International Journal of Security and Its Applications
Vol. 11, No. 5 (2017), pp.59-68
http://dx.doi.org/10.14257/ijsia.2017.11.5.06

E-Voting System using Homomorphic Encryption in a Cloud Based Environment

Rachna Jain [1], Sushila Madan [2], Bindu Garg [3],
Yogeesh Kapila[3] and Abhishek Gupta[3]

[1]Department of Computer Science, Banasthali Vidyapith University, Banasthali, India
[2]Department of CSE, Delhi University, Delhi, India
[3]Department of CSE, Bharati Vidyapeeth College of Engineering, Delhi, India
rachnabtec@gmail.com, sushila.madan@gmail.com, bindugarg80@gmail.com, yogeeshkapila.94@gmail.com, abhishekgupta.p16@imi.edu

Abstract

Cryptography as we know it, is among the most discussed topics in the security world. Any transaction, financial or social, any data, corporate or private is not secure in an environment such as the cloud, where everything is connected to everything. The only way to save anything or to make a transaction securely is to make it meaningless to the rest of the world. It can be made meaningless when converted to some other form and this some other form can only be obtained through encryption. In this paper we discuss various techniques of homomorphic encryption applied on cloud computing and the need of security over the cloud by citing relevant statistics. We then propose and implement a secure E-voting system using the paillier encryption scheme. The main goal of this research is to design a secure voting system using the internet platform to communicate between the voting system and voters.

Keywords*: Cloud Computing, Security, Homomorphic Encryption, Privacy*

1. Introduction

Cloud computing has caught the eyeball of the scientific as well as organizational communities due to its vast amount of uses and scope. A cloud server has an advantage of providing scalability and on-demand instant service to the user application. For instance an individual web hosting company hosts the web app on a single server/instance, so if that server goes down then the web app also stops working, but this is not the case with a cloud server. A cloud server such as Windows Azure gives the user flexibility to allot the number of instances, CPUs along with their computation power (RAM, HD space *etc.*,) to the deployed website or the app. Hence hosting is easier and more efficient on a cloud. Similarly, the organizations for storage of their data, need not worry about buying or maintaining the equipment to run large data centre's for storing their data. They can now buy services provided by a cloud vendor to efficiently and cheaply store their data and access it anytime, anywhere they require it.

Cloud can provide its services in 4 ways, namely: SaaS (Software as a service), PaaS (Platform as a service), IaaS (Infrastructure as a service), DaaS(Desktop as a service).In SaaS, basically an end user need not install a program on its machine in order to access or run it. The program or the software is installed on the server which the end users can access independently over the internet. So, the software is provided as a service by the cloud vendors. All google apps are an example of SaaS. In PaaS, a user is provided with a platform and given necessary services to build its own application, having features of the cloud such as scalability, availability etc. Basically a user gets an OS, program, database and a web server according to the need of the application development. Google app

* Corresponding Author

ISSN: 1738-9976 IJSIA

engine is an example of this type. In IaaS, a user is provided with the basic resources such as buying hardware for the machine and using those resources the user can do whatever it wants. The most common example of IaaS is buying of AWS by organizations to store and manage their data over the cloud. Basically organizations pay to run networks, virtualized servers and storage from the cloud. In DaaS, a user is provided with a virtual desktop. This means that the back end of a virtual desktop infrastructure is provided as service by the cloud. In this type of service the users need not worry about data storage, upgrades backup etc. All the backend maintenance and costs is handled by the cloud service provider.

However, organizations can't simply put all of their vital data on some cloud server due to security concerns. What if some malicious users got hold of that data from the server and tampered it? So data storage on the cloud can't be reliable until some security measures are applied. Therefore, the data was first encrypted and then stored on the server. Hence storing data on the cloud securely by encrypting it and finding new and efficient ways of doing so has been a challenge for the scientific community. Encrypting the data has the advantage that the data remains unintelligible to all unless it is decrypted, therefore protecting the data from malicious users and hackers. Thus data on the cloud is made secure by this. However, security comes at the cost of increased computation time and cumbersome procedure of decrypting the data always before its use, even by the original user. In order to solve this problem a technique of Homomorphic Encryption was devised. What's so special about homomorphic encryption? Homomorphic encryption is a way of encrypting the plain text in such a way such that if any operation is carried out on the cipher text and then decrypt the result, it is the same as if the operation had been carried out on plain text. This technique of encryption has huge applications as it provides the flexibility of operating on cipher texts instead of plain texts, which in turn lets a user data be stored on some server in encrypted form and the third party can manipulate the data and provide the result (encrypted) on the users behalf, not ever knowing what the real data is. With homomorphic encryption computations can be carried out straight away on cipher texts. Secondly, an added layer of security is provided for the user data. Now the data would be hidden even from the third party cloud server providers, hence even more secure data. Hence homomorphic encryption has huge applications especially in cloud computing. There are basically two types of homomorphic encryption schemes namely: partially homomorphic encryption schemes and fully homomorphic encryption schemes. In a partially homomorphic system, system allows only one operation to be performed on it such that it still retains its homomorphic property. For instance RSA is multiplicative cryptosystem whereas Paillier is additive cryptosystem. In a fully homomorphic system such as one given by Gentry, the system allows for more than one operation to be applied on it and the system still retains its homomorphic property. It is far more powerful than partially homomorphic cryptosystem.

