IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 2, Ver. 1 (Mar – Apr. 2015), PP 89-102
www.iosrjournals.org

Performance Evaluation of the Bingo Electronic Voting Protocol

Waleed A. Naji, Sherif Khattab and Fatma A. Omara
Department of Computer Science, Faculty of Computers and Information
Cairo University, Egypt

Abstract: *Research in e-voting aims at designing usable and secure electronic voting systems. This paper provides an empirical analysis of the computational performance of a prototype implementation of the Bingo electronic voting protocol. Bingo is a receipt-based end-to-end verifiable electronic voting protocol that claims the property of coercion resistance. According to this work, a prototype of the Bingo design has been described in terms of sequence and class diagrams. Also, its operation has been demonstrated using a case study of a sample election. Four main operations have been analyzed; initialization of cyclic groups, generation of dummy votes, zero-knowledge proof of fair vote distribution over candidates, and zero-knowledge proof of receipt correctness. The performance was affected by the cyclic group order, number of candidates, and number of voters.*

Keywords: *E-voting, Bingo voting, coercion resistance, zero-knowledge proof, commitments.*

I. Introduction

Voting plays an essential role in a democracy. The result of voting determines the future of a country. Voting must achieve a set of requirements. On the other hands, an e-voting system must achieve technical requirements, user interaction requirements, integration requirements, and (most critically) security requirements [1-4].

Security requirements of the e-voting system are classified into two group; voter-related requirements and voting-related requirements. Voter-related requirements include authentication, where voting process is accessible only by the voters who identify themselves; privacy (i.e., no one except the voter himself has the ability to determine who has been) voted; anonymity (i.e., no one can link or extract hints between voters and their votes); coercion-resistance, where each voter should be able to cast to her candidate without the influence of a coercer and voters cannot prove to the coercer or others their cast, even if the voter wanted to cooperate with the coercer; and individual verification, where each voter has the capability to guarantee that his vote has been cast for the selected candidate. Coercion-resistance is one important security requirement in the e-voting system.

Voting-related requirements include ballot box integrity, only registered voters can vote and after the voter cast no one should be able to modify, delete or detect the voter choice; tally accuracy, votes must be gathered and recorded in a correct way; fairness, no one should be able to know any information about the tally (voting) result before the official announcement of the tally; and auditability, the e-voting system allows for a third party to verify the voting process and that the final tally was calculated correctly.

Bingo is, an end-to-end verifiable electronic voting protocol [5]. It depends on a trusted random number generator (TRNG) for correctness and provides the voter with a receipt that the voter can use it for verification, but the voter cannot use this receipt to prove his choice. The work in this paper describes the design and implementation of a prototype of Bingo voting system. A case study of a simple election with ten voters and three candidates will be described to verify the system. Finally, empirical analysis of the computational performance of the Bingo voting implementation is described.

The effect of the cyclic group order, number of voters, and number of candidates has been studied. The cyclic group order was the crucial factor in the performance of Bingo voting system because the computation of the commitment depends on it. When the cyclic group order increases, the speed of computing a commitment decreases. Also, the number of voters and candidates play an important role because the number of commitments depends on the number of voters and candidates.

The paper is organized as follows; Section II describes the background of cryptographic schemes, techniques and tools used in Bingo voting system. In Section III, we the Bingo voting protocol is briefly described. Section IV describes the sequence and class diagrams of the implementation prototype of Bingo voting system and highlights the implementation details. Section V presents a case study of a simple election system using the proposed prototype and measured performance of the main operations of Bingo voting system. Finally, Section VI concludes the paper and briefly presents future work.

II. Background

E-voting protocols use many cryptographic schemes, techniques and tools (e.g. commitments and zero knowledge proofs) to provide the security requirements. This section describes commitments, zero knowledge proofs, mix-nets, efficient proofs of a shuffle of known content, and bulletin board, the cryptographic tools used in the Bingo voting protocol.

2.1 Cyclic Groups

A group $\mathbb{G}$ is a cyclic group if there is an element $g \in \mathbb{G}$, such that for each $b \in \mathbb{G}$ there exists an integer i such that $b = g^i$. Here, g is called a generator of the cyclic group $\mathbb{G}$ and the number of elements in $\mathbb{G}$ is called group order $|\mathbb{G}|$[5, 6]. In more details, if $a \in \mathbb{G}$, the set of all powers of a forms a cyclic group $\subseteq \mathbb{G}$ called subgroup ‹a›, which is generated by a. The number of elements in ‹a› is called the order of element a, which is calculated as the smallest value j such that $a^j \ mod\ |\mathbb{G}| = 1$. So, if $j = |\mathbb{G}|$, then a is a generator of the cyclic group $\mathbb{G}$. For example, $\mathbb{Z}_{19} = \{1, 2, 3 \ldots, 18\}$ has a group order $|\mathbb{G}| = 18$. So, if any element in this group has $order = 18$, that means this element is a generator. To calculate the order of any element, such as 7, the equation $7^j \bmod 19 = 1$ is applied. Here, the minimum is $j = 3$. So, 7 is not a generator of the cyclic group $\mathbb{Z}_{19}$. In the same way, by calculating the order for each element $a \in \mathbb{G}$, if $order = 18$, then this element is a generator of $\mathbb{Z}_{19}$. Table 1 shows the elements in $\mathbb{Z}_{19}$, element subgroups, element orders, and generators of $\mathbb{Z}_{19}$.

Table1: Elements, subgroups, orders and generators of $\mathbb{Z}_{19}$

Elements	Element subgroup	Element order
1	{2}	1
18	{1,18}	2
7,11	{1,7,11}	3
8,12	{1,7,8,11,12,18}	6
4,5,6,9,16,17	{1,4,5,6,7,9,11,16,17}	9
2,3,10,13,14,15 (generators)	{1,2,3,4,5, ….,18}	18

2.2 Commitments

Commitment is a cryptographic scheme used to temporarily hide a value and ensure that anyone cannot change it [7, 8]. The commitment scheme has two properties, a binding property, which ensures that the commitment contains only one value, and a reveal property, which guarantees that the receiver cannot know the value as long as he does not have the reveal information. We can say the commitment is perfectly hiding, because commitment opening is impossible without knowledge of revealing information r.

In e-voting schemes that depend on the commitment, perfect hiding is required to hide the voter choices to satisfy voters' privacy. And, the binding property is needed to prove the correctness of the tally (voter be convinced that her vote did not change). Pedersen commitment is one of the used commitment types in the e-voting schemes [9].Pedersen commitment scheme is computationally binding and unconditionally hiding. Pedersen commitment satisfies the binding and hiding properties if the discrete logarithm problem is hard. The Pederson commitment consists of three algorithms: setup, commit and reveal[9].

Pederson Setup: Choose two large safe primes p and q where $p = 2q + 1$. Then, create a cyclic group $\mathbb{G}$ with order p. Then choose a random secret $a \in \mathbb{Z}_q$ as input. In addition, choose two generators g and h where $h = g^a$ and $g, h \in \mathbb{Z}_p$. Finally, the output (public info) is $p, q,\ g$ and h.

Pederson Commit (m, r): Take a message m that one wants to commit to and choose r at random $(m, r \in \mathbb{Z}_q)$ and the output is commitment c, where $c = g^m h^r mod\ p$.

Pederson Reveal (c, m, r): Receiver calculates and verifies that $c = g^m h^r \bmod p$.

