Journal of Information Engineering and Applications
ISSN 2224-5782 (print) ISSN 2225-0506 (online)
Vol.7, No.1, 2017

Integrated Security Protocol for Electronic Election System

Dr. Mahmood Khalel Ibrahem
Al-Nahrain University - College of Information Engineering

Abstract
Electronic election systems are a sensitive application and security is a major concern. This process should be ensured to maintain the integrity and confidentiality of the casted votes and voters authentication before they casts their votes. Besides security, other issues need to be considered such as; simplicity, reliability, convenience, flexibility, mobility and cost. In this paper, an integrated security protocol and a prototype of a web-based electronic election system is proposed by integrating two techniques; the first technique is a new zero knowledge mutual authentication protocol based on Diffie-Hellman (D-H ZKP) key exchange algorithm, to enforce a mutual authentication between the election authority server and the voters and exchange secret key securely. The second technique is homomorphic encryption scheme to encrypt all the votes and perform the calculation of the votes without revealing any information about it to ensure the secrecy of the votes and maintain the confidentiality. The proposed protocol provides secure voting over the Internet and maintains the requirements of the election process.
Keywords: E-election, D-H ZKP, Homomorphic Encryption.

1. Introduction

Electronic election is the voting process held over electronic media, i.e. computers and information technologies. E-election systems can be categorized into two types; voting machines and Internet voting. Voting machines are specialized equipment located at election centers. They typically allow the voters to cast their vote easily and to process the ballots. On the other hand, Internet voting does not need physical election centers; voters can cast their votes using their own devices such as computers, smart devices (phones and tablets). In this paper, we focus on Internet voting systems, specifically security issues.

In election applications, security is one of the major issues, simplicity is another requirement to allow wide participation of citizens. With the addition to security and simplicity, such as reliability, convenience, flexibility, mobility and cost, need to be considered. In that respect, a well-designed protocol is necessary to incorporate all of these requirements. Computers and Internet has been used widely which makes election over the Internet a promising solution for many reasons such as; eliminating waiting time, minimizing voting error and election fraud [1]. Although advances in network security offer a wide range of protocols to design election system with reasonable security, but still carefully designed protocols and enhancements of the applications are necessary to protect them from the network threats [2].

Two approaches of electronic voting security protocols can be noticed; blind signatures and homomorphic encryption. In blind signatures, the voter supplied with a token, "a blindly signed message", unknown to anyone except himself. The voter sends his token together with his vote over a secure channel. These schemes require voter's participation in more rounds. In homomorphic encryption, the voter interacts with the authority server to encrypt his vote. According to the homomorphic property, the encrypted votes are summed and the result of the election is computed after decrypting the sum by the authorities [3].

In this paper, a theoretical background of zero-knowledge proof, homomorphic encryption and modified Diffie-Hellman algorithm for mutual zero-knowledge is discussed. An integrated security protocol and a prototype of secure electronic election system using zero-knowledge with modified D-H algorithm and homomorphic encryption is presented with examples. Finally analysis and conclusions of the proposed system is discussed.

2. Characteristics of Secure E-Election

Electronic election system has major properties that should be considered during the design and implementation processes of the election system. The following is a main requirements for electronic voting systems that are considered in the electronic voting literature [4]:

2.1 Accuracy: A validate vote should not be possible to altered or eliminated, and it is not possible for an invalid vote to be counted in the final tally.
2.2 Democracy: An election system should allow only eligible voters to vote, and it ensures that each eligible voter can vote only once.
2.3 Privacy: No one (including election authority) can track (link) any ballot to a voter, and no voter can prove that he/she voted to specific candidate.
2.4 Integrity: Votes should not be able to be modified, forged, or deleted without detection.
2.5 Verifiability: The election system should allow anyone to verify that all votes have been counted correctly.

2.6 Convenience: The election system should allow voters to cast their votes quickly, in one session, with minimal equipment and skills.
2.7 Mobility: The election system should be designed with no restrictions (other than logistical ones) on the location from which a voter can cast a vote.
2.8 Other properties might be added according to the nature of the election system such as; reliability, audibility, accountability, Transparency, Cost-effectiveness, etc.

3. Zero Knowledge Authentication

Authentication systems include verifying entities credentials by some authorized server to grant them an access to a specific system. Zero-knowledge proof is an authentication system which allows one party to prove to a second party via some public information that he possesses some secret information without revealing the secret to the second party [5].

ZKP is used essentially to construct a trusty interaction between two parties (prover and verifier) without revealing secret information to prove their identities [6].

In ZKP system, a prover prove to the verifier that some statement is true by passing public numbers about the validity of the assertion and nothing about the secret. In a proof of knowledge, the prover also prove to the verifier that he possess a "witness" for the given statement. The interactions between involved parties are designed in such a way that the secret information cannot be revealed or guessed. At the end of interaction session, the verifier knows that the claimant possesses the secret or not, nothing more. The result of the interaction is either accepted or rejected [6].

3.1 Definition of Zero-Knowledge Proof

"ZKP computation model is defined as an interactive proof system (P,V), where P is a *prover* and V is a *verifier*. This system is used to prove a statement for a language over $\{0,1\}$. Given L a language over $\{0,1\}$*, for a membership instance $x \in L$, P and V must share the common input x, Proof instance is denoted as $(P,V)(x)$" [7].
The prover (P) and the verifier (V) exchanges a sequence of proof statements, p_1, v_1, p_2, v_2... p_n, v_n over some communication channel. Each element p_i, v_i exchanged is bounded by polynomial in $|y|$ and proof instance $(P,V)(y)$ must terminate in polynomial time in $|y|$. The prover claims that $y \in L$. After completion of the interaction session, the result should be of form $(P,V)(y) \in \{Accept, Reject\}$ representing verifier (V) acceptance or rejection of prover (P) claim [8].
The following properties should be satisfied in ZKP [9, 10]:
3.1.1 Completeness: if the statement is true, then an honest *prover* who follows the protocol should be able to convince the *verifier*.
3.1.2 Soundness: if the statement is false, then any cheating *prover* (even if it has computationally unbounded power) cannot convince the *verifier* except with very small probability.
3.1.3 Zero-knowledge: if the statement is true, cheating *verifier* knows nothing other than the given statement.

