International Journal of Future Generation Communication and Networking
Vol. 13, No. 1s, (2020), pp. 331-- 341

Securing Pharmaceutical Data Using Homomorphic Encryption

Dhruv Bhagadia[1], Mustafa Bhanpurawala[2], Devansh Dalal[3], Pratik Kanani[4]
[1,2,3,4] *Dwarkadas J. Sanghvi College of Engineering*
University of Mumbai
Mumbai – 400056, India.

Abstract

In an age where efficient storage and access of data is pivotal for the functioning of any application whatsoever, cloud computing becomes an important field of study in providing unlimited storage and efficient access of data. But along with its benefits, cloud storage invites various kinds of problems mostly pertaining to the security of user data on the cloud. In the current scenario the user is at the mercy of the third party (the one responsible for providing cloud storage) to take the necessary measures to protect its client's data from unauthorized access and also uphold the trust between the client and itself that no kind of mishandling or misuse of data would be done. Recent research pertaining to cloud computing is majorly focused on designing and implementing cryptographic techniques that enables user to be responsible for the privacy of his/her data that is being stored on the cloud server. Homomorphic Encryption is one of those cryptographic techniques that allows manipulation of data in its encrypted form without having to decrypt it. Pharmaceutical industries is a domain where there are high chances of security breaches and applying Homomorphic Encryption can help minimize them.

Keywords: *Cloud Computing, Homomorphic Encryption, Paillier Cryptosystem, Pharmaceutical Industry, Security and privacy.*

1. Introduction

Vast amounts of sensitive and confidential information is generated by the pharmaceutical industry in the form of medicine formulas, employee information, medical records. The industry needs to ensure that their information is kept secure from cybercrime.

Cyber criminals can have devastating effects on the pharmaceutical industry by accessing or destroying their information. Hence it is crucial to budget a sufficient amount of money and resources to maintain a functional Information Security Management. The process of securing data from unauthorized access, exposure or destruction is known as Information Security.

Attacks by cyber criminals can cost billions of dollars to organizations and consumers. Based on the data received from the Identity Theft Resource Centre, medical/healthcare breaches accounted 46 percent [1] of reported data breaches in the United States. Some companies don't even realise they are facing such a threat and thus take no security measures whatsoever.

An employee or a contractor, someone inside the organization, poses a greater threat to the pharmaceutical industry than a hacker. They are a liability since they are entrusted to handle sensitive data of the company. Dealing with malicious and resentful insiders is much more difficult than outsiders. Insiders can cause harm by committing a mistake and causing accidental disclosure, abusing their record access privileges, accessing information for profit or attacking the system voluntarily.

One of the examples [1] of an inside attack is of Shionogi, a Japanese pharmaceutical company. Their former employee, Jason Cornish, obtained passwords while working at Shionogi and later used it to delete contents of fifteen virtual hosts. Due to this attack the company could not deliver products, send emails or cut checks. The damage caused was estimated at 800,000 million dollars.

Another major threat faced by the pharmaceutical industry is the production of counterfeit drugs. Counterfeit drugs have fatal consequences, including patient deaths. Unlike other types of counterfeited products such as watches or sunglasses, it is not easy for consumers to

ISSN: 2233-7857 IJFGCN

determine if they are buying a counterfeit drug or not. Some of the examples of counterfeit drugs are Epogen and Lipitor. Leakage of sensitive information regarding the drug can lead to the production of counterfeit drugs.

Due to these reasons it has become mandatory for the pharmaceutical industry to develop an Information Security Management system. An efficient encryption algorithm can be used to prevent hackers from obtaining unauthorized access to sensitive data. Also insiders will not be given access to the complete data, they will only be given access to data pertaining to their department. This will ensure that insiders cannot cause harm by leaking information.

Based on our research we found that Homomorphic Encryption finds no application whatsoever in pharmaceutical industries. Given the sensitivity of the information that is handled in pharmaceutical industries we aim to design a working system that deals with such information the Homomorphic way. We try to implement a web-application that allows employees of a particular pharmaceutical company to update sensitive data homomorphically that's stored on the cloud or database server. The scope of our application pertains to keeping track of the components used in developing a particular medicine. Here the employee on a daily basis updates the amount of particular components used, through the web portal which is reflected on the data stored in the databases homomorphically i.e. without involving any kind of decryption process on the data stored on the cloud/database server.

