Note that dynamic ballot mechanism is not a simple user interface implementation; it is a part of the protocol itself and it is employed in the protocol layer, not only in the user interface layer. So it is not a software solution.

We assume that any ballot **B** contains *n* candidates: **B** = {C_1, C_2, ... , C_n}, C_i representing a different candidate for each dynamically generated ballot. For *n* candidates, voters may take '*n!*' different ballots.

An example set of dynamic ballots for four candidates can be as follows:

$$\mathbf{B_1} = \{C_2, C_1, C_4, C_3\},\ \mathbf{B_2} = \{C_1, C_2, C_3, C_4\}$$
$$\mathbf{B_3} = \{C_4, C_1, C_3, C_2\},\ \mathbf{B_4} = \{C_3, C_2, C_1, C_4\} \quad (1)$$
$$\mathbf{B_5} = \{C_2, C_1, C_4, C_3\}$$

Therefore, counting authorities cannot count intermediate results and furthermore voters do not have to involve more than one round. With dynamic ballots, any participant or authority including the counter cannot gain any knowledge about the tally before the counting stage. In fact this assures the fairness of the protocol. Voting stage consists of 2 phases: Ballot obtaining phase and vote casting phase. Overview of voting stage is shown in Figure 1.

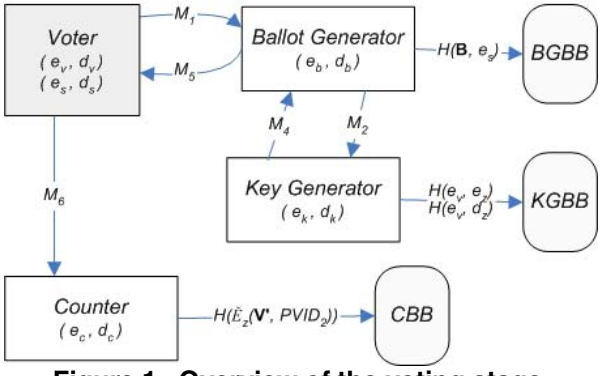

Figure 1. Overview of the voting stage.

3.2.1. Ballot Obtaining Phase. Voter creates session public-private key pairs (e_s, d_s) and (e_v, d_v). The former is used for Ballot Generator; the latter is used for Key Generator. Voter employs these keys in order to obtain dynamic ballot and voting key. Voter encrypts e_v and election date with Key Generator's public key and

produces $\check{E}_k(e_v, \text{ElectionDate})$. Election date is used to make the message more meaningful for Key Generator and to be easily identified by Key Generator. Then, voter creates the message M_1:

$$M_1 = \check{E}_b(\text{PVID}_1, \check{E}_k(e_v, \text{ElectionDate}), e_s)$$

Voter sends M_1 to Ballot Generator. As soon as receiving the message M_1, Ballot Generator decrypts it. Ballot Generator checks the PVID_1 by applying PVID Authority's public key. If the check fails, Ballot Generator discards the message. If it succeeds, Ballot Generator signs $\check{E}_k(e_v, \text{ElectionDate})$ and then generates the message M_2:

$$M_2 = \check{E}_k(\check{D}_b(\check{E}_k(e_v, \text{ElectionDate}), e_b))$$

Ballot Generator's public key, e_b, is also encrypted inside the message body in order to identify any message corruption. Instead of e_b, any pre-defined value can be used. For this message and the further messages, we prefer to use public keys. Ballot Generator sends the message M_2 to Key Generator. Key Generator decrypts the message M_2 and checks Ballot Generator's signature on it. If it is a valid message, Key Generator proceeds to further steps. Key Generator creates a voting key pair (e_z, d_z). Voting keys are used by the voters to cast their candidate selections to Counter. Key Generator saves generated key pair (e_z, d_z) in VotingKeyList, which is an internal list of voting keys. It publishes hash of voter's public key with voting key's public one and private one separately as $H(e_v, e_z)$ and $H(e_v, d_z)$ in Key Generator's Bulletin Board (KGBB). $H(e_v, e_z)$ is used by the voter to verify the correctness of the voting key and $H(e_v, d_z)$ is used by Counter to prevent Key Generator's manipulation on the generated voting keys. Key Generator saves (e_v, e_z, d_z) in VotingKeyList and generates M_3 and M_4:

$$M_3 = \check{E}_v(\check{D}_k(e_z, \text{ElectionDate}), e_v)$$
$$M_4 = \check{E}_b(\check{D}_k(M_3, e_k))$$