Also, Pedersen commitments have the multiplicatively homomorphic property, which allows to mask or re-randomize a commitment c without knowledge of c's content. $c` \coloneqq c * h^{r`} = g^m h^r * h^{r`} = g^m h^{(r+r`)}$ In this case $c`$ is $commit(m, (r` + r))$. So, the reveal information is $(r` + r)$. The homomorphic property is useful in e-voting to keep the tallying secret by masking a set of $c's$ to obtain a set of $c`s$ and then reveal $(c`, m, r + r`)$. Here, the link between c and m is difficult because r is unknown.

2.3 Zero-Knowledge Proofs (ZKP)

ZKP [10, 11] is an interactive proof system between a prover, P, and a verifier, V. P wants prove to V that a statement is correct without revealing the secret information. For example, in authentication, the verifier system should not know the password of the user. At the same time, the verifier must ensure that the password is correct to allow P (i.e., the user) to log in to the system. ZKP has three properties; completeness (an honest prover is capable to convince an honest verifier that the statement is correct), soundness (a dishonest prover that does not have a secret cannot convince an honest verifier that the statement is true), and zero-knowledge (the proof does not leak any secret information).

E-voting schemes that use ZKP exploit the zero-knowledge property to satisfy coercion resistance in the e-voting system because in the ZKP the verifier ensures that statements are correct (her vote has been counted), but she cannot get the secret information to show that to anyone.

2.4 Mix-nets

Mix-nets have been proposed by Chaum[12]. A mix-net is a building block used for anonymity. So, mix-nets are very important in the e-voting protocols to satisfy the anonymity requirement. By shuffling a set of inputs using many servers (called mix servers), every server receives a set of input elements M and shuffles them by changing the position of every element to anonymize them, then send the output M' (where $shuffle\,(M) \rightarrow M')$ to the next server as input. $M = \{m_1, m_2, \ldots\ldots, m_n\}$ is the input and $M' = \{\ m'_{\pi(1)}, .., m'_{\pi(n)}\ \}$ is the output, where π is a random permutation.

For the e-voting schemes, shuffles must achieve two properties; secrecy (it is infeasible to establish a link between input (m_1) and output $(m'_{\pi(1)})$), and verifiability (there exists a proof that M is indeed a permutation of M' such that for each $m' \in M'$ there exists only one value $m_i \in M$ and vice versa).Since the permutation in a mix-net remains hidden, the correctness of shuffling must be proved by zero-knowledge proofs, called shuffle proofs.

Shuffle proofs can be done by several ways, such as the efficient shuffle proof (ESP) proposed by Neff [13]. The ESP depends on the invariance of polynomials under the permutation of their roots: $(\prod_{i=1}^{n}(m_i - x) = \prod_{i=1}^{n}(m_{\pi(i)} - x))$ where $x \in \mathbb{Z}_q$ and x is the random value chosen by the verifier. If polynomials are equal, then the mixing result is correct. The shuffle proofs have been improved by Groth [14, 15].

Groth proposed a more efficient proof of a shuffle of known content by generating the interactive ZKP protocol between the prover P and verifier V. This proof is used to prove the contents of a set of commitments $\mathrm{C} = \{\mathrm{c}_1, \mathrm{c}_2, \ldots\ \mathrm{c}_\mathrm{n}\} \subset \mathbb{Z}_\mathrm{p}$ and each c contains only one message from $M = \{m_1, m_2, \ldots\ m_n\} \subset \mathbb{Z}_q$ without revealing which commitment contains which message. The protocol inputs are divided into common input, which is the set of commitments C and the set of messages M and secret inputs private to P, which are the set of random numbers $(r_1, r_2, \ldots\ r_n)$ and permutation π, where $c_i = commit(m_{\pi(i)}, r_i)$ and i ranges from $1\,..\,n$. The protocol operates as follows:

V chooses at random $x \in \mathbb{Z}_q$ and sends x to P.Then, P and V calculate $\hat{c} = commit(x, 0)$and perform multiplication proof steps. In the first step P calculates $p_i = \prod_{k=1}^{i}(m_{\pi(k)} - x)$ where $i = 1..n$ and $c_{pi} = commit(p_i, r_{pi})$ where $r_{pi} \in \mathbb{Z}_q$ is chosen randomly except $r_{pn} = 0$. In the second step, for $i = 2\,..\,n$ then P calculates and sends $c_{ai} = commit(a_i, r_{ai})$ and $c_{bi} = commit\left(\left(m_{\pi(i)} - x\right) * a_i, r_{bi}\right)$ where a_i, r_{ai} and r_{bi} are chosen at random. In the third step, V chooses d_i at random and sends it. In the fourth step, P calculates and sends $e_i = p_{i-1} * d_i + ra_i, r_{ei} = r_{pi-1} * d_i + ra_i$, and $f_i = r_i(p_{i-1} * d_i + r_{ai}) - r_{pi} * d_i - r_{bi}$. Finally, V verifies that $h^f = \frac{\hat{c}^{ei}}{(c_{pi})^{di} * c_{bi}}$and $c_{pn} = g^{\prod(m_i - x)}$.

2.5 Bulletin board

A bulletin board is a cryptographic building block proposed by Cohen[16-18]. It is a one way secure asynchronous broadcast to display the messages from one party to another. Every published message cannot change or get deleted and is provided with a time stamp. The e-voting schemes used the bulletin board to display the election information such as candidates' list and election results.

III. Bingo Voting

Bingo voting is an end-to-end (E2E) voting scheme proposed by Bohli et al [19-21]. Bingo voting depends on a trusted random number generator TRNG in correctness and uses some cryptographic techniques and tools (e.g., commitments, zero knowledge proofs and bulletin boards).Bingo voting consists of three phases; pre-voting phase, voting phase and post-voting phase.

Suppose that where j is the number of eligible voters. For every candidate A_i, the voting authority generates j random numbers N_i where $i = 1..j$. Then, these numbers are committed, shuffled, and published on the bulletin board as dummy votes (DV), which are pairs of commitments. The first commitment is of the random number and the second is of candidate ID, that is, $commit(A_i)$. So, every $DV =< commit(N_j), commit(A_i) >$. Also, the proof that every candidate has the same number of dummy votes is published. It is done by opening the second commitment using a zero knowledge proof[11].

In the voting phase, as in traditional elections, voter's eligibility is checked by the voting authority, and then the voter enters to the voting booth and casts for intended candidate by pressing a button corresponding to the candidate. The TRNG generates a fresh number and the voter visually verifies that the fresh number appears

next to the selected candidate. Then, the voting machine prints a receipt, which consists of a fresh number besides the candidate whom the voter has chosen. Each other candidate gets a random number from their unused dummy votes and these dummy votes are marked as used.

The post-voting phase starts when the election is finished. The voting authority publishes a list of digital copies of all receipts issued and a list of all unused dummy-votes with the respective commitment and reveal information. A non-interactive zero-knowledge proof proves the correctness. Here, every voter can check if her receipt was published on the bulletin board correctly, and if the tally result is done correctly.

IV. Bingo Implementation

This section describes the design and implementation of Bingo prototype.

4.1 Bingo Voting Design

This subsection describes the sequence and class diagrams of Bingo prototype design. Fig. 1 depicts the sequence diagram of the pre-voting phase. The committee member represents the election authority. The committee represents the software component that receives a command from election authority and executes it. The database represents the component that manages a secure database. The commitment manager is the component that computes a commitment of a message. A bulletin board represents the component used as a channel to communicate with auditors. The voting machine component represents the component that controls the voting machine. The TRNG component represents the interface that connects with TRNG. The election auditor is any party who wants to check the correctness of the Bingo voting. And, the verifier component is used to check the correctness of the Bingo voting.