2. Homomorphic Encryption

2.1. What is Homomorphic Encryption?

Homomorphic Encryption is an encryption technique that allows specific computations on data in its encrypted form (cipher texts). The result is always encrypted data (cipher text), which results from operations performed on encrypted data of plain text. Homomorphic Encryption provides the basis of manipulating data without decrypting it first [2]. There are variations of Homomorphic Encryption based on the type of operational capabilities it can provide. Following are the different types of Homomorphic Encryption Systems:

- Fully Homomorphic Encryption System.
- Partially Homomorphic Encryption System.
- Somewhat Homomorphic Encryption System.

2.2. Partially Homomorphic Encryption

A type of Homomorphic Encryption system that allows a single type of operation on the encrypted data that is either addition or multiplication. Though bounded by its operational capabilities, these algorithms are more practical and can be efficiently executed in desired applications. Partially Homomorphic Encryption finds its major application in secure e-voting systems, banking system and secure computations on data stored in the cloud.

Some of the systems that make use of Partial Homomorphic properties are as follows:

- ElGamal encryption
- Benaloh
- Paillier

3. Paillier Cryptosystem Algorithm

Paillier Cryptosystem is a cryptographic algorithm that makes use of Homomorphic properties to ensure the security of the user's data. As discussed above Paillier Cryptosystem is a Partially Homomorphic System. It only allows the addition of two encrypted data. Paillier Encryption System has the following properties [3]:

ISSN: 2233-7857 IJFGCN

- It's a public key system, which allows anyone with the knowledge of public key to perform encryption on the desired data. Decryption on the other hand can be performed by those who have the knowledge of private keys which are mostly the trusted parties.
- It is highly probabilistic, that is there is no way to figure out if two cipher texts are encryption of the same plaintext are not.
- It allows Homomorphic addition of two encrypted data.

3.1. Working of Paillier Algorithm

Based on an in-depth analysis we can state that the complexity of Paillier Cryptosystem depends on the randomness of prime numbers selected. Following are the steps involved in Paillier Algorithm [4]:

- Select two large prime numbers, p and q.
- Calculate the product x = p*q, such that gcd(n, ɵ(n))=1, where ɵ(n) is Euler Function.
- Choose a random number g, where g has order multiple of n or gcd(L(g^{λ} mod n^2) n) =1, where L(t) = $\frac{t-1}{n}$ and λ=lcm(p-1, q-1).
- The public key is composed of (g, n), while the private key is composed of (p,q,λ).
- The Encryption of a message m < n is given by:

$$c=g^m r^n \bmod n^2$$

- The Decryption of cipher text c is given by:

$$m=(L(c^{\lambda} \bmod n^2)(L(g^{\lambda} \bmod n^2))^{-1})\bmod n$$

4. Review of Literature

M. Nassar et al. [3] presented a research paper explaining the importance of Paillier Algorithm in cloud applications and its implementation. This paper forms the basis of our understanding regarding the Paillier Algorithm and how its Homomorphic properties can be exploited to implement our desired application. The Paillier Algorithm used in our application is designed along the lines of the research carried out in this paper.

V. Sidorov and W. K. Ng, wrote a paper on performance Evaluation of Oblivious Data Processing Emulated with Partially Homomorphic Encryption Schemes [5]. There are various Homomorphic Encryption Algorithms available. To decide which one suits our application better, we analyzed the above research paper. The above mentioned paper states that there exist no empirical way of comparing the algorithms and based on one's own application and the type of operations involved , a particular algorithm should be selected.

Das, Debasis [6] presented a paper on secured cloud computing using Homomorphic Encryption that highlighted loopholes present in securities of cloud computing. Besides this, it introduced us to the idea of making use of homomorphic algorithms to perform operations on already encrypted data present on cloud which in turn prevents from online security breaches. The paper proposes a scheme and compares the results with existing standards, integrating homomorphic operations with multi-party calculations.

Shao et al. [7] presented a paper regarding an efficient way of implementing AES algorithm Since Homomorphic Encryption doesn't allow us to encrypt textual data in accordance with our proposed methodology. Therefore we analyze this paper to gain insight on the working of the AES algorithm which is used in our proposed methodology to encrypt the textual data on the server.

5. Methodology

Our proposed scheme tries to strengthen the shortcomings present in the security of pharmaceutical industries. The two primary users of our application are Manager and Employees present in pharmaceutical industries. Employees/Researchers perform the work of experimentations while developing medicine, because of which there are frequent changes to the quantities of the components present in the medicines. These pieces of information are very

ISSN: 2233-7857 IJFGCN

confidential and hence should be visible to only authorized people like the manager. The manager holds all the rights like adding/removing employees, watching detailed descriptions of medicines, etc. All the keys used for cryptographic operations are stored securely (encrypted form) on the servers adding an extra bit of security to the application. Let us analyze how each user namely the Manager and the Employees interact with the system.