Key Generator sends M_4 to Ballot Generator. Ballot Generator decrypts the message and checks Key Generator's signature. Afterwards Ballot Generator creates a dynamic ballot **B** by using a random number generator function. Then it publishes the hash of dynamic ballot **B** and voter's session public key e_s which is $H(\mathbf{B}, e_s)$ in Ballot Generator's Bulletin Board (BGBB). $H(\mathbf{B}, e_s)$ is published to give an opportunity to the voter to verify the correctness of dynamic ballot. Ballot Generator saves the $(\text{PVID}_1, M_3, \mathbf{B}, e_s)$ in BallotList, which is an internal list of dynamic ballots. Then it produces M_5:

$$M_5 = \check{E}_s(\check{D}_b(M_3, \mathbf{B}, e_b))$$

Ballot Generator sends M_5 to the voter. Voter decrypts the received message by applying Ballot Generator's public key and extracts M_3 and dynamic ballot **B**. In order to verify the obtained dynamic ballot, voter calculates $H(\mathbf{B}, e_s)$ and checks against the BGBB.

Later, voter decrypts the message M_3 and applies Key Generator's public key in order to extract voting key e_z. Voter creates $H(e_v, e_z)$ and verifies the result against the KGBB. At this point voter has dynamic ballot **B** and voting key e_z; and he is ready to carry out vote casting.

3.2.2. Vote Casting Phase. Voter selects his candidate and creates his candidate selection **V'** using the dynamic ballot **B**. Voter encrypts **V'** with voting key e_z. Then he constructs the message M_6:

$$M_6 = \check{E}_c(\text{PVID}_1, \check{E}_z(\mathbf{V'}, \text{PVID}_2), e_v)$$

Voter sends M_6 to Counter, in other words voter casts his vote. There is no need to any anonymous communication channel to cast vote since PVID scheme is employed. Nobody can make any relation between voter's real registration identity and PVIDs due to the definition of PVID scheme. Hence voter can easily send **V'** as well as PVID. **V'** is voter's candidate selection in the dynamic ballot. So, it has a contextual meaning depending on the ordering of candidates in the dynamic ballot **B**. For example, the following candidate selections may be done by voters for the given sample ballot set in equation (1):

$$\mathbf{V_1'} = 2, \mathbf{V_2'} = 2, \mathbf{V_3'} = 3, \mathbf{V_4'} = 3, \mathbf{V_5'} = 3 \qquad (2)$$

Counter decrypts the message M_6 and extracts PVID_1 as well as encrypted candidate selection $\check{E}_z(\mathbf{V'}, \text{PVID}_2)$. Counter performs PVID Authority's public key on PVID_1 to check the validity of PVID_1. If it is valid, Counter processes the request; else discards the message. Counter creates the hash of encrypted **V'** as $H(\check{E}_z(\mathbf{V'}, \text{PVID}_2))$ and publishes it in Counter's Bulletin Board (CBB). Counter saves encrypted **V'** by appending the date and time of it to VoteList as $(\text{PVID}_1, \check{E}_z(\mathbf{V'}, \text{PVID}_2), e_v, \text{DateTime})$. VoteList is an internal list of voters' candidate selections associated with PVIDs. Later Counter sends an acknowledgement message $\check{E}_v(\check{D}_c(\mathbf{Ack}))$ to the voter in order to inform him. As soon as receiving the **Ack**, the voter checks the CBB to verify individually his vote. Voter finds the sequence number of $H(\check{E}_z(\mathbf{V'}, \text{PVID}_2))$ in CBB and keeps the sequence number as a receipt. Then the voter's voting session is over.

3.3. Counting Stage

Counting stage is performed after the election period has been completed. During the election period, Ballot Generator, Key Generator, and Counter publish hash of subsets of relevant information on bulletin boards. Before proceeding the counting of votes, Ballot Generator, Key Generator, and Counter announce the SubBallotList ($|PVID_1, \mathbf{B}|$), SubVotingKeyList ($|e_v, d_z|$), and SubVoteList ($|\check{E}_z(\mathbf{V'}, PVID_2)|$) respectively.

Counter compares the sublists against the hash values in bulletin boards. Any passive observer or organization can also check the consistency of the election by using announced lists and bulletin boards.

Then Counter starts counting. Firstly, it matches each item in VoteList $|PVID_1, \check{E}_z(\mathbf{V'}, PVID_2), e_v,$ DateTime| with corresponding items in SubVotingKeyList $|e_v, d_z|$ over voter's session key e_v. Afterwards Counter obtains a list $|PVID_1, \check{E}_z(\mathbf{V'}, PVID_2), e_v,$ DateTime, $d_z|$.