3.2 ZKP Protocols

One of the famous zero-knowledge schemes is Fiat-Shamir identification scheme; which is an interactive zero-knowledge proof that permits the (*prover*), to convince the (*verifier*), that he possesses a secret without disclosing the secret. For full description of the scheme refer to [10, 11].

The scheme is a probabilistic so that a dishonest claimant has a 50 percent chance of fooling the verifier and passing the test (by predicting the value of the challenge) in other words, the verifier assigns a probability of ½ to each round of the test. If the probability decreases to $(1/2)^{20}$ or $9.54*10^{-7}$ then it is highly improbable that the prover can guess 20 times correctly. Unfortunately, decreasing the probability will increase the computation cost [12, 13].

A new ZKP protocol proposed by [14] based on Diffie-Hellman Key Exchange Algorithm (D-H ZKP) "in the sense that both parties (the prover and the verifier) exchange non secret information without revealing secret information to get one identical secret key" [14].

Both parties (prover and verifier) exchanges public numbers and calculates shared secret key (K_{alice} and K_{bob}), hence they can exchange encrypted data using generated key. Using the shared secret key, they encrypt their replies (R_1 and R_2) into (C_1 and C_2). Encrypted replies is decrypted into (R_1' and R_2') and compared with the original replies (R_1 and R_2) to verify the claim of the claimant. At the end of the process both parties can either prove their honesty or rejected in a mutual authentication process. The proposed algorithm is deterministic; hence there is no soundness error and no chance that a cheating prover can convince the verifier [14].

The interaction between the prover and the verifier is designed in such a way that passing public numbers without revealing their secret numbers using the properties of discreet logarithm to generate shared secret key. Replies are encrypted by one side and decrypted by other side using shared secret key to avoid man-in-the-middle-attack and prove mutual authentication. A slight enhancement has been made on the protocol to

reduce unnecessary processing. For more detail refer to the original protocol in [14]. Table-1 states the protocol. Figure-1 illustrates D-H ZKP protocol diagram.

Table 1. Modified D-H ZKP protocol.

1. Alice (the prover) chooses a large random number x, such that $0 < x < p$ and calculate $R_1 = g^x \bmod p$.
2. Alice sends R_1 to Bob (the verifier).
3. Bob (the verifier) chooses another large random number y, such that $0 < y < p$ and calculate $R_2 = g^y \bmod p$, $K_{Bob} = (R_1)^y \bmod p$, and $C_1 = E(K_{Bob}, R_2)$.
4. Bob sends $(R_2 \| C_1)$ to Alice.
5. Alice, calculates $K_{Alice} = (R_2)^x \bmod p$, decrypt $(R_2' = D(K_{Alice}, C_1))$ and verify $(R_2 = R_2')$. If they matched then she proceeds; otherwise the verifier is dishonest and terminates.
6. Alice encrypt $(C_2 = E(K_{Alice}, R_1 \| R_2)$ and send it to Bob.
7. Bob decrypt C_2 to get R_1' and R_2', $(R_1 = D(K_{bob}, C_2)$.
8. Bob verify $(R_1 = R_1')$; if they are equal then Alice is verified (Accepted), otherwise it is a dishonest prover (rejected.)

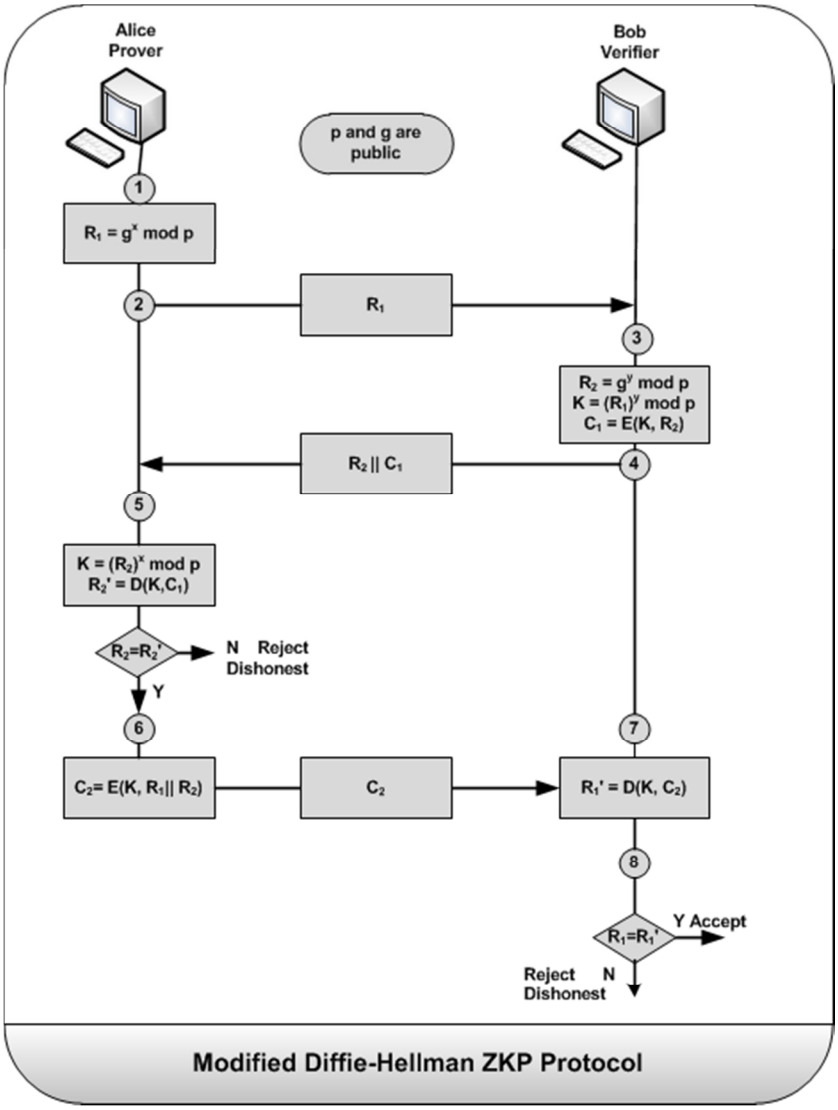

Figure 1. Modified Diffie-Hellman ZKP protocol [14]

4. Homomorphic Encryption

Homomorphic encryption schemes are cryptographic constructions which enable to securely perform operations on encrypted data without ever decrypting them, which is of great importance because of its wide and growing range of applications such as; cloud computing, e-voting, and information retrieval. In homomorphic encryption, the sum of two encrypted values is equal to the encrypted sum of the values. Homomorphic property is useful in e-voting applications in the sense that the sum of a group of encrypted values (votes) is verified and counted without revealing their contents [15].