Internet voting systems could be popular and successful in coming days. Each voter can vote for candidates remotely through the internet. Voters can easily open websites, software or applications on their computers or smart phones to vote anytime, anywhere. However, their voting system can be hacked if it does not have any algorithm or protocol to protect it. Attackers can easily attack on the packets of voters which are moving on the internet and they can change information that is sent over voting server. Additionally, they can interrupt votes which are sent by voters over the internet. Therefore, researchers are being design methods to defend against such attacks using cryptography algorithms so that vote can be encrypted before sending it to the server. This idea was first mentioned in 1981. E-voting process consists of several steps, many of which can be found in the traditional voting. It is more accessible and convenient for voters than the traditional voting. The big advantage is that it does not require voters to present at the poll station. Another advantage, voters can vote from anywhere as soon as they meet the requirements to vote. However, to implement a secure voting system is difficult. Here are some requirements for a secure voting system to be adapted.

(i)Auditability: Reliable and demonstrably authentic election records should be generated.

(ii)Authentication: Only authorized voters should be able to vote. Voters can request to change their information if necessary. A voter's information can be collected from the birth certificate or other similar document.

(iii)Reliability*:* Systems should work robustly to prevent electoral frauds or attacks from outside the system. The E-voting system should be very reliable. The result of an election must be correct and shows up to voters after the election ends.

(iv)Voter Confirmation*:* The voting system should send an email to the voter to confirm that his or her vote has been received by the system correctly. At the end, the result of the voting also can be sent to the voters so that they will know which candidate is a winner.

*(v)*Uniqueness: No voter should be able to vote more than once. This requirement is necessary and same as the traditional voting. This feature prevents coercion or buying votes.

(vi)Accuracy: Voting systems should record the votes correctly. After the vote is recorded, the voter should be able to check if his vote was recorded or not. If he tries to re-vote or change the vote, the system should prevent that.

(vii)Integrity: A vote is just for one time decision and cannot be changed. No one should be able to determine how the voter voted, so no one can change the vote.

(viii)Verifiability*:* Verifying that the votes are correctly counted in the final tally should be possible. This feature is mandatory. It has to be audited and tested before the system is used.

E-voting is the most convenient to vote. It is excellent on equality, building a trust in electoral organization, adding reliability to election results and increasing the overall efficiency of the polling process. However, to build an E-voting system that can work perfectly over the internet is a big challenge. Two major challenges that can be considered are security and supporting a large number of voters. Most E-voting solutions cannot work with large number of voters. E-voting also faces security issues because voters vote and send their votes over the internet which is not a controlled environment. In addition, voting under an electronic voting system occurs automatically without any human supervision. Certainly, voters would like to vote by paper votes at a post office or polling station rather than through an E-voting system because they do not trust the E-voting as their ballots are transferred over the Internet. For example, in 2012, the federal government allowed to use the E-voting for Canton of zurich voters. The voters had three options to vote: ballot voting, postal voting, and Internet voting. However, only 20 percent of the votes were cast through the internet voting. To address some of these challenges, in this research, we have developed an E-voting system that can allow a large number of voters. Specifically, paillier algorithm is used to support a large number of voters and secure a vote when it is counted. To secure a vote on the internet, the secure socket server/client protocol is used to transfer information over the internet. Section 1 is Introduction. Section 2 describes the Related work. In Section 3, Proposed model of secure voting using paillier is discussed. Section 4 is Conclusion and Future Scope.

2. Related Work

RSA [Ron Rivest, Adi Shamir, Leonard Adleman] [1] gives very popular public key cryptographic technique which is based on the factoring problem.RSA is partially homomorphic encryption scheme, supporting multiplicative operations. It uses a key size of 1024 to 4094 bit. However, one of the drawbacks to the RSA encryption algorithm is that it leaks a singular plaintext bit in every cipher text. This bit is known as the Jacobi symbol of the plaintext, and is either 1or 0. Due to this an attacker can guess what the original plaintext was. This drawback was countered by the Goldwasser Micali scheme.

Pascal Paillier [2] in 1999 gives paillier cryptosystem which is a probabilistic public key cryptographic technique. The cryptosystem is based upon computing the nth residue classes which is considered as a difficult problem.

Taher Elgamal [3] in 1985 gives public key cryptographic technique which has the DiffieHellman key exchange algorithm as its base. This algorithm can be defined over any cyclic group C. Its security depends upon the diffculty of a certain problem in C related to computing discrete logarithms. ElGamal Cryptosystem performs multiplicative homomorphic encryption property.