Counter simplifies the list by decrypting the encrypted candidate selections ($\check{E}_z(\mathbf{V'}, PVID_2)$) with the corresponding private keys (d_z) and produces the list $|PVID_1, PVID_2, \mathbf{V'}|$ which is voters' candidate selections. Counter checks the $PVID_2$ by applying PVID Authority's public key. If the check fails, Counter discards the vote. Overview of counting stage is shown in Figure 2.

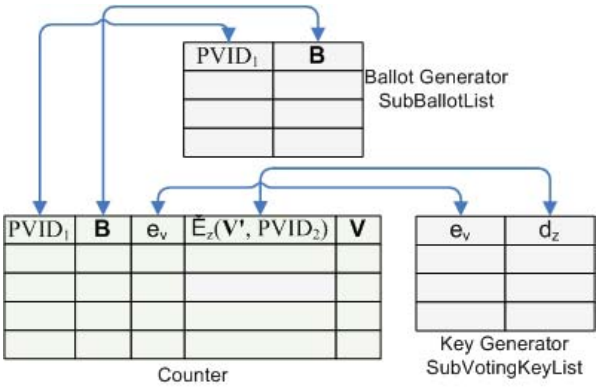

Figure 2. Overview of counting stage.

Since PVID scheme is employed and the proposed protocol allows recasting, voter can vote several times. Date and time of each casting are kept by Counter. Only the latest cast is taken into consideration.

Later, Counter matches the candidate selections (**V'**) in the list $|PVID_1, PVID_2, \mathbf{V'}|$ with corresponding dynamic ballots (**B**) in SubBallotList $|PVID_1, \mathbf{B}|$ over $PVID_1$. Then, Counter obtains a list $|PVID_1, PVID_2, \mathbf{V'}, \mathbf{B}|$ which is in fact the list of voters' actual votes. An actual vote **V** is defined as:

$$\mathbf{V} = C_i \in \mathbf{B} \qquad \text{where } i = \mathbf{V'},\ \mathbf{B} = \{C_1, C_2, \ldots, C_n\}$$

For the given sample ballot set in (1) and sample candidate selection set in (2) the election result becomes as in Table 1. Thus, the final tally is: $\mathbf{C_1} = 2$ votes, $\mathbf{C_2} = 1$ vote, $\mathbf{C_3} = 1$ vote, $\mathbf{C_4} = 1$ vote.

Table 1. A sample election result.

B	V'	V
B1 = {C2, C1, C4, C3}	**V1'** = 2	**V1** = C1
B2 = {C1, C2, C3, C4}	**V2'** = 2	**V2** = C2
B3 = {C4, C1, C3, C2}	**V3'** = 3	**V3** = C3
B4 = {C3, C2, C1, C4}	**V4'** = 3	**V4** = C1
B5 = {C2, C1, C4, C3}	**V5'** = 3	**V5** = C4

At the end of the counting stage, Counter announces the list of $|PVID_1, H(\check{E}_z(\mathbf{V'}, PVID_2)), \mathbf{V}|$ in consistent with the order in the CBB as well as remarking the discarded votes. Now votes are easily tallied and the election result is announced.

One of the major contributions of this paper is to give an opportunity to voter to perform individual verifiability while casting his vote without revealing his identity. In each stage voter can check and individually verify intermediate outcomes against bulletin boards. In case of any corruption he can make objection. After counting stage has been completed voter can individually verify his vote with his $PVID_1$ and the receipt of sequence number of $H(\check{E}_z(\mathbf{V'}, PVID_2))$ by using the announced lists.

4. Security Analysis

We provide the sketch of proofs that state how the DynaVote protocol fulfills the e-voting requirements.

Lemma 1 *(Privacy)*: A particular voter and his cast vote is unlinkable.

Sketch of Proof: PVID Authority issues blind signature on voter's blinded IDs after checking voter's eligibility. Since the blind signature scheme is used, any particular registration ID is not linkable to any PVID and any particular PVID is not linkable to any registration ID. Voter does not use his registration ID after obtaining PVID; instead he uses his PVID in next stages. Therefore privacy is assured.

Lemma 2 *(Eligibility)*: Only eligible and authorized voters can vote.

Sketch of Proof: We employ PVID scheme which guarantees that only eligible voters can obtain valid PVIDs. PVID Authority issues blind signature on voter's blinded IDs after checking voter's eligibility. Only eligible voters' blinded IDs are blindly signed by PVID Authority. Ineligible people's blinded IDs

cannot be signed without being detected since threshold cryptography is applied to distribute the authority over *n* parties. In order to sign any request at least *t* parties should come together. Therefore authentication and authorization are fulfilled by PVID scheme. In the proposed protocol voter can vote multiple times, just the latest one is counted, the rest are discarded. Thus, the proposed protocol achieves eligibility requirement.