According to Fig. 1, the supreme committee members for elections commands the committee component to generate the dummy votes. Then, the committee component inquires about the number of voters and the number of candidates from the database. It is assumed that this information has been entered previously. In addition, the committee component generates N dummy votes, where N is equal to the number of voters * the number of candidates. Then, the commitment for each DV is computed by the commitment manager and stored in the database. The committee member is then notified. After that, the committee member publishes the dummy votes by using the committee component, which queries all DVs from the database and then displays them on the bulletin board.

Fig. 2 shows that the election committee has a sub-committee supervising the elections at each polling location to verify the voter's eligibility and that the voter has not voted before. Then, it allows the voter to enter the voting booth to vote and mark in front of the voter's name that he has voted to preventing him from double voting. After that, the committee member signals to the committee to start the vote and the committee notifies the voting machine (VM) to start voting. When the voter enters the voting booth, he sees the VM, TRNG and printer. The voter votes by using the VM, which displays the list of candidates. The VM receives the voter's choice and then asks from the TRNG for a new fresh number (FN), which represents the voter's vote. The TRNG generates FN and displays FN on its screen and sends the FN to the VM. The VM creates the receipt and prints it, of which the voter takes a copy. Finally, the voter goes out of voting booth, and this process is repeated for all voters.

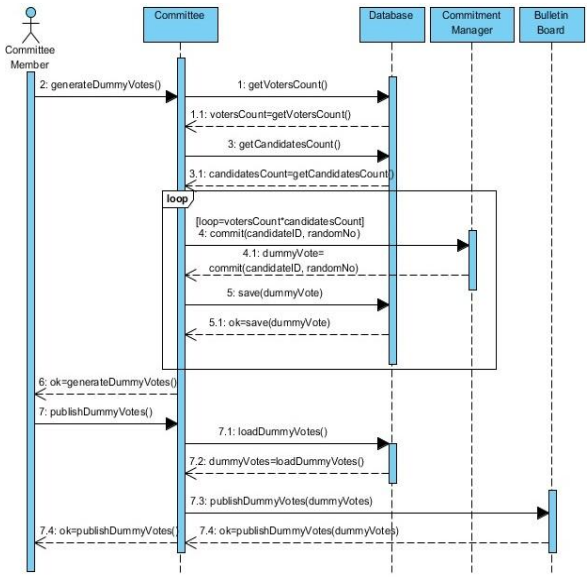


Fig. 1: Pre-voting phase sequence diagram

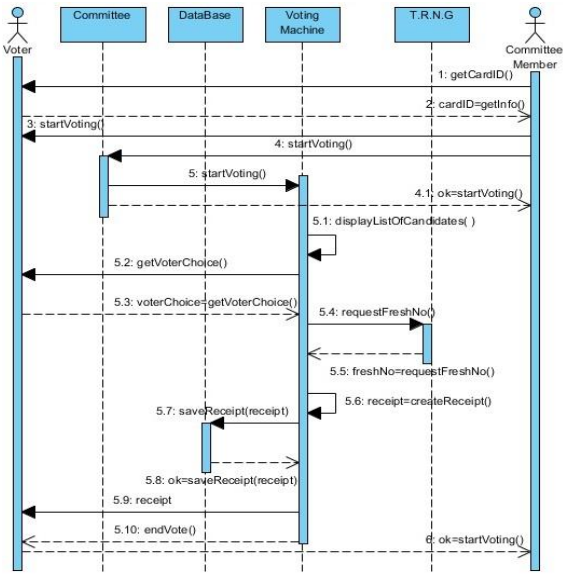


Fig. 2: Voting phase sequence diagram

Fig. 3 shows that the committee member commands committee component to calculate the result, and then the committee inquires about the unused dummy votes from the database. Then, committee component calculates how many each candidate has remaining of his dummy votes (the result of the elections). Then, the result is published on the bulletin board and the committee member is notified that the result was published successfully. The committee member requests to display reveal information or proof of correctness. The committee inquires the database for the secret information used to open the commitments. Also, the committee component inquires for all receipts and publishes the reveal information and all receipts. Then, it notifies the committee member that the reveal information has been published successfully.

Fig. 4 shows that the election auditor, which represents the person who wants to verify the contents of a receipt through the verify component, enters a receipt number to the verify component to the committee component and inquires about the commitments of this receipt from the committee. The committee component gets the commitments from the database and returns the commitments to the verify component, which submits it to the election auditor, who can check if the commitments exist on the bulletin board. After that, the election auditor starts verifying by using the verify component, which generates a random challenge sent to the committee. The committee requests secret information in this receipt and calculates a proof which proves that the commitments contain the receipt data. This is done by Groth shuffle proof. Finally, it sends proof Information to the verify component, which in turn verifying if calculations are done correctly. Then it notifies the election auditor of the result of the verification process.