Craig Gentry [4, 5] in 2009 proposed a scheme which allows all types of calculation on the data stored in the cloud, the optimal approach would be of fully homomorphic encryption which is able to execute all types of operations on encrypted data without the need of decryption. In 2009 Craig Gentry of IBM proposed the first encryption system "fully homomorphic" that computes any number of additions and multiplications and thus calculate any type of function on encrypted data. The first fully homomorphic system was given by Gentry in 2009 and it was a lattice based system. An application of fully homomorphic encryption scheme could be submitting an encrypted query to any search engine, which would process the encrypted query and display results in an encrypted from, which can be decrypted by the user. By way of this, even the search engine or the Cloud service provider wouldn't know what the query was and what the result was, hence enforcing data privacy and security.

Josh Cohen Benaloh [6] in 1994 proposed Benaloh Cryptosystem which is basically an extension of the Goldwasser-Micali Cryptosystem. This scheme overcomes that limitation by allowing blocks of data being encrypted at a single pass instead of bit by bit in Goldwasser-Micali. The security of the scheme relies on the problem of higher residuosity. This scheme exhibits additive homomorphic property.

D. Boneh *et. al.,* [7] in 2005 first proposed a fully homomorphic system, but it allowed only a single multiplication operation along with the addition. Thus this was called a somewhat homomorphic. The additive property is same as of the ElGamal variant. This scheme uses elliptic curves of Composite order for its implementation.

Katja Schmidt-Samoa [8] suggested Schmidt-Samoa Cryptosystem which is based on the integer factorization problem, similar to Rabin Cryptosystem. However, it has the advantage of not producing any ambiguity in the decryption process, like in Rabin scheme, at the cost of encryption speed. It has an advantage over RSA as it is based on much harder integer factorization however, encryption is slow due to exponent function.

Michael O. Rabin [9] in 1979 proposed rabin cryptosystem which is similar to the RSA cryptosystem. It is also based upon the problem of factorization. The problem it relies on has been proved as hard as integer factorization. It is commonly referred as a four-to-one function encryption scheme.

Sha Goldwasser & Silvio Micali [10] in 1982 proposed Goldwasser Micali Cryptosystem. It is unique in the sense that it was the first probabilistic encryption algorithm designed which was proven to be secure under ideal cryptographic assumptions. The probabilistic nature of the algorithm ensures randomness, i.e. it generates a different cipher text each time it is used to encrypt a plaintext. It encrypts the plaintext bit by bit. Its security depends on the problem of quadratic residuosity modulo. This cryptosystem exhibits additive homomorphic property.

3. Proposed Model of Secure Voting using Paillier

The traditional voting system has certain drawbacks, having a lot of steps and not easy for voters who don't have easy access to the voting machines. Moreover, there is also a big security concern included such as what if someone changed the votes for a favoured candidate or tampering of votes. Use of Encryption for storing and computing result of voting on Cloud has emerged as a successful way of carrying out the voting procedure [11]. The internet voting systems are the most successful and popular today. The voters can anytime, any-where and from any device connected to the internet can access the application for voting and cast their votes safely. The paillier cryptosystem being an additive homomorphic cryptosystem, it can be used for the secure voting process [12]. All the votes gathered could be stored on a secure cloud server, using the paillier encryption technique and then in order to compute the result of the voting process, the encrypted votes can be added homomorphically to produce the result. The decryption of the encrypted result would give the winner of the elections.

Algorithm for Casting and Calculating Votes

Let NC = Candidate Number

Now if vote is cast for candidate NC then,

i. It is first converted to a suitable No using base as 10 as:

$x = 10^{2*(N_C-1)}$

ii. Similarly all such values are computed and added together.

$X = x_a + x_b + x_c + x_d + \cdots$

iii. From *X* it can then comfortably take out votes for each participating candidate

Note: For each candidate the total no. of votes is depicted by two digits.

This is the basic algorithm for vote generation and counting. This procedure can be strengthened by use of paillier encryption to encrypt the votes and then adding the votes homomorphically to produce the 'ENCRYPTED SUM OF VOTES'.

This encrypted sum upon decryption and some further computations gives the votes for a particular candidate and hence produces the result.

Using paillier encryption along with voting algorithm is explained stepwise as:

Steps

1) Key Generation

Public Key

Choose $n = p.q$ where $p \& q$ are large primes and $gcd\big(pq.(p-1, q-1)\big) = 1$.

Select a no. g at random such that $g \in Z^*_{n^2}$ and $gcd\left(\frac{g^{\delta} mod(n^2-1)}{n}, n\right)$=1. The prime numbers $p \& q$ should be chosen according to need.

Private Key

$\delta = lcm(p-1, q-1)$ $\delta(n)$ being the Carmichael Function.