"The encryption algorithm E() is homomorphic if; given E(x) and E(y), one can obtain E(x $\perp$ y) without decrypting x and y for some operation ($\perp$)", the homomorphic property can be represented by the following equation [15, 16]:

$$E_k(m_1) \perp E_k(m_2) = E_k(m_1 \perp m_2) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

Where; E_k represents the encryption done on messages m_1 and m_2 with some key k. $E_k(m_1) \perp E_k(m_2)$, is a calculation in a group G, whereas $E_k(m_1 \perp m_2)$ is a calculation in a group H. The `$\perp$', is a group operator corresponding to each group, and may be different for G and H.

The homomorphic property offers counting and verifying the votes without revealing the individual votes. After the election, the encrypted votes are accumulated into single, encrypted, sums. The election authorities then decrypt the resulting sums to obtain final results. According to the homomorphic property, these sums should be equal to the sums of the decrypted individual votes. Accordingly, counting the votes is done without learning the individual values of the votes. [17].

The operation ($\perp$) could be multiplication (*) and/or addition (+). Fully homomorphic cryptosystem support additive and multiplicative homomorphism (i. e. $\perp$ = * and +). The first such system is a lattice-based cryptosystem developed by Craig Gentry in 2009 [15]. Gentry's Ph.D. thesis [17], provides additional details. Fully homomorphic crypto systems are considered powerful systems in the sense that it could be run by an untrusted party without revealing its inputs and internal state. However, fully homomorphic schemes suffer from sharp increasing of computational time while increasing level of security. To obtain 2k (k is an integer from some range of candidates) security level, the ciphertext and computation time are high-degree polynomials in k [18, 19].

Partial homomorphic cryptosystem using either additive or multiplicative homomorphism. Some examples of partially homomorphic cryptosystems are; RSA and ElGamal which they are inhabit a multiplicative homomorphism. RSA algorithm uses multiplicative homomorphic encryption such that; given $c_i = E(m_i) = m_i^e \bmod N$; then for two plain information m_1, m_2 and their ciphers c_1, c_2, we can write [20]:

$$c_1 = m_1^e \bmod N$$
$$c_2 = m_2^e \bmod N$$
$$c_1 \cdot c_2 = m_1^e \cdot m_2^e \bmod N = (m_1 \cdot m_2)^e \bmod N$$

Where; c cipher message, m plaintext, e encryption key, and N modulo.

Another partial homomorphic schemes which uses addition operation only (+) such as Paillier [15]. The homomorphic property is preserved such that;

$$E_k(m_1) + E_k(m_2) = E_k(m_1 + m_2) \quad \ldots\ldots\ldots\ldots \quad (2)$$

Where, E_k represents the encryption done on messages m_1 and m_2 with some key k. $E_k(m_1) + E_k(m_2)$, is a calculation in a group G, whereas $E_k(m_1 + m_2)$ is a calculation in a group H. In partially homomorphic schemes, the homomorphic property is preserved and works as follows; votes are encrypted individually as they enters and added together to form summation counters for each candidate [21]. At the end of the election, the encrypted summation is decrypted to reveal the final results of the votes. Hence, no one can relate the votes to their voters. Partially homomorphic scheme is implemented in this research to maintain efficiency in terms of time and space [22, 23].

Figure-2 illustrates bulletin board in (a) regular plaintext votes and (b) in homographic encryption form. Using additive homomorphic property, only numbers in row of totals are decrypted, hence maintain votes privacy.

Voters	Candidates		
	X	Y	Z
V1	1	0	0
V2	0	1	0
V3	1	0	0
Totals	2	1	0

a. Regular bulletin board with; X, Y, Z represents Candidates, Vi represents Voters, and 0 or 1 represents votes for candidates

Voters	Candidates		
	X	Y	Z
V1	$C1_x$	$C1_y$	$C1_z$
V2	$C2_x$	$C2_y$	$C2_z$
V3	$C3_x$	$C3_y$	$C3_z$
Totals	C_x	C_y	C_z

b. bulletin board with homomorphic encryption, Cix, Ciy and Ciz represents encrypted votes using some public key crypto system

Figure 2. Bulletin Board a. Regular b. With Homomorphic Encryption

5. Integrated E-Election Protocol

The proposed protocol is a hybrid cryptographic algorithm which makes use of the benefits of both cryptographic approaches. It uses public key scheme for authentication and secure key exchange, and symmetric cryptographic scheme for encryption. The protocol assumes that the voters and candidates are registered in databases; voter database (DB_v) and candidates database (DB_c) respectively by election authority. The protocol comprises of four stages; voter validation stage, authentication stage, voting stage and election results stage. Figure-3 illustrates the proposed protocol. Stages are described as follows:

5.1 Stage I: Voter Validation Stage: In the voter validation stage, the voter starts by sending a request to the server asking for permission to vote, the server check the eligibility of the voter by searching voter database (DB_v) and make his decision to allow or reject the request. Stage I consists of steps 1-3 of the protocol shown below.

5.2 Stage II: Authentication Stage: Authentication stage consists of two parts; part one (stage II-A) authenticate server to voter and part two (stage II-B) authenticate voter to server. Both parts uses modified Diffie-Hellman protocol (D-H ZKP), described in table-1 above to perform mutual authentication and exchange shared secret key. In this stage the process may terminates if any party appears dishonest. By the end of this stage, if not terminates; voter proved legible, both server and voter are proved honest and server sends candidates list to voter. Stage II consists of steps 4-8 of the protocol shown below.