Lemma 3 *(Uniqueness)*: Only one vote for a voter is counted.

Sketch of Proof: In counting stage, Counter obtains a final list $|PVID_1, PVID_2, \mathbf{V'}, \mathbf{B}|$. $PVID_1$ is the primary key for this list and is unique. Voter can recast, however the last vote is taken into consideration and the previous ones are discarded. Thus, there is no PVID duplication in the list. Since the $PVID_1$ is unique in the list and can be verifiable using PVID Authority's public key, there is no chance that more than one vote is counted for any voter. Therefore, uniqueness is achieved.

Lemma 4 *(Uncoercibility)*: Voter cannot be coerced to cast his vote in a particular way.

Sketch of Proof: The proposed protocol allows recasting. Even if someone coerces voter, voter casts by that way. Later, he can change his vote, by recasting new one and then the old one is discarded in counting stage. Same logic can be applied to vote selling. So, practically it is not possible to coerce voter or to buy vote from voter, since nobody can know whether the current vote will be the final one. Therefore, uncoercibility is achieved.

Lemma 5 *(Fairness)*: No partial tally is revealed before the end of the voting period.

Sketch of Proof: Counting comes after the voting stage is completed so no one can gain any partial knowledge about the tally before the counting stage; as a consequence, voting is not effected. Since we are employing dynamic ballots, Counter just knows voter's candidate selection which does not reveal any information without ballot. Even if Ballot Generator provides Counter the corresponding dynamic ballot **B**, Counter could not extract the voter's cast vote since the voting key, which is maintained by Key Generator, is required. Thus, Counter could not calculate any partial result. So, this requirement is achieved.

Lemma 6 *(Accuracy)*: Any vote cannot be added, altered, deleted, invalidated or copied in the final tally without being detected.

Sketch of Proof: During the voting stage, voter verifies each step before proceeding to next one. When he obtains dynamic ballot **B** and voting key e_z, he checks KGBB and BGBB; in case of corruption he can object to Ballot Generator. After voting, he also verifies CBB to assure that his vote is listed. The detailed explanation is given in individual verifiability requirement analysis.

Voting and counting authorities have bulletin boards and they publish all relevant information in them. Counter counts votes using the sublists provided by Ballot Generator and Key Generator. For consistency, Counter compares the sublists against the hash values in bulletin boards.

Any single authority cannot alter, delete, invalidate or copy any vote since the modification causes inconsistency with the bulletin boards. Moreover, voter verifies his vote and makes objection. Any single authority cannot add any vote since a vote consists of a dynamic ballot **B** and a voting key e_z. Even if Ballot Generator, Key Generator and Counter conspire together, they cannot add a new vote since they cannot create fake PVIDs. PVID Authority cannot issue fake PVIDs since threshold cryptography is applied. So, accuracy is achieved.

Lemma 7 *(Robustness)*: Any coalition of voters or authorities cannot disrupt the voting or influence the election and final tally.

Sketch of Proof: The dishonest voter cannot disrupt the voting, he has just right over his vote, so he may only disrupt his vote. Even if he sends more than one votes, in this case, just last one is counted since $PVID_1$ is unique. Voter is aware of that his previously sent votes will be discarded if he sends more than one vote.

As bulletin boards are employed, hash of all information related with voter's vote is recorded publicly. Thus, any authority corruption can be revealed. If any authority conspires with voter, they can just corrupt that voter's vote. Therefore, this requirement is achieved.

Lemma 8 *(Individual Verifiability)*: Each eligible voter can verify that his vote is counted correctly.

Sketch of Proof: Key Generator publishes $H(e_v, e_z)$ in KGBB. $H(\mathbf{B}, e_s)$ is published in BGBB. Voter attempts to create same hash values by using dynamic ballot **B**, voting key e_z and his session keys e_v and e_s. If he obtains same values, he proceeds to send his candidate selection to Counter.

In ballot obtaining phase, if voter receives corrupted voting key e_z, he could not generate proper $H(e_v, e_z)$. He can object this situation by showing (e_v, e_z) and $H(e_v, e_z)$. If voter does not receive proper dynamic ballot **B**, he can prove that the dynamic ballot does not match with the hash values published in BGBB by showing $(\mathbf{B}, e_s)$ and $H(\mathbf{B}, e_s)$. Therefore Ballot Generator and Key Generator are required to respond to voter properly. Otherwise, voter can easily prove any improper responses.

In vote casting phase, voter checks CBB as soon as receiving the acknowledgement from Counter by creating same hash value for **V'** as $H(\check{E}_z(\mathbf{V'}, PVID_2))$.