Modular Multiplicative Inverse: $\mu = \big(L(g^{\delta} mod\, n^2)\big)$-1 $mod\, n$

Where L is defined as $L(u) = \frac{u-1}{n}$

2) Encryption

$E(m_i) = c_i = g^{m_i} * r_i^n \; mod\, n^2$ where $r \in Z^*_n$

In order to compute the result and get sum of the votes, there is need to find the product of all the encrypted votes modulo n^2

T = total votes therefore,

Tally of votes = $\prod_{i=1}^{T} c_i \; mod\, n^2$

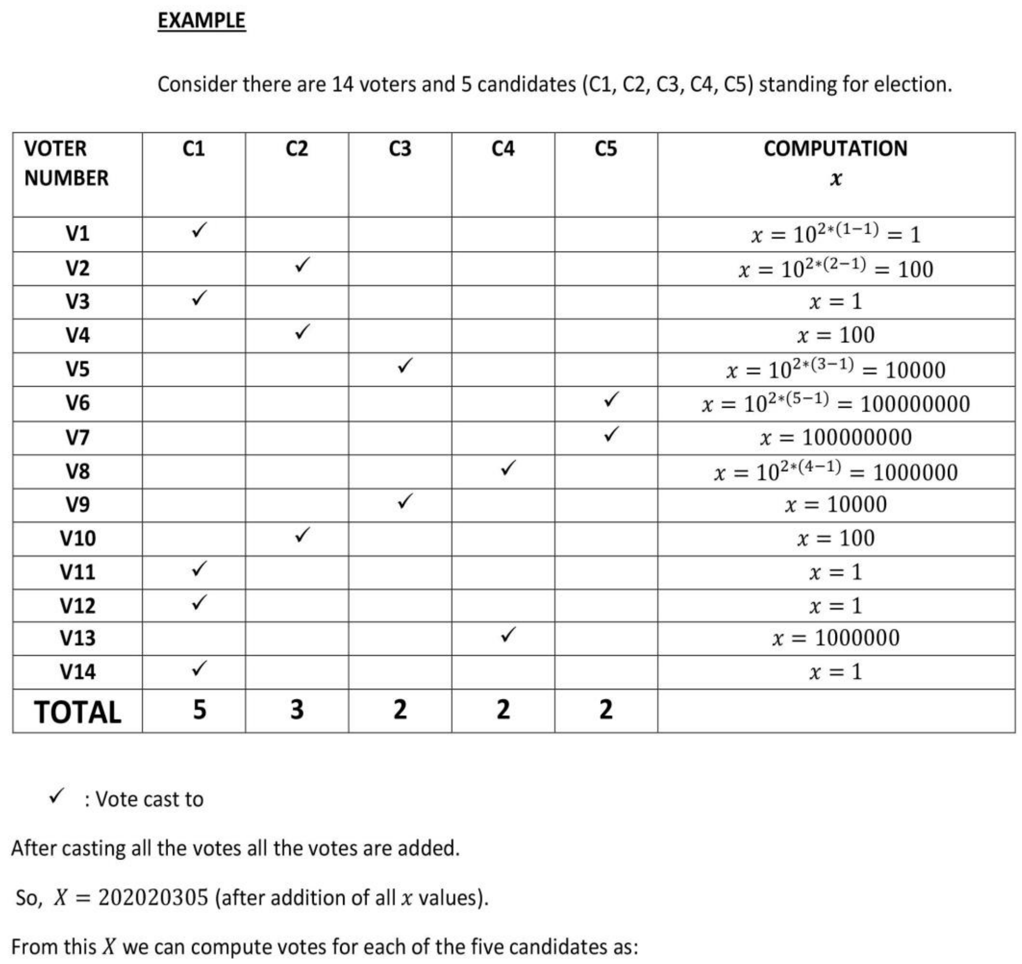
3) Decryption

According to homomorphic properties of Paillier:

$m = L\left(g^{\delta} mod\ n^{2}\right) * \mu\ mod\ n$

Decryption of Tally of votes is as: = $\sum_{i=1}^{T} m_i\ mod\ n$

At the end we can get the tally of votes by which we can extract the total number of votes for each candidate y using division and the remainder theorem. The below mentioned example would help get a clear picture.



EXAMPLE

Consider there are 14 voters and 5 candidates (C1, C2, C3, C4, C5) standing for election.