5.3 Stage III: Voting Stage: In voting stage, the voter receives candidates list from the server, submit his ballot and transmit it to the server. The server encrypts the ballot homomorphically using homomorphic encryption scheme discussed in section 4. Stage III consists of steps 9-13 of the protocol shown below.

5.4 Stage IV: Election Results: At the end of the election time announced by the election authority, the administrator performs homomorphic encryption on the ballots database and calculates the summation of votes for each candidate. Stage IV consists of step 14 of the protocol shown below.

5.5 proposed protocol

Notation		
b:		Ballots
CD_i:		Candidate
D-H ZKP:		Modified Diffie Hellman protocol for Zero-Knowledge proof
DB_v:		Voter Database
DB_c:		Candidates Database
DB_b:		Ballots Database
E( ):		Homomorphic Encryption
ID_v:		Voter Identification No.
K_s:		Shared Secret Key
M_i:		Error Messages
PU_s		Server Public Key
PR_s		Server Private Key
S:		Server
V:		Voter
Seq.	**From/to**	**Activity**
Stage I: Voter Validation Stage		
1	**V**	**Generate public keys (p, g), secret key (x), calculate $R_1 = g^x \bmod p$**
2	**V ➔ S**	**Request for voting (ID_v, p, g, R_1)**
3	**S**	**Check voter eligibility using voter database (DB_v); if not, terminate with message (M_1), else continue**
Stage II-A: Server Authentication Stage		
4	**S**	**Generate secret key (y), calculate; $K = (R_1)^y \bmod p$, $R_2 = g^y \bmod p$, and $C_1 = E(K, R_2)$**
5	**S ➔ V**	**Sends ($R_2 \| C_1$) to voter**
6	**V**	**Server Authentication; Calculates $K = (R_2)^x \bmod p$, decrypt ($R_2' = D(K, C_1)$) and verify ($R_2 = R_2'$) . If they matched; then proceeds, otherwise terminate with message (M_2) (dishonest verifier)**
Stage II-B: Voter Authentication		
7	**V ➔ S**	**Encrypt ($C_2 = E(K, R_1 \| R_2)$) and send it to server**
8	**S**	**Voter Authentication; Decrypt C_2 to get R_1' and R_2', ($R_1' = D(K, C_2)$) verify ($R_1 = R_1'$) ; If they matched then; proceeds, otherwise terminate with message (M_3) (dishonest prover)**
Stage III: Voting Stage		
9	**S➔ V**	**Send candidates list (CD_n) from candidates database (DB_c), public key (PU_s)**
10	**V ➔ S**	**Submit vote, validate vote if not valid; return with message (M_4)**
11	**V ➔ S**	**Encrypt vote and transmit to server $E(PU_s, CD_1)$, $E(PU_s, CD_2)$, …, $E(PU_s, CD_n)$**
12	**S**	**Store votes in Ballot database (DB_b)**
13	**S ➔ V**	**Send receipt of vote to voter (M_5)**
Stage IV: Election Results		
14	**A➔ S**	**If end of election, perform homomorphic encryption of ballots $E(PU_s, CD_1)$, $E(PU_s, CD_2)$, …, $E(PU_s, CD_n)$**
Messages explanation		
M_1		**Voter is not found in the database or not legible to vote**
M_2		**Server is not authenticated (dishonest)**
M_3		**Voter is not authenticated (dishonest)**
M_4		**Invalid vote**
M_5		**Vote has been recorded successfully**