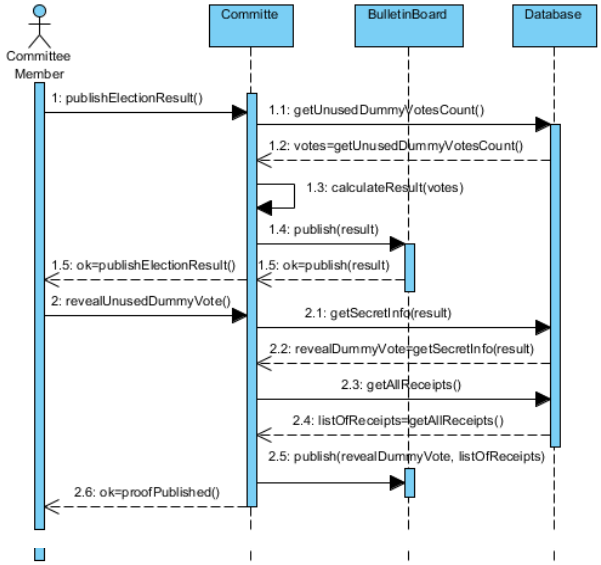


Fig. 3: Tally sequence diagram

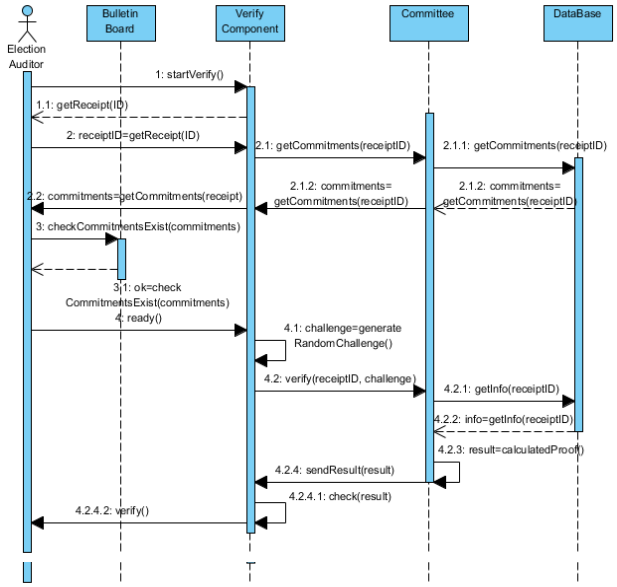


Fig. 4: Verify sequence diagram

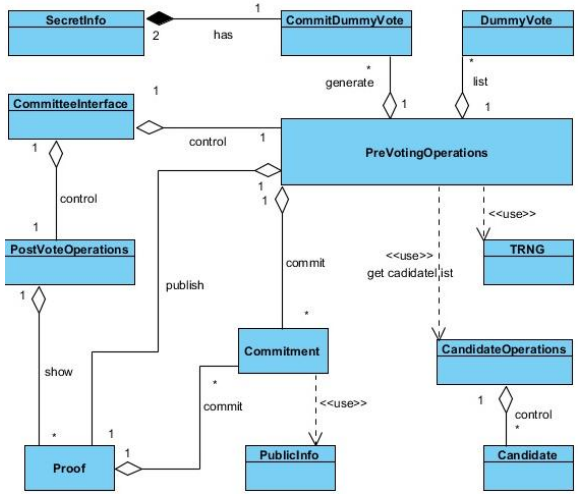


Fig. 5: Committee class diagram

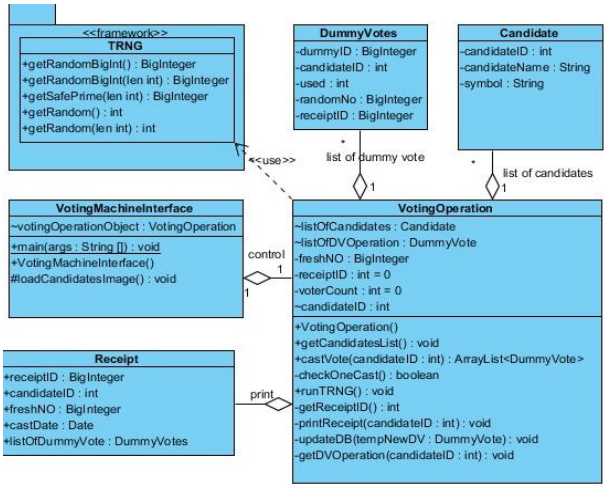


Fig. 6: Voting class diagram

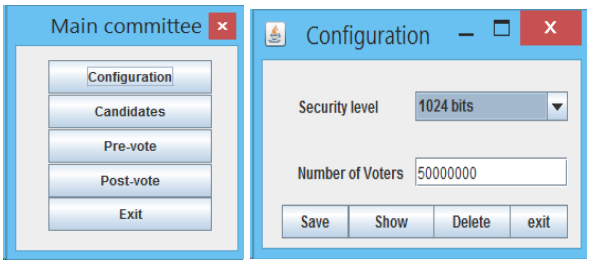
Fig. 5 shows the class diagram that describes the association between the committee component classes. The PrevotingOperation class encapsulates all procedures needed before the voting operation, such as generating dummy votes. The CandidateOperations class encapsulates all procedures needed after the election complete operation, such as calculating results. The PostVoteOperation class encapsulates all needed procedures after the voting operation, such as verifying correctness. The Candidate class represents a real candidate in the e-voting system, including all methods and attributes related to this actor. The Commitment class encapsulates all methods and properties of computing commitments. The DummyVote class represents a dummy vote value before committing. The CommitDummyVote class represents the dummy vote value after commitment. The PublicInfo class encapsulates the needed public data in the commitment scheme. The SecretInfo class encapsulates the needed secret data in the commit operation. The TRNG class encapsulates all methods and properties of generating fresh random numbers.

Fig. 6 shows the association between the Voting component classes. The VotingOperation class encapsulates all procedures that happen during the voting process. The Receipt class encapsulates all methods and properties of generating a receipt.

4.2 Bingo Voting Implementation

We have implemented Bingo using Java 1.8 in approximately 7,000 lines of code. We have used a library called UniCrypt[22] to perform needed cryptographic functions. UniCrypt is an open source Java library developed by E-Voting Group (EVG) to simplify the implementation of cryptographic voting protocols. UniCrypt consists of two layers; mathematical fundamental layer, based on the concept of groups and their elements, and cryptographic-primitives layer, which provides many cryptographic schemes. One of the advantages of UniCrypt is not relying on a third party library that requires the Java standard edition runtime environment, so it can be deployed on smart mobiles also.

Our implementation consists of four subprograms; Committee, Voting Machine, TRNG and Verifier.



(a) Main committee

(b)Configuration of pre-voting process

```
//Get random gOrder as safe prime with length =lengthOfP
RandomNoGeneratorrn = newRandomNoGenerator();
BigIntegergOrder = rn.getSafePrime(lengthOfP);
//Create a cyclicGroup with order gOrder
CyclicGroupcg = GStarModSafePrime.getInstance(gOrder);
```

Fig. 8: Code of cyclic group creation

Fig. 7: Sample screens from the Bingo Voting prototype

Fig. 7(a) depicts the main committee interface. When the voting authority user clicks on the Configuration button, the configuration interface appears as in Fig. 7(b). In addition, a Candidate interface appears if the Candidates button is pressed. The Pre-vote interface appears if the Pre-vote button is clicked. Also, if the user presses the Post-vote button, the post-vote interface appears. The Exit button is used to close the window.

Fig. 7(b) depicts the configuration interface, through which the authority member can change the security level from the combo box and enters the number of voters into the text box. If the level of security is high, the computation overhead increases and, as a result, the speed of computing commitments decreases. Then, clicking the Save button instructs the system to save this information. The Show button displays the public information of the cyclic group, and the Delete button is used to delete the public info.

As an example, if security level is 1024 bits, it means that the length of $p = 1024$ bits, where p is the order of cyclic group $\mathbb{G}$ (see Fig. 8), it is used for the Pedersen commitments (see Fig. 10). The public information consists of p and q (the two safe prime values), where $p = 2q + 1$, and g and h (two generators of the cyclic group $\mathbb{G}$). After that the system generates the public info and the election committee enters the candidates' information by the Candidates interface.

Then, the system creates j random numbers for each candidate, where j is the number of voters, and saves all these random numbers in a secure database. After that, the dummy votes of all candidates ($DV = commit(randomNO),\ commit(candidateID)$) are published as a list of dummy votes. The last step in the pre-voting phase is to prove that all candidates have the same number of dummy votes. This is done using an efficient proof of a shuffle of known content by Groth protocol (see Fig. 9). The protocol's public inputs are the cyclic group, the set of messages $M = \{m_{\pi(1)}, m_{\pi(2)}, \ldots\ldots, m_{\pi(n)}\}$ that represents the shuffled candidate IDs,

and the set of second commitments from the dummy votes $C = \{c_1, c_2, \ldots c_n\}$. The private inputs of the prover are the set of random numbers $(r_1, r_2, \ldots r_n)$ and the permutationπ, where $c_i = commit(m_{\pi(i)}, r_i)$. Then, P and V perform the steps illustrated in Fig. 9.