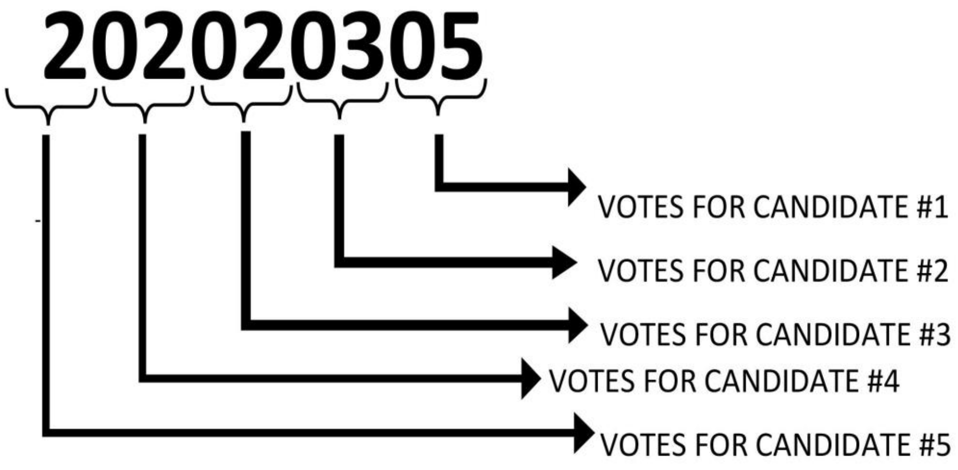
VOTER NUMBER	C1	C2	C3	C4	C5	COMPUTATION x
V1	✓					$x = 10^{2*(1-1)} = 1$
V2		✓				$x = 10^{2*(2-1)} = 100$
V3	✓					$x = 1$
V4		✓				$x = 100$
V5			✓			$x = 10^{2*(3-1)} = 10000$
V6					✓	$x = 10^{2*(5-1)} = 100000000$
V7					✓	$x = 100000000$
V8				✓		$x = 10^{2*(4-1)} = 1000000$
V9			✓			$x = 10000$
V10		✓				$x = 100$
V11	✓					$x = 1$
V12	✓					$x = 1$
V13				✓		$x = 1000000$
V14	✓					$x = 1$
TOTAL	**5**	**3**	**2**	**2**	**2**	

✓ : Vote cast to

After casting all the votes all the votes are added.

So, $X = 202020305$ (after addition of all x values).

From this X we can compute votes for each of the five candidates as:

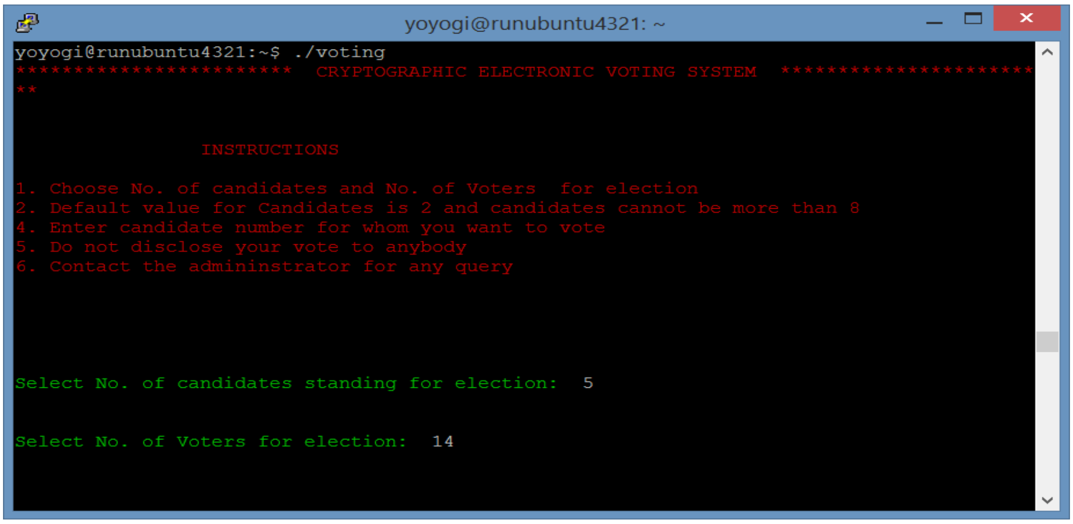


Result of the Voting Process after all computations is as follows:
Votes for Candidate #1 = 5 (WINNER)
Votes for Candidate #2 = 3
Votes for Candidate #3 = 2
Votes for Candidate #4 = 2
Votes for Candidate #5 = 2

Now this algorithm can be strengthened using the paillier cryptosystem technique. All the casted votes could first be encrypted and stored on the cloud server and then they could be added homomorphically to produce the encrypted result. The decryption and further computation of the result would then finally give the winner of the election process.

Below are some screenshots for the implementation of the example explained above.