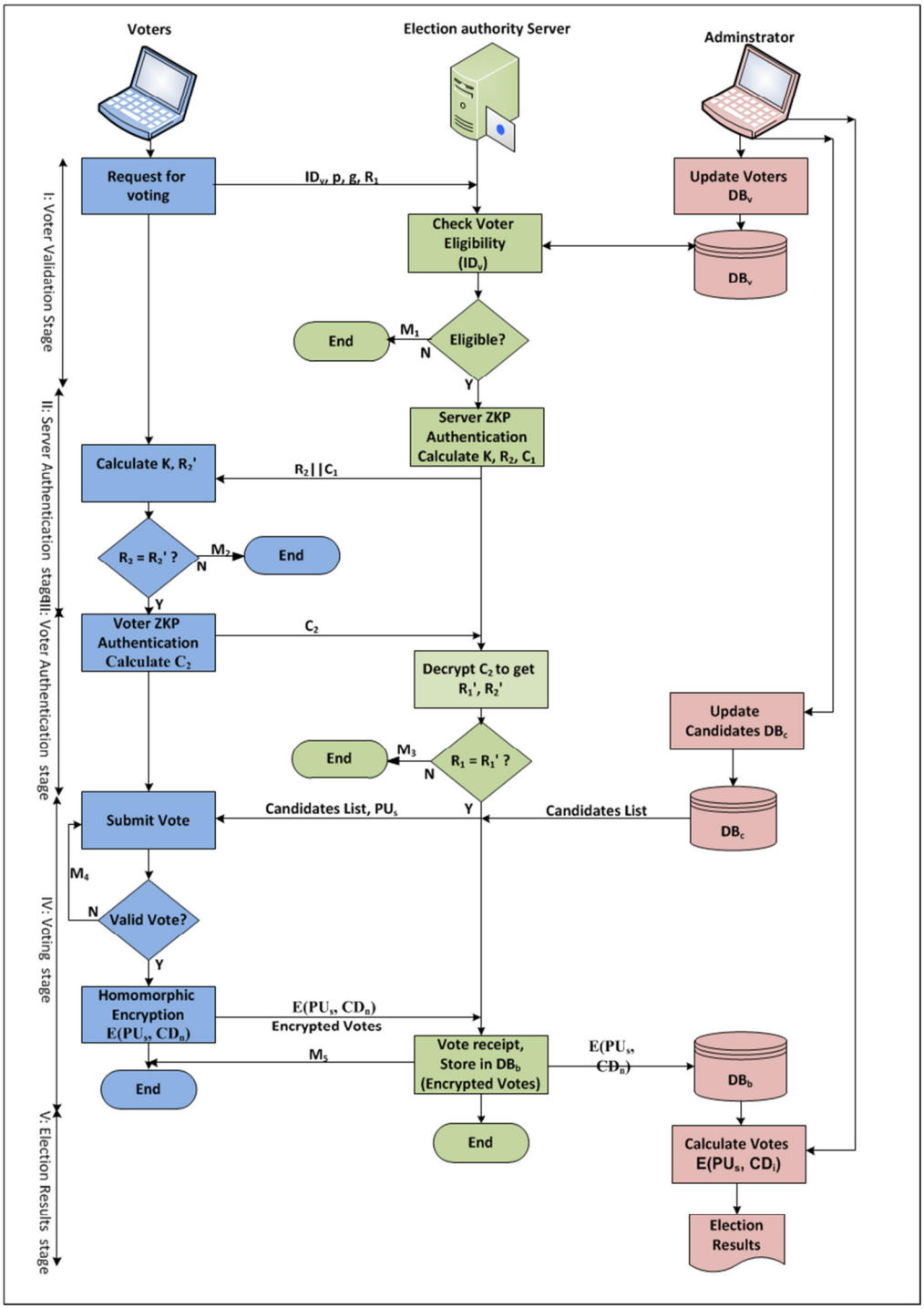

Figure 3. Integrated security Protocol for E-Election System

6. System Design

The three tiered architecture has been chosen to implement the proposed system to allow a central server location for all the application logic, database, which allows reuse, consistency, flexibility to potential changes and uniformity of applications in this environment. The three-tiered layering facilitates some of design and development problems, easy and efficient development work [24].

The three tiers are: the client interface tier (The presentation layer which supports the graphical user interface (GUI)), the application logic tier which performs business rules and logic of the application, and the database tier.

The interface layer in the proposed system offers the voter an easy and convenient entry to interact with the system and cast his vote. The application logic tier controls functions interaction, messages exchange and

manipulating of information flows. Finally, the data management is performed by the database layer, which store, manage and retrieve information needed for the application [24, 25]. Figure-4 illustrates the architecture of the proposed system.

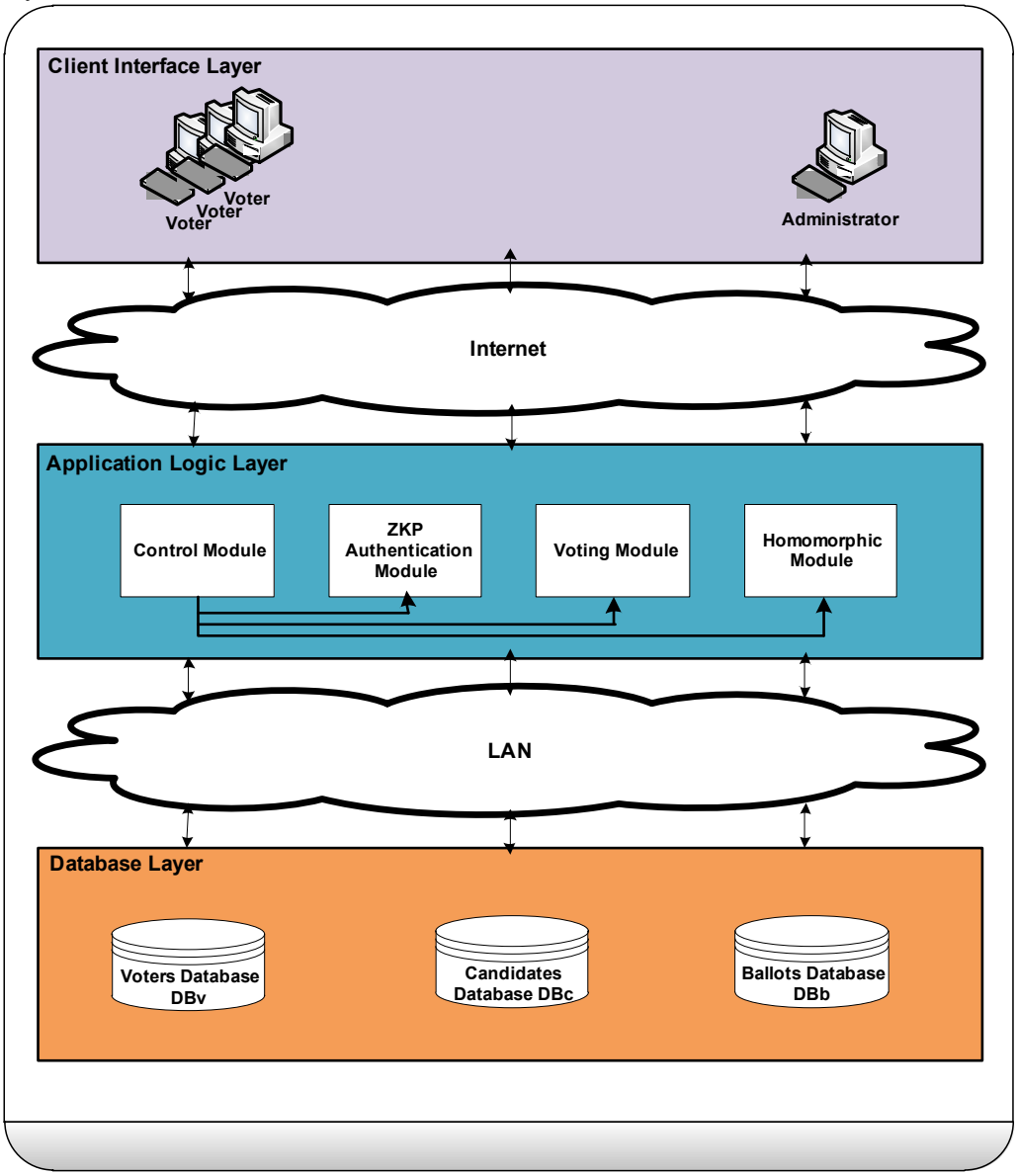

Figure 4. Secure E-Election System Architecture

6.1 Client Interface Layer

The client interface layer is used to facilitate the interaction between the clients with the application and manage the input/output data. The clients gain access to the application by using any existing web browser software. The client interface layer contains all required functions to receive data from the client, passing it to the logic layer and display information received from the logic layer (sending requests and receiving replies). Clients of the application are voters and administrator, as they described below.