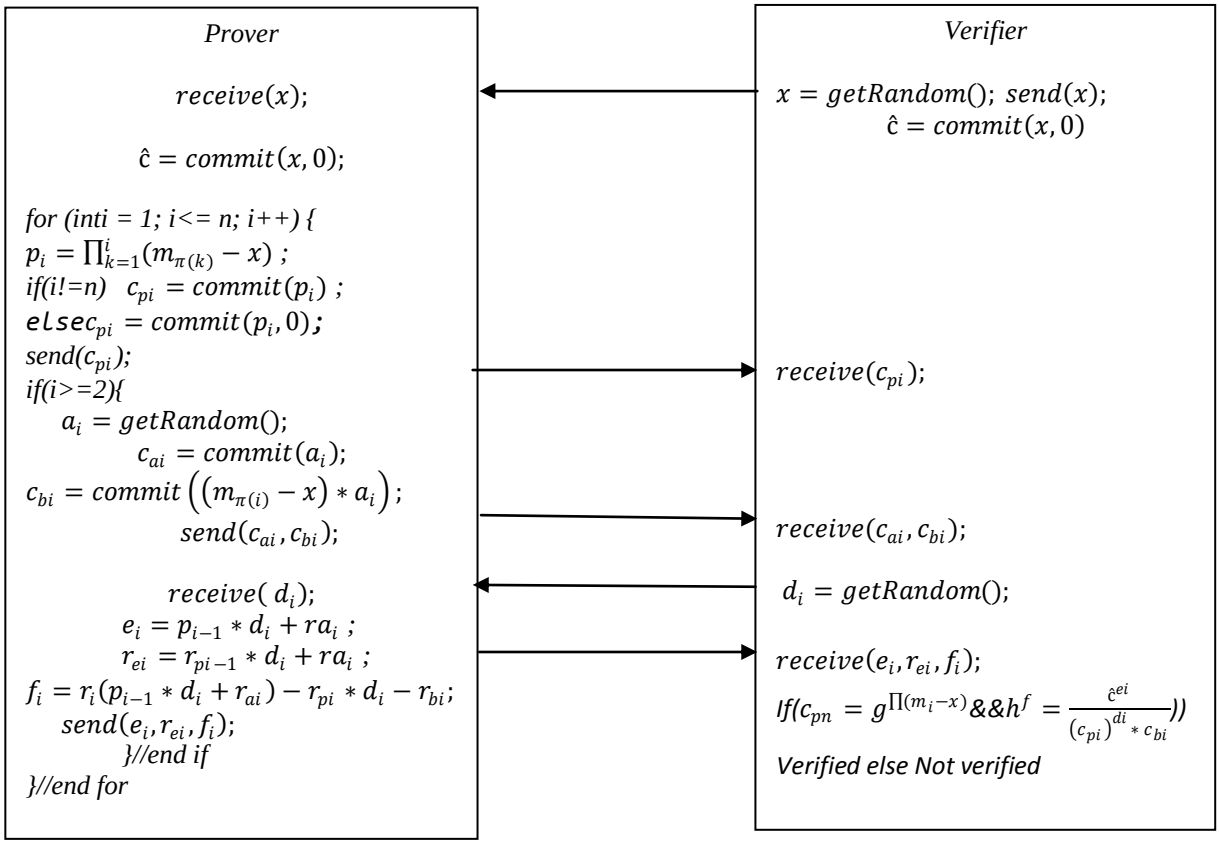


Fig. 9: Pseudocode of efficient proof of a shuffle of known content

The Candidates interface in Fig 12, allows the committee user candidates' information (candidate number, candidate name and candidate symbol). To add new candidate, the user enters the candidate information and clicks Save button. The Delete button is used to delete the displayed candidate information. In Fig 12, candidate number 1 will be deleted if the Delete button is pressed. The Search button is used to find a particular candidate's information. When pressing the Search button, the input box will ask the user to enter the candidate's number. The screen displays the candidate's information if the Candidate's number exists or notifies the user that there is no information for this id. To delete all candidates, the button Delete all is used.

In the Election Day, the subcommittee in the location of the voting booth checks each voter's eligibility, as in any traditional election, and allows the voter to enter the voting booth to cast her vote. In the booth the voting-machine (VM), subprogram displays a list of candidates (see Fig. 14(a)). We assume the numbers photos are the candidates' symbols. When the voter chooses her candidate by pressing the candidate symbol, the voting-machine requests a fresh number from the TRNG. Here, the voter must ensure that the fresh number (FN) has assigned to her selected candidate by checking if the fresh number displayed on the screen of TRNG matches the fresh number beside her selected candidate (see Fig. 14(b) and (c) . As shown in the Fig. 14, the voter has chosen the second candidate, because the FN appears next to the second candidate. The selected candidate gets one vote and each of the other candidates takes a random number from his unused dummy votes; the dummy vote is then marked as used. This means that each candidate loses one vote except the selected one. Then, the VM prints a receipt, the voter takes the receipt and keeps it to use later in the verification step. Fig. 15 shows the receipt which is given to the voter. This receipt does not show the voter's choice, because no one can distinguish the FN in the receipt.

After the election is finished, the committee calculates the results by tallying and opening unused dummy votes only. In the prototype, this is done by going to the main committee interface (Fig 7(a)) and clicking the Post-vote button. Then the Post-vote interface appears as in Fig. 16. If the user presses the Publish result button, the result appears (see Fig. 17).

To prove the correctness of the result, the authority publishes the reveal information of the commitments of all unused dummy votes and publishes all receipts. Also, the voter or any third party can verify

the correctness of receipts. This is done in two steps; in the first step the voter checks if his receipt exists and in the second step the voter verifies the validity of the content of the receipt (see Fig. 18).

```
//Get group order into p
BigIntegerp = publicInfo.getpGroupOrder();
//Create cyclic group with order p
CyclicGroupcg = GStarModSafePrime.getInstance(p);

//Put the message generator into G
G=publicInfo.getMessageGenerator();
Element g = cyclicGroup.getElementFrom(G);

//Put the random generator into H
H=publicInfo.getRandomNumberGenerator();
Element h =cyclicGroup.getElementFrom(H);

//Create Pedersen commitment with two
generators
PedersenCommitmentSchemepc =
PedersenCommitmentScheme.getInstance(h, g);

//Pass a message to Pedersen commitment
Element m,r;
m= pc.getMessageSpace().getElementFrom(message);

//Get random element to be used in Pedersen
commitment
r =pc.getRandomizationSpace().getRandomElement();

//Commit the message
c = pc.commit(m, h);
```

Fig. 10: Code of commitment of a message

```
BigInteger p = publicInfo.getpGroupOrder();

BigInteger h=publicInfo.getRandomNumberGenerator();

for (inti = 2; i<listOfd.size(); i++) {
        //calculate the ĉ^ei
numerator = listOfCl.get(i).modPow(listOfe.get(i),p);

        // calculate the (c_pi)^di
equation1 = listOfCp.get(i).modPow(listOfd.get(ip);

        // calculate the (c_pi)^di * c_bi
denominator = equation1.multiply(listOfCb.get(i)).mod(p);

        // calculate the ĉ^ei / ((c_pi)^di * c_bi)
leftSide =
numerator.multiply(denominator.modInverse(p)).mod(p);

// calculate the h^f
rightSide = h.modPow(listOff.get(i),p);

        //Verify
        if (leftSide.compareTo(rightSide) == 0)
                return true;
            else
                return false;
```

Fig. 11: Code for verifying if $h^f = \hat{c}^{ei}/(c_{pi})^{di} * c_{bi}$

V. Performance Evaluation of Bingo Prototype

This section describes a case study to evaluate the performance of the Bingo prototype.

5.1 Case study

The following case study represents a simple election with ten voters and three candidates. The information of the candidates is entered in the per-voting phase. Then, a cyclic group is created. In this example, we assume the group order is 32 bits to simplify the example, but in real voting the group order must be large (e.g., 1024 bits) to guarantee security. The cyclic group and the candidates are shown in Table 2 and Table 3 respectively.

Then, the election authority creates n random numbers for every candidate where n is the voters' number. So, in total $totalNo = n * i$ are generated, where i is the number of candidates. Table 4 shows every candidate and its random numbers.

After creating 30 random numbers for all candidates, the authority creates the dummy votes (Table 5). Every dummy vote is a pair of commitments; the first commitment is of the random number $commit(n)$ and the second commitment is of candidate ID $commit(candID)$. So, every $DV = commit(n), commit(candID)$. Where $commit(n) = g^n * h^r mod\ p$ and $commit(candID) = g^{candID} * h^r mod\ p$, where r choice at random and g and h are two generators of the cyclic group.

Let us take the first random number of the first candidate (see Table 4) as an example of creating dummy votes. $DV1 = commit(718737022),\ commit(1)$. $commit(718737022) = (2060739020)^{718737022} * (2252587668)^{748076828}\ mod\ 2638495943 = 506775964$ and $commit(1) = (2060739020)^1 * (2252587668)^{747039234}\ mod\ 2638495943 = 661220067$.