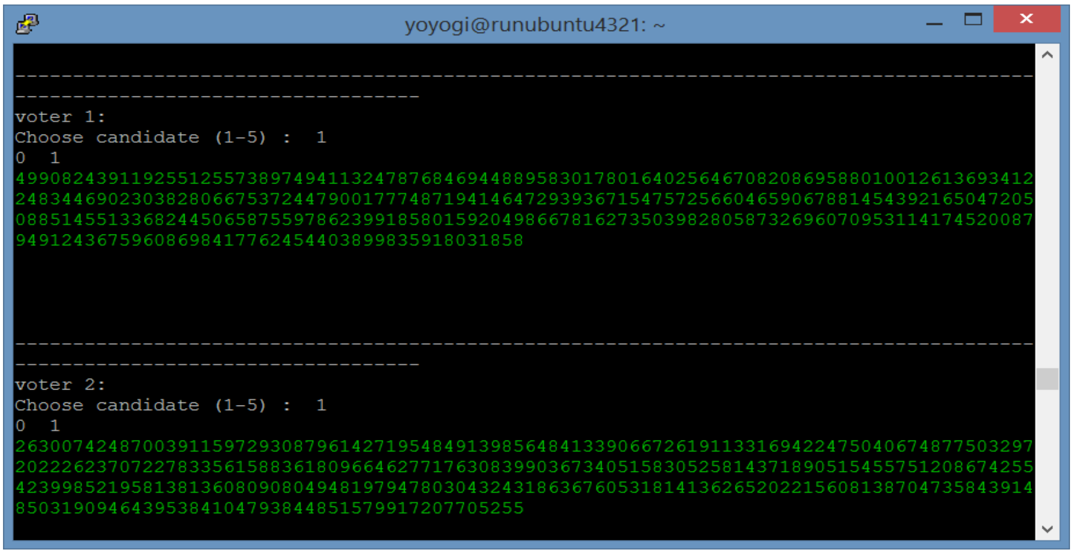
Step 1: First step is to provide input for the voting system which is selection of number of candidates standing for election and number of voters.



```
yoyogi@runubuntu4321:~$ ./voting
************************  CRYPTOGRAPHIC ELECTRONIC VOTING SYSTEM  **********************
**


                INSTRUCTIONS

1. Choose No. of candidates and No. of Voters  for election
2. Default value for Candidates is 2 and candidates cannot be more than 8
4. Enter candidate number for whom you want to vote
5. Do not disclose your vote to anybody
6. Contact the admininstrator for any query




Select No. of candidates standing for election:  5


Select No. of Voters for election:  14
```

Figure 1. Input for the System

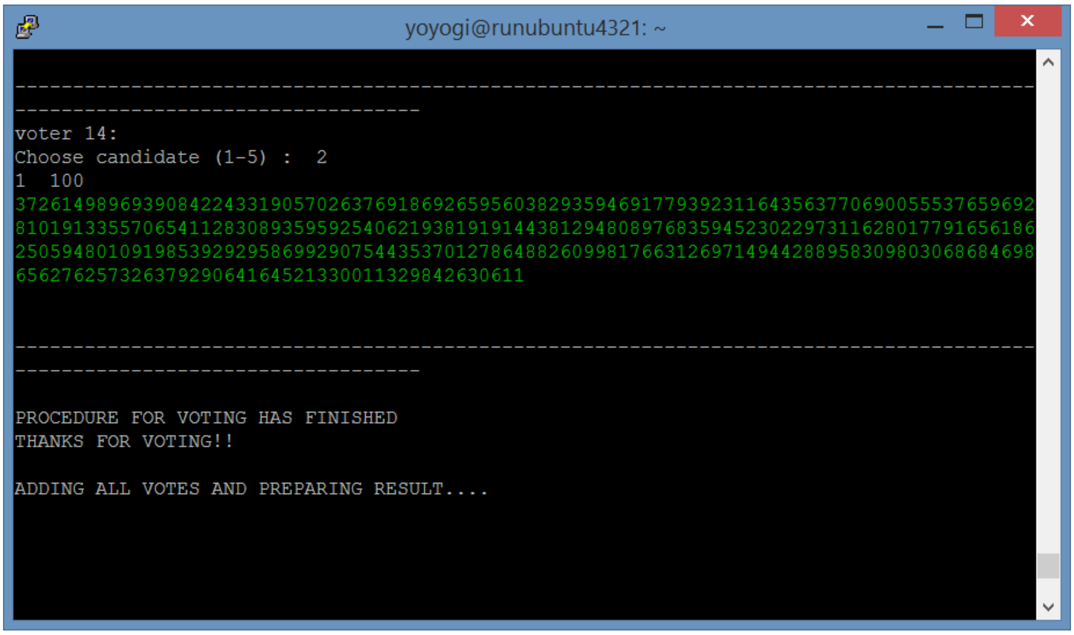
Step 2: In this step the input is encrypted into an array which is shown in green using the paillier algorithm.



```

----------------------------------------------------------------------------------------
-----------------------------------
voter 1:
Choose candidate (1-5) :  1
0  1
4990824391192551255738974941132478768469448895830178016402564670820869588010012613693412
2483446902303828066753724479001777487194146472939367154757256604659067881454392165047205
0885145513368244506587559786239918580159204986678162735039828058732696070953114174520087
94912436759608698417762454403899835918031858




----------------------------------------------------------------------------------------
-----------------------------------
voter 2:
Choose candidate (1-5) :  1
0  1
2630074248700391159729308796142719548491398564841339066726191133169422475040674877503297
2022262370722783356158836180966462771763083990367340515830525814371890515455751208674255
4239985219581381360809080494819794780304324318636760531814136265202215608138704735843914
85031909464395384104793844851579917207705255
```