6.1.1 Voter

Voter has mainly a user interface which communicates with application layer. The voter is supposed to enter his name and National Identification Number (NIDN) that has been previously delivered to him through election authority offices together with his randomly generated secrete number in a sealed card. The National Identification Number entered by the voter is compared with the stored national number in the database. Invalid name or national number will be rejected and a second chance for the voter will be given. Valid data will lead the voter to second stage.

The voter enters his secret number and communicates with the election authority server (honest verifier) through D-H ZKP protocol to establish a secure mutual authentication and prove to each other that they are

illegible and honest. Figure-5 illustrates communication screen between voter and the server. Additional information such as; (R_1, R_2, K) has been showed in the screen for test purposes. On successful completion of this step, candidate names are dynamically retrieved from the candidates database (DBc) and displayed to the voter. The user has an option to log out or cast his vote. If the voter completes the vote, the (has voted) field in voters database (DBv) is updated so that the user cannot log in once more.

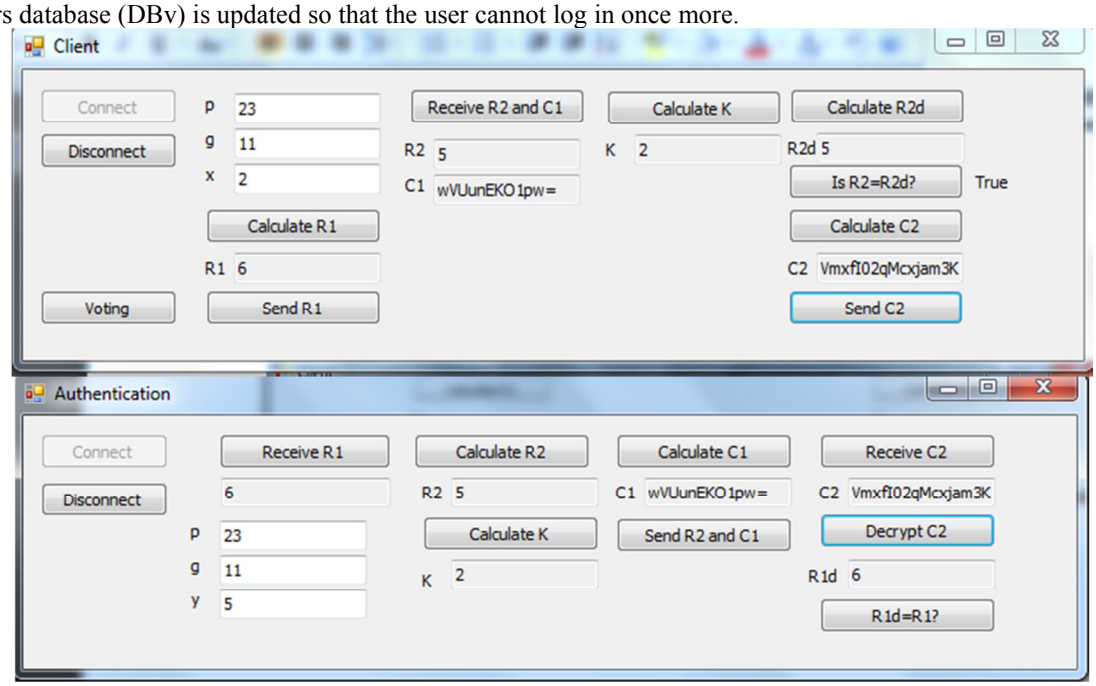

Figure 5. Voter-Server Authentication Communication Screen

6.1.2 Administrator

Administrator has a special username and password to login to the system. After he/she has logged into the system, he will get new screen which allow him to add, update or delete voters or candidates. At the end of election, the administrator can perform homomorphic encryption and calculate the results of the election. Administrator screen is shown in figure-6.

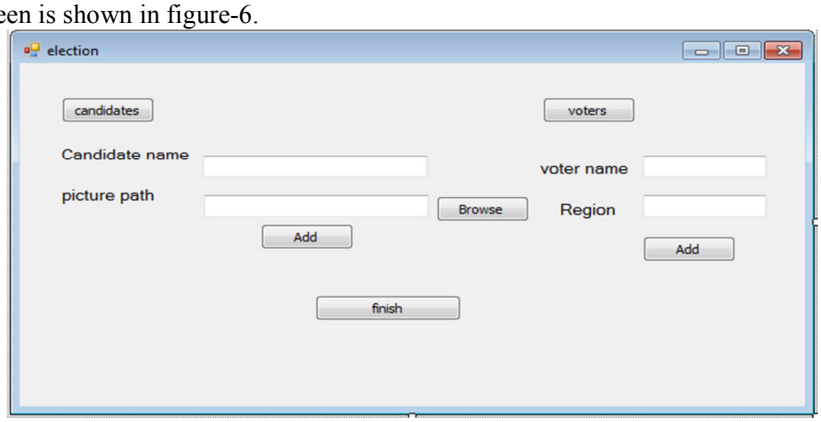

Figure 6. Administrator Screen

6.2 Application Logic Layer

The application logic layer is the middle layer, which connect the user interfaces and the underlying database, hiding technical details from the users. Modules in this layer receive clients request from the interface layer and perform appropriate function. Application logic layer consists of a control module and three functional modules. The control module controls the flow of execution and transferring required information between them and the database layer. Figure-7 illustrates control module flowchart. The functional modules are;