So, $DV1 = (506775964, 661220067)$. Then, the dummy votes are shuffled and published with a proof that every candidate has the same number of dummy votes. This proof opens the second commitment of all dummy votes, so that the candidate IDs appear. This is a proof of a shuffle of known content (see Fig. 9).

Here, the verifier can check if the left side equals the right side or not; if equal, that means every commitment $c_i \epsilon (c_1, c_2, \ldots c_i)$ contains only one message $m_i \epsilon (m_{\pi(1)}, m_{\pi(2)}, \ldots\ldots, m_{\pi(i)})$. So, the verifier can check if every ID appears the same number of times, in the messages$(m_{\pi(1)}, m_{\pi(2)}, \ldots\ldots, m_{\pi(i)})$; if yes, then every candidate has the same number of dummy votes and vice versa. All previous steps are called pre-voting phase, which should be happens before the election starts.

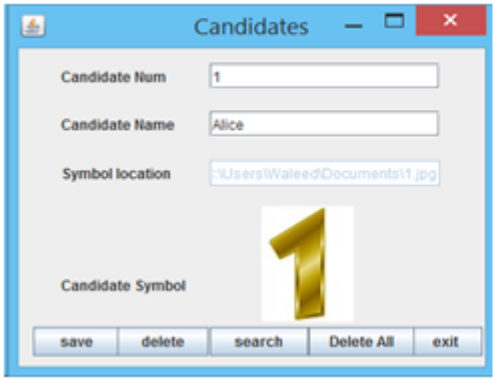
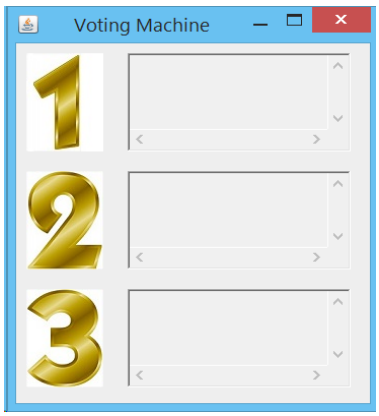


Fig. 12: Candidates interface

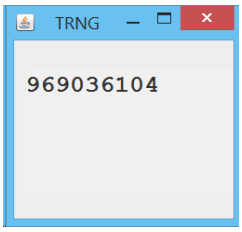
Fig. 13: Pre-vote interface

As in Fig.14, the voter chooses the second candidate because the fresh number generated by the TRNG is displayed next to the second candidate. A voter can know if the voting machine has cheated her by checking whether the fresh number appears next to her choice on the VM screen and in the receipt (Fig. 15) or not. The second candidate keeps one dummy vote (calculated as a vote in the tally) and the other two candidates lose one dummy vote (the voter can ensure the latter by verifying that her receipt is formed correctly).



(a)Voting Machine (VM) before casting

(b) VM after casting



(c)TRNG interface

Fig. 14: Screens at the voting booth

Dear voter

Receipt Generated On - Wed Feb 18 20:50:53 EET 2015

Bingo Voting (Receipt NO 1)		
1	Alice	540679138
2	Bob	969036104
3	John	648514493

keap this receipt to verify...

Fig. 15: Receipt

Table 2: Cyclic group info

p	2638495943
q	1319247971
g	2060739020
h	2252587668

Table 3: Candidate info

ID	Name
1	Alice
2	Bob
3	John

Fig. 16: Post-vote interface

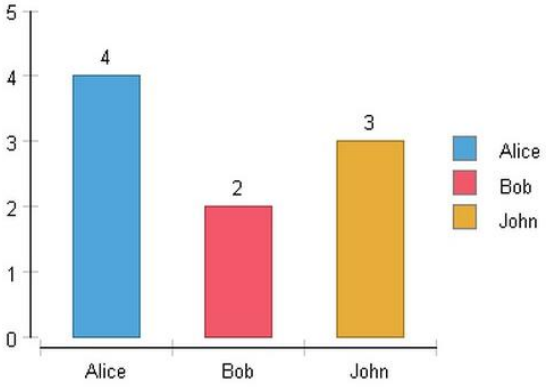


Fig. 17: election result

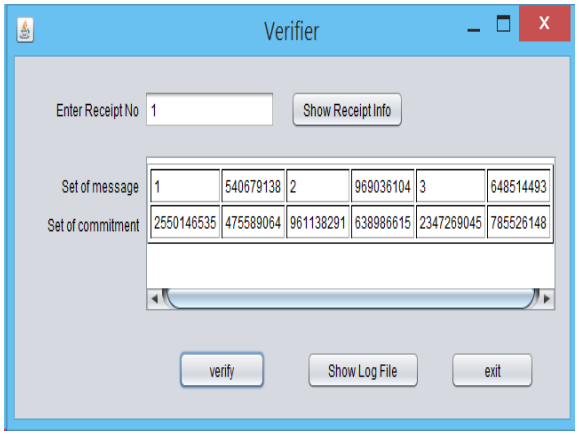


Fig. 18: Verifier interface

Table 4: Candidates' random numbers

Alice	Bob	John
718737022	949637843	648514493
664061844	727674345	1061417450
961533900	826256578	819496873
597953815	872008370	717687494
828036001	584078372	812468695
763463492	571099849	1046829965
898817955	1003839261	674460312
758952278	656563607	618638857
754109819	780406295	900347419
540679138	725424571	697942052

In the post-vote phase, the authority calculates the election result by collecting the remaining dummy votes for every candidate and subtracting the absentees. If we assume that one of the ten voters is absent and four voters choose the first candidate, two voters choose the second candidate, and three voters choose the third candidate, the remaining random numbers for every candidate after the voting finishes are shown in Table 6. The election result $= remainDV - absent$. For $cand1 = 5 - 1 = 4$, for $cand2 = 3 - 1 = 2$, and for $cand3 = 4 - 1 = 3$.

Then, all unused or remained dummy votes are published with their reveal information to the bulletin board. The authority publishes digital copies of all receipts created by the voting machine in the voting phase and the proof that every receipt is constructed correctly (see Table 7). If a voter inquires about receipt number 1(see Fig. 18), then the receipt information exhibits in the editor pane. The set of messages is $M = \{(1, 540679138), (2, 969036104), (3, 648514493)\}$, which appears in receipt number 1 (see Fig. 15). Here, the voter guarantees that her receipt information was not changed. In addition, the voter can search for her receipt information after election authority publishes digital copies of all receipts (Table 9). The editor pane in Fig. 18 shows the set of commitments $C = \{(2550146535, 475589064), (961138291, 638986615), (2347269045, 785526148)\}$. For correctness, the voter must insure that every commitment from C contains only one message from M. This proof is done by efficient proof of a shuffle of known content (see Fig. 9).

In addition, the voter verifies that her receipt contains $numberOfCandidates - 1$ dummy votes (published in pre-vote phase). These dummy votes must be one of each candidate except the selected candidate. Because every uncounted dummy vote appears in a receipt, this means that the candidate who has this dummy vote lost one vote. Here, voter searches in the published dummy votes (Table 5) and finds that the dummy votes with IDs (4, 7) are used in the receipt (number 1). But, no one can know these dummy votes are for which candidates; otherwise the voter's choice will be revealed. Since the number of candidates is three, this receipt contains two dummy votes. This means that this receipt affects the election result by adding one vote. Since the voter guaranties during the voting process (individual verification) that the FN has gone next to her chosen candidate, this receipt contains two dummy votes and each candidate is represented once in this receipt (in the set of message each candidate ID appears once). So this receipt is formed correctly.