Figure 2. Voting Procedure

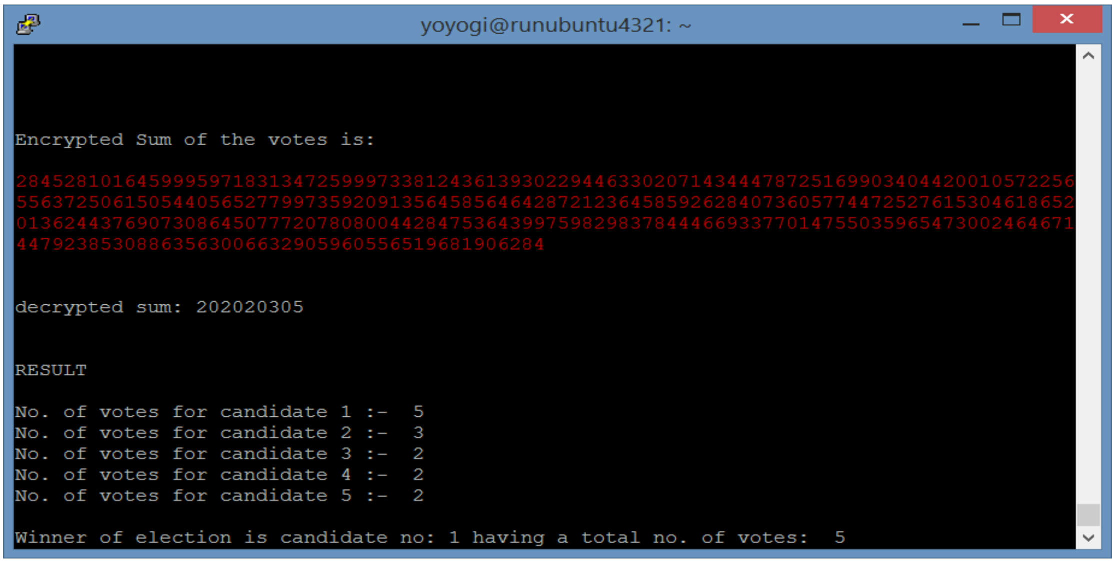
Step 3: In this step end of voting procedure is shown as every candidate voted for voters.



```
----------------------------------------------------------------------------------------
-----------------------------------
voter 14:
Choose candidate (1-5) :  2
1  100
3726149896939084224331905702637691869265956038293594691779392311643563770690055537659692
8101913355706541128308935959254062193819191443812948089768359452302297311628017791656186
2505948010919853929295869929075443537012786488260998176631269714944288958309803068684698
65627625732637929064164521330011329842630611


----------------------------------------------------------------------------------------
-----------------------------------

PROCEDURE FOR VOTING HAS FINISHED
THANKS FOR VOTING!!

ADDING ALL VOTES AND PREPARING RESULT....
```

Figure 3. End of Voting Procedure

Step 4: In this step result of voting system is shown which is the encrypted sum in red that is produced homomorphically adding all the encrypted votes and then it is decrypted to finally extract the result of the elections.



```
Encrypted Sum of the votes is:

2845281016459995971831347259997338124361393022944633020714344478725169903404420010572256
5563725061505440565277997359209135645856464287212364585926284073605774472527615304618652
0136244376907308645077720780800442847536439975982983784446693377014755035965473002464671
44792385308863563006632905960556519681906284


decrypted sum: 202020305


RESULT

No. of votes for candidate 1 :-  5
No. of votes for candidate 2 :-  3
No. of votes for candidate 3 :-  2
No. of votes for candidate 4 :-  2
No. of votes for candidate 5 :-  2

Winner of election is candidate no: 1 having a total no. of votes:  5
```

Figure 4. Result of Voting

(Note: The encrypted sum (red) is produced homomorphically adding all the encrypted votes and then it is decrypted to finally extract the result of the elections)

4. Conclusion and Future Scope

We have seen that why security over the cloud is imperative and so we have discussed techniques of making the data on the cloud secure by use of various homomorphic encryption schemes. The security would be ten folds if the schemes had been fully homomorphic schemes, in which any computation could have been performed on the cipher text without compromising the confidentiality of raw data. We also propose and implement a secure e-voting system over the cloud using paillier cryptosystem. In this system the number of voters can be large enough for practical implementations and the same can be further increased too according to the need and scenario. Moreover, we have implemented the SSH protocol for transactions to the cloud, hence providing a secure and encrypted medium for the process. The system having been deployed on the cloud and SSH protocol supports platform independence *i.e.*, it can be easily accessible from anywhere using any device connected to the internet such as browsers, mobiles iPad etc. Hence making the voting process an easy task yet robust and secure. In future a proper GUI could be made for E-Voting to make it easier and interactive for the users. Also we are going to analyse the use of octonion algebra in homomorphic cryptosystems. By the use of octonions the cryptosystem can be made more robust and efficient. It would reduce the key size but at the same time increase the security of the cryptosystem. Hence the use of octonions for implementing cryptographic security over the cloud could be studied and also implemented for the e-voting system.