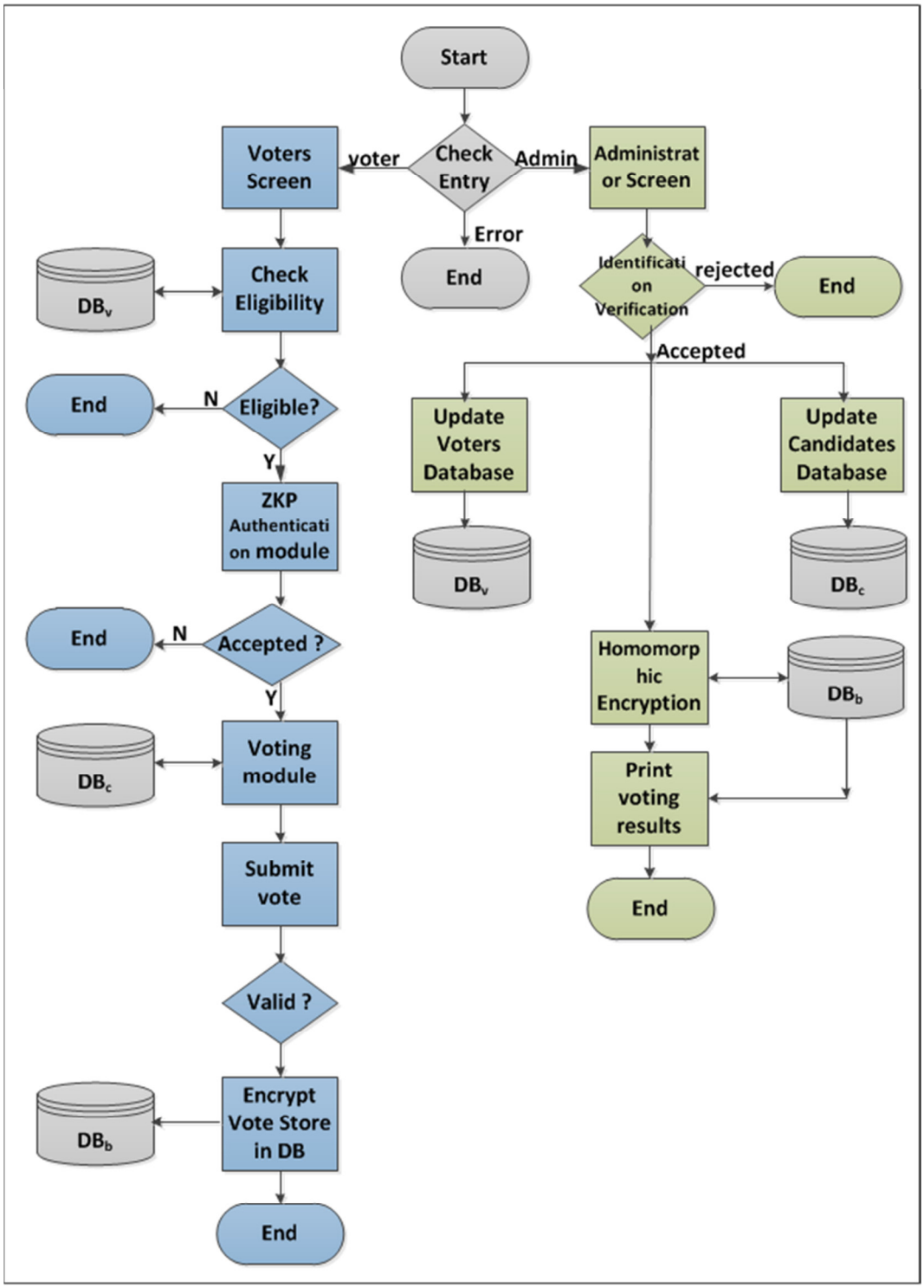

Figure 7. Control Module Flowchart

6.2.1 ZKP Authentication Module

Each voter must be authenticated through zero knowledge proof, using D-H ZKP protocol discussed earlier in section (3.2). A claimant (voter) and the verifier (server) will interact and communicating messages to prove to each other that they are illegible. During this process a secret key is generated in both sides securely. After exchanging messages, the voter will be either accepted and proceeds to the voting stage, or rejected and halt the system.

6.2.2 Voting Module

Voting module is designed to conduct the process of voting. Authenticated voter can selects his candidates in his/her region from a pull down menu and cast his vote. The votes will be transmitted to the server, encrypted and stored in the database using homomorphic module.

6.2.3 Homomorphic Module

Homomorphic encryption module is used to encrypt the votes and storing them in the database as illustrated in figure-8. In this module the sum of a group of encrypted votes is verified without revealing those encrypted votes using homomorphic module, after all the voting procedure completes**.** Homomorphic protocol is discussed in section (4).

At the end of the election process, this module will be executed again to process the votes and calculate the results homomorphically (i. e. using the homomorphic properties) to produce the final election results.

	VoterId	V1	V2	V3	V4	V5
▶	1	WuR5ZJNYEy/a...	y9jkhJmOF2RZ...	FytTHIk1856Tqj...	XCMgPgm2ILb...	IZMwA94HI4Jh...
	2	tJLJc0xFe7H1DK...	cYLckEUpkLczV...	a0DoikZBITj5be...	IAHYw3MIGVF...	o95Nat6RynNk...
	3	aou1bR8690HE...	hiWDvFk0qNLe...	LZYMi0pxvliJ0A...	VINj8hbORVBw...	d82yPumJzeZm...
	9	on/BVL9VU6gc...	eyUugMjDxgT2...	IK2BUk856maK...	0RpDS3Wn/8LX...	ni0ct2FR4jcSEQ...
*	NULL	NULL	NULL	NULL	NULL	NULL

Figure 8. Encrypted Votes Table

6.3 Database Layer

The database layer is used to store and retrieve application data according to the instructions of the application logic. Data can be processed and stored back in the database.

In this application the database consists of three tables. The first table called “Voter Database (DB_v)”, which stores the voter's data. Second table called "Candidate Database (DB_c)" used to store the candidates data. The third table called "Ballots Database DB_b)" used to store encrypted votes as illustrated in figure-8 above.

7. System Analysis

In this paper, two techniques have been used to design and implement an integrated security protocol for electronic election system and maintain its requirements. The first technique is a new zero knowledge protocol used to ensure secure mutual authentication between the voter "prover" and the election authority server "verifier" with the addition to exchange key securely. This technique is used to ensure that both parties are legal and hence prevent election fraud and maintain democracy which allows legal voters to vote once only. This technique has been proved theoretically and during the implementation test.

The second technique is homomorphic encryption protocol which ensures the confidentiality of the votes, in the sense that no one can relate any vote to a specific voter, besides, voter can't prove the way that he/she voted. Homomorphic properties have been proved throughout the related literatures.

The proposed e-election system prevents incomplete ballots and hence can be easily verified by comparing total voters with total votes as it appears in the final report of the system.

The proposed system is a web-based application and hence it is flexible and mobile which help voters to cast their vote from any location that they can cast their vote with minimal skills and no restrictions.