Table 5: Generated dummy votes in the case study election

DVID	Commit(random)	commit(CandidateID)
1	506775964	661220067
2	452596593	199116972
3	190290780	2604765173
4	2550146535	475589064
5	949955150	1943771889
6	687724326	258376349
7	638986615	961138291
8	264946713	1197452649
9	1264144908	660019061
10	2453224082	1702997694
11	949583802	981193728
12	2116084166	1739685557
13	1937467131	1762373354
14	51602341	2327002076
15	2255223524	146135369
16	565736580	1739788070
17	229710590	1005168288
18	96793866	1102136801
19	1452535935	1713666352
20	1414560182	1846923293
21	2481927530	2336506451
22	2548125545	1807127465
23	1354796730	1865007420
24	261107135	1712116904
25	2219800873	310350223
26	1161970051	2532924964
27	1191530419	2292780905
28	1215461090	1794448993
29	2391083785	1451610246
30	2625762441	164963088

Table 7: All receipts election

ReceiptID	CandidateID	RandomNO
1	1	540679138
	2	969036104
	3	648514493
2	1	921751750
	2	872008370
	3	1046829965
3	1	912986837
	2	780406295
	3	697942052
4	1	828036001
	2	1031942604
	3	819496873
5	1	754109819
	2	656563607
	3	568822455
6	1	718737022
	2	826256578
	3	545346373
7	1	763463492
	2	584078372
	3	944457595
8	1	807879909
	2	1003839261
	3	812468695
9	1	917287570
	2	725424571
	3	674460312

5.2 Bingo voting performance

In this subsection, the performance of our Bingo prototype is described. The proposed prototype is executed on a PC with 4 GB RAM and Intel Pentium dual core CPU T3200 @2.00 GHz with Windows 8 (32-bit). We measured the computation time (CT) of creating cyclic groups with 128, 256, 512, 1024 and 2048-bit group orders. Also, we measured the computation time of creating dummy votes with 512-bit group order. Receipt proof is used to prove to voter that her receipt contents are correct. This proof uses an efficient proof of a shuffle of known content protocol by Groth. The proof computation time depends on the number of candidates and the order of cyclic group (CG).

Fig. 19 shows the CT of receipt proof with three candidates while changing the cyclic group order. The CT increased with CG order because the commitment length increased. Also, the number of candidates affected the CT of receipt proof (Fig. 20) because the number of commitments that must be proved depends on the number of candidates.

According to proposed Bingo prototype, Bingo proves that every candidate has the same number of dummy votes. To do that we must open the second commitment of each dummy vote then the candidate ID appears. Then, grouping every candidate ID and verifying that each candidate has the same number of dummy votes are done. The computation time of first proof depended on CG order and the number of dummy votes where $dummyvotes = candidatesNo * votersNo$. In Fig. 21, 300 dummy votes (three candidates and 100 voters) were used with changing CG order. Computation time increased as the CG order increased because the commitment length increased. Also, when the number of dummy votes increased the number of commitments increased too and computation time increased (see Fig. 22).

Table 6: Remaining candidates' dummy votes

Alice	Bob	John
	949637843	
664061844		
		819496873
597953815		
828036001		
	571099849	
		674460312
758952278		
		900347419
718737022	725424571	697942052

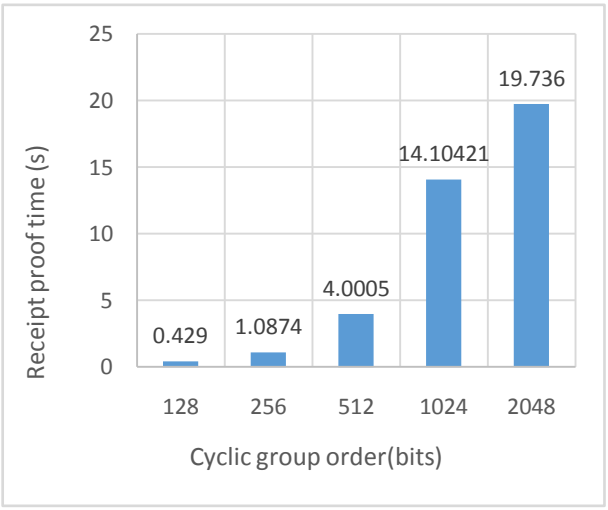


Fig. 19: Receipt proof computation time vs. cyclic group order

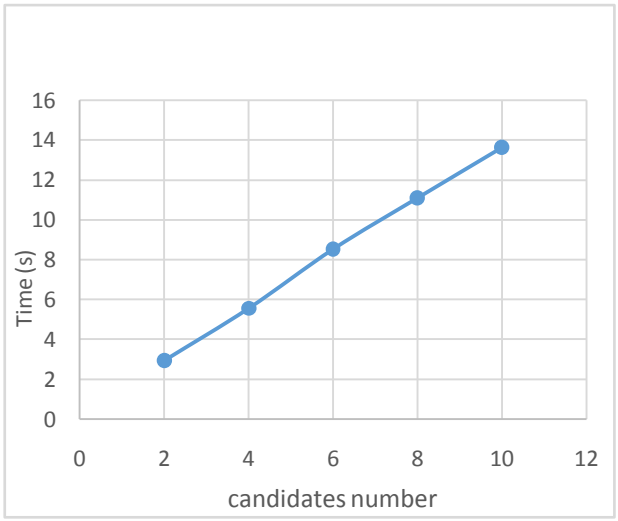


Fig. 20: Receipt proof computation time vs. candidates' number

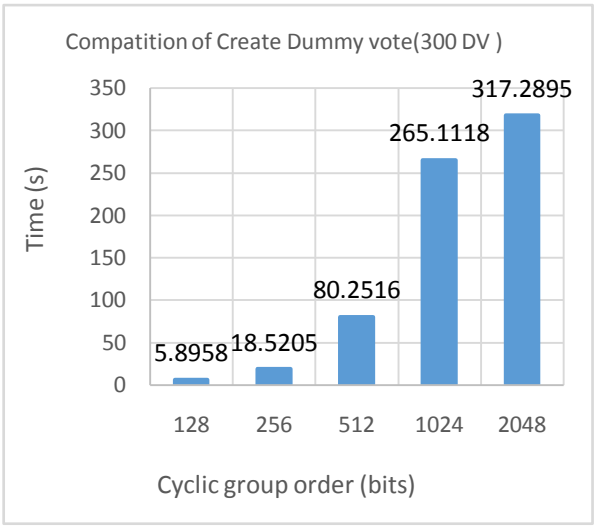


Fig. 21: Creation Time of 300 dummy votes vs. cyclic group order

The time to create dummy votes was effected by two factors: the cyclic group order and the number of dummy votes. Fig. 21 shows, the time increased with cyclic group order and Fig. 22 shows the computation time of creating dummy votes with 512-bit cyclic group if change the dummy numbers. As noted in Fig 22 when the number of dummy votes increased, the computation time also increased linearly. Approximately 3.95 dummy votes were created every second.