References

[1] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. Communications of the ACM, 21(2): 120-126, 1978. Computer Science, pages 223-238. Springer, 1999.

[2] Pascal Paillier, Public-key cryptosystems based on composite degree residuosity classes, In 18th Annual Eurocrypt Conference (EU-ROCRYPT'99), Prague, Czech Republic, volume 1592, 1999.

[3] O. Ugus, A. Hessler, and D. Westhoff, "Performance of additive homomorphic EC-Elgamal encryption for TinyPEDS." Fachgespräch Sensornetzwerke **(2007)**, pp. 55.

[4] C. Gentry, "A Fully Homomorphic Encryption Scheme", **(2009)**.

[5] C. Gentry, S. Halevi, "Implementing Gentrys Fully-Homomorphic Encryption Scheme" **(2011)**.

[6] J. Benaloh, "Dense Probabilistic Encryption". (PS). Workshop on Selected Areas of Cryptography. **(1994)**, pp. 120-128.

[7] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts", TCC 2005, crypto.stanford.edu/ dabo/pubs/papers/2dnf.pdf.

[8] A New Rabin-type Trapdoor Permutation Equivalent to Factoring and Its Applications.

[9] M. Rabin, "Digitalized Signatures and Public-Key Functions as Intractable as Factorization", MIT Laboratory for Computer Science, January **(1979)**.

[10] S. Goldwasser, S. Micali, "Probabilistic encryption and how to play mental poker keeping secret all partial information." In Proceedings of 14th Symposium on Theory of Computing, **(1982)**, pp. 365-377.

[11] S. Choinyambuu, "HS09 Homomorphic Tallying with Paillier". Retrieved from: http://security.hsr.ch/msevote/seminarpapers/HS09 Homo-morphic Tallying with Paillier, **(2010)**.

[12] "Secure Voting System Using Paillier Homomorphic Encryption", Cuong Ngo, **(2014)**.

Authors

Rachna Jain, is currently working as an Assistant Professor in CSE Department of Bharati Vidyapeeth College of Engineering since Aug'2007. Prior to this she worked as a lecturer in N.C College of Engineering for one year. She completed her B.Tech(CSE) with honors from Kurukshetra University and ME (CTA) from Delhi University. She is Pursuing Phd from Banasthali Vidyapith. She has a total teaching experience of around 9 years. She is an active member of CSE Department and played an important role in various departmental and college level activities. She is also a co-coordinator of Cloud Computing In-house Summer Training programmed for second year students organized in the college every year. She is working closely in the areas of Speech, robotics, Network Security and Cloud Computing. She has also authored many papers, published in IEEE International Conferences, and International Journals of computer applications(IJCA), International Journal of Science Technology And management(IJSTM). She has also attended a number of workshops and faculty development programs inside and outside college.

Sushila Madan, is an Associate Professor at Lady Shri Ram College for Women, University of Delhi, Delhi. She is Ph.D. from Delhi University, M. Tech. (software systems) from BITS-PILANI and M. Sc. in Applied Mathematics from IIT Delhi. She has also submitted a project funded by UGC titled "Security Risk Management in E-Commerce". Dr. Madan has authored books on IT, Multimedia and web technology, Management Information and Control Systems, E-commerce and Essential PC tools. To her credit there are a number of research papers which have appeared in leading journals; some of which have been presented in conferences and included in conference proceedings; leading national level magazines and in-house journals of corporate sector. Moreover, white papers on the Internet are also available. Also she is a member of the Computer Society of India.

Bindu Garg, With 11+ years of experience in academia and Industry, she joined Bharati Vidyapeeth college of Engineering as Associate professor in August 2013. She completed her PhD in CSE from Jamia Millia Islamia, Central University at New Delhi under guidance of Prof. M.M Sufyan Beg and Prof. A.Q Ansari. She did her B.Tech (CSE) and M.Tech (CSE) with honors. Prior to Bharati Vidyapeeth, she worked as acclaimed faculty in India and abroad. She was youngest to be awarded with Dr RAJENDRA PRASAD AWARD" by International Eminent Educationists Forum of India on 5th Sept 2008 to acknowledge meritorious achievements in the field of education, commitment of teaching and social work dedication. She was founder of NGO-YUKTI to serve weaker section of society. She is proficient in English, Hindi and French(DELF A1 Certified). She has attended multiple FDP's (Faculty development programs) and professional trainings. She has membership of many societies like: IEEE, CSI and ISTE. She is in review committee of numerous national & international conferences/journals. Her key research areas are: Soft Computing, Analysis & designing of algorithms and Time Series Prediction. she has 22 research publications in her credit.

Yogeesh Kapila, is currently a Master's student at University of South California, Specializing in computer security. He completed his Bachelors from Bharati Vidyapeeth College of Engineering during which he was introduced to the domain of security and encryption.

Abhishek Gupta, is currently a Management student at International Management Institute.He completed his Bachelors from Bharati Vidyapeeth College of Engineering during which he was introduced to the domain of security and encryption.