With the addition to the previous characteristics, the system provides reasonable reduction in the monetary cost, effort, and time required compared with traditional election systems.

8. Conclusions

This paper presents the design and implementation of an integrated security protocol for electronic election system assuming that voters and candidates has been already registered in an electronic databases. The proposed system consists of four stages; the first stage is voter validation stage to ensure his eligibility to vote. Second stage is to authenticate voters and election authority mutually with high level of security. In these phases, the system assures of security properties such as; authentication, trust, transparency, eligibility, and privacy.

The third stage is the voting stage, which is based on homomorphic cryptography techniques through which privacy and confidentiality are maintained. The fourth stage is the results stage, it consists of two steps; counting and auditing, which counts and audit the votes of each candidate. In this stage, homomorphic properties are applied to assure accuracy, verifiability, reliability, and fairness.

Other election requirements, Convenience, flexibility and mobility are also maintained throughout the design of the system using flexible three-tier web based architecture.

We conclude that the proposed system has met security requirements while maintaining electronic election properties.

References

[1] Ivan Damgard, et al, (2003), "***The theory and Implementation of Electronic Voting System***", Advances in Information Security Vol. 7, Springer, ISBN 1-4020-7301-1.

[2] Feng Hao, Peter Y. A. Ryan, (2016), "***Real-world Electronic Voting: Design, Analysis and Deployment***", CRC press.

[3] Yong-Sork, Her et al., (2005), "***Ballot-Cancellation Scheme In E-Voting System***", E-Government Workshop '05 (eGOV05), Brunel University, UK.

[4] Kapali Viswanathan,(2005), "***Towards Robustly Secure Electronic Voting Systems***", the 2nd Annual IIT Kanpur Hacker's Workshop (IITKHACK05). India.

[5] Jitendra Kurmi Ankur Sodhi,(2015), " ***A Survey of Zero-Knowledge Proof for Authentication***", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 1.

[6] Endre Bangerter, et al**,** (2009), **"*On the Design and Implementation of Efficient Zero-Knowledge Proofs of Knowledge***", Proceedings of the 2nd ECRYPT Conference on Software Performance Enhancement for Encryption and Decryption and Cryptographic Compilers (SPEED-CC'09), Berlin, Germany.

[7] Mohr, Austin, (2010), "***A Survey of Zero-Knowledge Proofs with Applications to Cryptography***". http://austinmohr.com/work/files/zkp.pdf, cited January.

[8] Maurer Ueli, (2009), "***Unifying Zero-Knowledge Proofs of Knowledge***", Africacrypt, pp: 272–286.

[9] Krantz, Steven G., (2007), **"*Zero Knowledge Proofs*",** AIM Preprint Series, Volume 10 No. 46.

[10] Michael Backes and Dominique Unruha, (2010), "***Computational Soundness of Symbolic Zero-Knowledge Proofs***", Journal of Computer Security, Vol. 18, No. 6, pp: 1077-1155.

[11] Forouzan, Behrouz A.(2008), "***Cryptography and Network Security***", McGraw-Hill, Int. Ed.

[12] Fischer, Michael J., (2010), "***Cryptography and Computer Security***", Lecture Notes, Department of Computer Science, Yale University**.**

[13] Kizza, Joseph M, (2010), "***Feige-Fiat-Shamir ZKP Scheme Revisited***", International Journal of Computing and ICT Research, Vol. 4, No. 1, pp: 9-19.

[14] Mahmood Khalel Ibrahem, (2012), "***Modification of Diffie–Hellman Key Exchange Algorithm for Zero Knowledge Proof***", International Conference on Future Communication Networks, Baghdad, pp 147-152, IEEE.

[15] Liam Morris, (2013), ***"Analysis of Partially and Fully Homomorphic Encryption"***, Department of Computer Science, Rochester Institute of Technology, Rochester, New York.

[16] Craig Gentry, (2009), "***Fully Homomorphic Encryption Using Ideal Lattices***", Proc. Of the 41st ACM Symposium on Theory of Computing (STOC), pp: 169-178, NY.

[17] Craig Gentry, (2009), ***"A Fully Homomorphic Encryption Scheme***", Ph. D. theses, Stanford University.

[18] Louis J. M. Aslett, Pedro M. Esperan¸ca and Chris C. Holmes, (2015), "***A review of homomorphic encryption tools for encrypted statistical machine learning***", Cornell University, **arXiv:1508.06574**.

[19] Nigel Smart and Fre Vercauteren, (2010), "***Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes***", Springer Berlin Heidelberg, Vol. 60, PP: 420-443.

[20] N. Vamshinath, K. Ruth Ramya, Sai Krishna, P. Gopi Bhaskar, Geofrey L. Mwaseba, and Tai-hoon Kim, (2015), "***Homomorphic Encryption for Cluster in Cloud",*** International Journal of Security and Its Applications, Vol. 9, No. pp. 319-324.

[21] S. Ramachandram, R.Sridevi, and P. Srivani, (2013), ***"A Survey Report On Partially Homomorphic Encryption Techniques In Cloud Computing"***, International Journal of Engineering Research & Technology (IJERT), Vol. 2 Issue 12.

[22] Ruchita Tekade, Prof. Reena Kharat, Varsha Magade, Marjina Shaikh, and Pallavi Mendhe, (2016), "***E-Voting System using Visual Cryptography & Homomorphic Encryption***", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 5, Issue 1.

[23] Raghul.H, Ramagopal.R.N, Saravanan.B, Guhapriya.T and , Anitha.R, (2015), "***Data Security in Federated Cloud Environment using Homomorphic Encryption Technique***", International Journal of Emerging Technology and Advanced Engineering, Volume 5, Issue 4.

[24] Shu-Ching Chen, et al., (2003), "***A Three-Tier System Architecture Design and Development for Hurricane Occurrence Simulation***", Proc. of the IEEE Int. Conference on Information Technology Research and Education (ITRE), USA.

[25] G.O. Ofori-Dwumfuo and E. Paatey, (2013), **"*The Design of an Electronic Voting System***", International Journal of Recent Technology and Engineering (IJRTE), ISSN: 2277-3878, Volume-2, Issue-1.