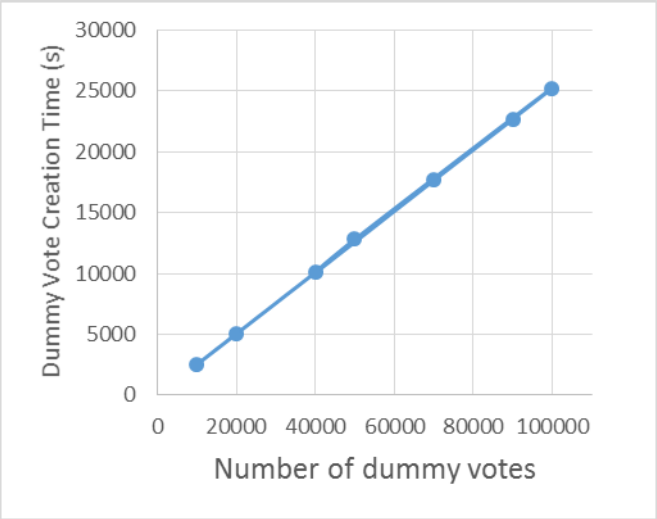


Fig. 22: First proof computation time vs. cyclic group order

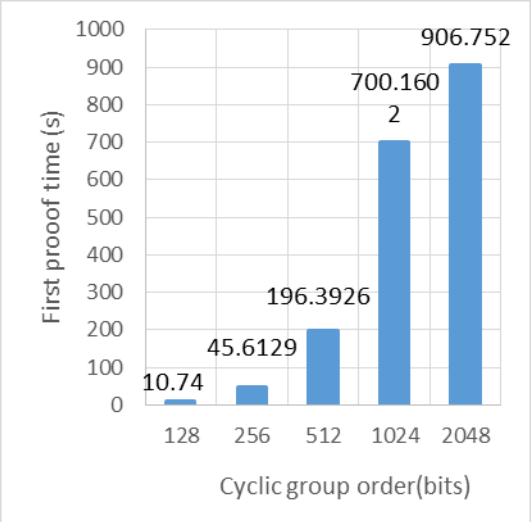


Fig. 23: Creation time of dummy votes vs. number of dummy votes at 512-bit cyclic group order

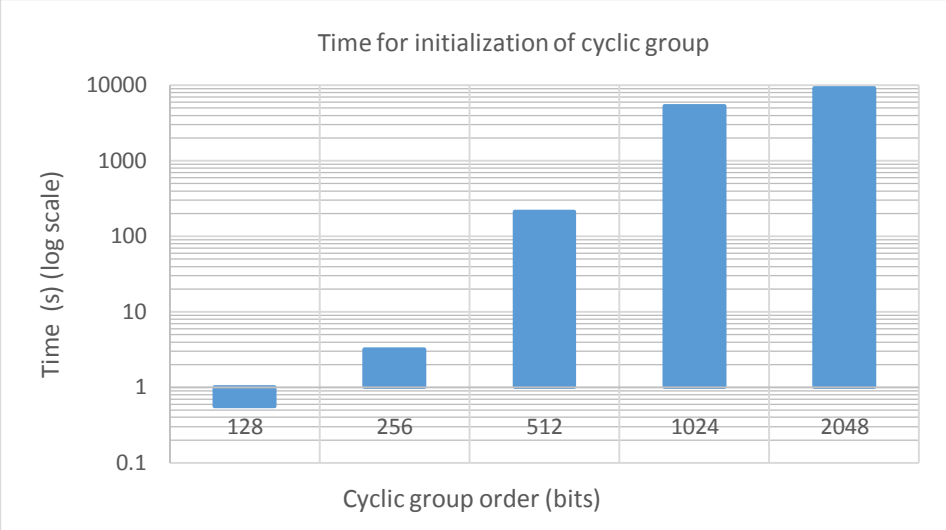


Fig. 24: Computation time for creating acyclic group vs. cyclic groups order

The computation time for initialization of the cyclic group depended on the cyclic group order (Fig. 24).When the cyclic group order increased the commitment length increased and the discrete logarithm became harder to compute, thereby the commitment security was stronger. But, the speed decreased.

VI. Conclusions

This paper studies the performance of the cryptographic operations used in the Bingo e-voting protocol. It also explains the sequence and class diagrams, implementation details, and user interface of a prototype implementation of Bingo voting. We demonstrate the prototype implementation using a case study of an election with three candidates and ten voters. Finally, we measured the performance of the Bingo voting implementation while changing the cyclic group order and the number of dummy votes.

Our next steps are to analyze the Bingo voting security and evaluate its coercion-resistance and privacy properties, comparing the protocol with other end-to-end e-voting protocols, such as Scantegrity, ScantegrityII, ThreeBallot and Punchscan.

References

[1]. R. Jardí-Cedó, J. Pujol-Ahulló, J. Castellà-Roca, and A. Viejo, "Study on poll-site voting and verification systems," vol. 31, no. 8, pp. 989-1010, 2012.

[2]. M. F. Mursi, G. M. Assassa, A. Abdelhafez, and K. M. Abo, "On the Development of Electronic Voting: A Survey," vol. 61, no. 16, pp. 1-11, 2013.
[3]. R. Küsters, T. Truderung, and A. Vogt, "A game-based definition of coercion resistance and its applications," vol. 20, no. 6, pp. 709-764, 2012.
[4]. L. AhmedQubati, S. Khattab, and I. Farag, "Survey on End-to-End Verifiable Cryptographic Voting Systems," vol. 100, no. 16, pp. 43-57, 2014.
[5]. A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, Handbook of applied cryptography: CRC press, 2010.
[6]. MathWorld. (February 21). Group Order. Available: http://mathworld.wolfram.com/GroupOrder.html
[7]. G. Brassard, D. Chaum, and C. Crépeau, "Minimum disclosure proofs of knowledge," vol. 37, no. 2, pp. 156-189, 1988.
[8]. D. Chaum, I. B. Damgård, and J. Van de Graaf, "Multiparty computations ensuring privacy of each party's input and correctness of the result," in Advances in Cryptology—CRYPTO'87, 1988, pp. 87-119.
[9]. T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in Advances in Cryptology—CRYPTO'91, 1992, pp. 129-140.
[10]. O. Goldreich, S. Micali, and A. Wigderson, "How to prove all NP statements in zero-knowledge and a methodology of cryptographic protocol design," in Advances in Cryptology—CRYPTO'86, 1987, pp. 171-185.
[11]. S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," vol. 18, no. 1, pp. 186-208, 1989.
[12]. D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," vol. 24, no. 2, pp. 84-90, 1981.
[13]. C. A. Neff, "A verifiable secret shuffle and its application to e-voting," in Proceedings of the 8th ACM conference on Computer and Communications Security, 2001, pp. 116-125.
[14]. J. Groth, "A verifiable secret shuffe of homomorphic encryptions," in Public Key Cryptography—PKC 2003, ed: Springer, 2002, pp. 145-160.
[15]. J. Groth, "A verifiable secret shuffle of homomorphic encryptions," vol. 23, no. 4, pp. 546-579, 2010.
[16]. J. D. C. Benaloh, Verifiable secret-ballot elections: Yale University. Department of Computer Science, 1987.
[17]. J. C. Benaloh and M. Yung, "Distributing the power of a government to enhance the privacy of voters," in Proceedings of the fifth annual ACM symposium on Principles of distributed computing, 1986, pp. 52-62.
[18]. J. D. Cohen and M. J. Fischer, "A robust and verifiable cryptographically secure election scheme," in 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, 1985, pp. 372-382.
[19]. J. M. Bohli, C. Henrich, C. Kempka, J. Muller-Quade, and S. Rohrich, "Enhancing Electronic Voting Machines on the Example of Bingo Voting," vol. 4, no. 4, pp. 745-750, 2009.
[20]. M. Bär, C. Henrich, J. Müller-Quade, S. Röhrich, and C. Stüber, "Real world experiences with Bingo Voting and a comparison of usability," in Workshop On Trustworthy Elections, WOTE, 2008.
[21]. J.-M. Bohli, J. Müller-Quade, and S. Röhrich, "Bingo voting: Secure and coercion-free voting using a trusted random number generator," in E-Voting and Identity, ed: Springer, 2007, pp. 111-124.
[22]. P. Locher and R. Haenni, "A Lightweight Implementation of a Shuffle Proof for Electronic Voting Systems," no.