If the value does not match, he can object to Counter by illustrating **V'** and e_z. Thus, Counter could not modify the voter's candidate selection **V'**. Voter can verify his vote by checking his $PVID_1$ and the sequence number of $H(\check{E}_z(\mathbf{V}', PVID_2))$ by using the announced lists. Therefore, individual verifiability is fulfilled.

Lemma 9 *(Universal Verifiability)*: Any participant or passive observer can verify that the published tally is correctly computed from correctly cast votes.

Sketch of Proof: At the end of the election, before counting, all authorities publish their sublists. As soon as Counter announces the election result all authorities publish complete lists. Besides, bulletin boards are employed. So any participant or passive observer can check whether votes are counted correctly or not. Counter has responsibility to verify all results and publish them. So, universal verifiability is fulfilled.

5. Discussion

DynaVote has one shortcoming which appears when Ballot Generator, Key Generator and Counter conspire and work together. They can modify cast votes due to the recasting. However, voter can verify his vote on the Counter's published list and make an objection providing his PVIDs. It does not violate voter's privacy since PVIDs are pseudo identities which are unlinkable with real registration ID. Currently, we are studying to find a proper solution for this specific issue.

In this paper, we just mentioned the core e-voting requirements and illustrated how DynaVote fully satisfies them. Besides, there are some desirable requirements. DynaVote also satisfies many of them such as dispute-freeness, scalability, efficiency, and mobility. As a future work, we will describe in detail how DynaVote achieves these requirements as well. In addition, we are planning to implement DynaVote as a proof of concept.

6. References

[1] A. Fujioka, T. Okamoto, and K. Ohta, "A practical secret voting scheme for large scale elections," *AUSCRYPT'92*, Australia, pp. 244-251, 1992.

[2] L. Cranor, and R. Cytron: "Sensus: A security-conscious electronic polling system for the Internet," *Hawaii Int. Conf. on System Sciences,* Hawaii, 1997.

[3] O. Cetinkaya, and Ali Doganaksoy, "A practical privacy preserving e-voting protocol using dynamic ballots," *2nd National Cryptology Symposium,* Ankara, Turkey, 2006.

[4] O. Cetinkaya, and A. Doganaksoy, "PVID: Pseudo-voter identity scheme for e-voting protocols," *First Int. Workshop on Advances in Information Security,* Vienna, Austria, 2007.

[5] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of ACM*, Vol. 24, pp. 84-88, 1981.

[6] A. Baraani, J. Pieprzyk, and R. Safavi, "A practical electronic voting protocol using threshold schemes," *Centre for Computer Security Research*, University of Wollongong, Australia, 1994.

[7] T. Okamoto, "Receipt-free electronic voting schemes for large scale elections," *5th Security Protocols Workshop*, LNCS 1163, Springer-Verlag, pp. 125-132, 1997.

[8] J. Benaloh, and D. Tuinstra, "Receipt-free secret-ballot elections," *Proc. of the 26th ACM Symp. on the Theory of Computing*, 544-553, 1994.

[9] R. Sampigethaya, and R. Poovendran, "A framework and taxonomy for comparison of electronic voting schemes," *Elsevier Computers & Security*, Vol. 25, No. 2, pp. 137-153, 2006.

[10] D. Chaum, P. Y. A. Ryan, and S. Schneider, "A practical, voter-verifiable election scheme," *ESORICS'05*, Milan, Italy, pp. 118-139, 2005.

[11] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," *EUROCRYPT'97*, Germany, 1997.

[12] A. Acquisti, "Receipt-free homomorphic elections and write-in voter verified ballots," *ISRI Technical Report CMU-ISRI-04-116,* Carnegie Mellon Uni., PA, 2004.

[13] M. Hirt, and K. Sako, "Efficient receipt-free voting based on homomorphic encryption", *EUROCRYPT'00*, Bruges, Belgium, pp. 539-556, 2000.

[14] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections", *ACM Workshop on Privacy in the Electronic Society*, VA, pp. 61-70, 2005.

[15] Y. Mu, and V. Varadharajan, "Anonymous secure e-voting over a network", *14th Annual Computer Security Applications Conference*, AZ, pp. 293-299, 1998.

[16] I. Ray, I. Ray and N. Narasimhamurthi, "An anonymous electronic voting protocol for voting over the Internet", *3rd Int. Workshop on Advanced Issues of E-Commerce and Web-based Information Systems*, CA, 2001.

[17] C. C. Yang, C. Y. Lin, and H. W. Yang, "Improved anonymous secure e-voting over a network", *Information & Security*, Vol. 15-2, pp.181-194, 2004.