5.1. Manager

Manager of a particular department has the responsibility of keeping a watch on the progress in the development of a particular medicine and details of the components. Whenever a manager logins he/she the keys which are required to view the obscured information are first fetched and then all entries are decrypted so as to show the manager the details. The flowchart shows the steps followed for a manager:

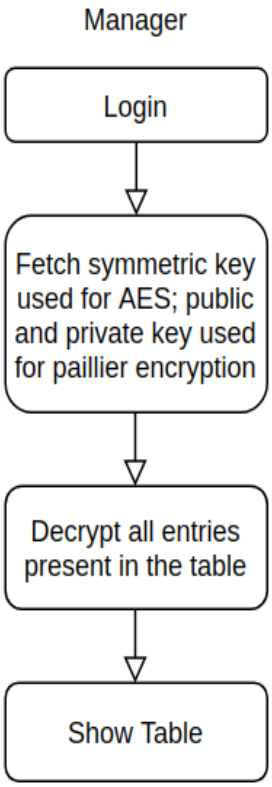


Figure 1. Use Case Diagram of Manager

5.2. Employee

The second category of users of our system are the researchers/Employee who are actually responsible for the development of a medicine. These people shouldn't be shown the entire details of the medicine because that in-turn would lead to security breaches. The flow of steps begin by the researcher having to enter the details of his research like component, quantity and price. After that an algorithm runs which first checks for whether the database already has that particular component or not. If yes then quantity and prizes are added Homomorphically else a new entry is made into the table. The keys required to do this task are first fetched from the server and then the procedure shown in the flowchart is followed.

ISSN: 2233-7857 IJFGCN

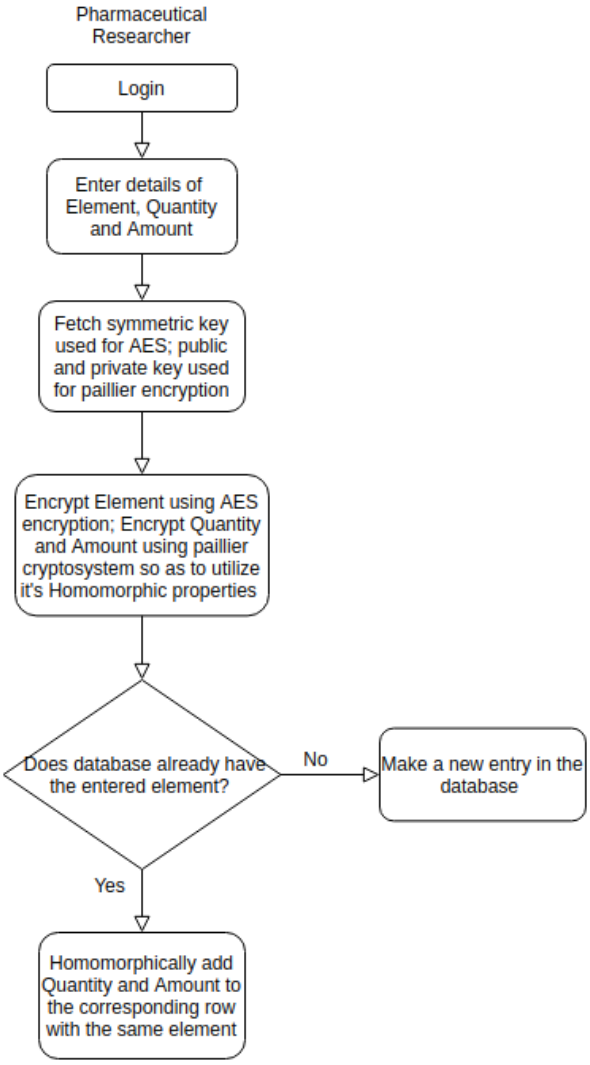


Figure 2. Use Case Diagram of Employee

6. Implementation

We make use of two algorithms to implement our methodology namely, Paillier Cryptosystem and AES algorithm.

6.1. Paillier Cryptosystem Algorithm

For our application we make use of the Paillier Cryptosystem algorithm to encrypt the numerical data involved in our web-application that is the cost of component and quantity of the component. Apart from encryption it allows to make manipulations (mostly addition) on the data in its encrypted form homomorphically.

Based on the algorithm discussed earlier we designed a set of functions in order to implement the Homomorphic properties of the Paillier Cryptosystem algorithm. These functions are as follows:

- gen_prime(n): generates an n bit prime integer.
- gen_keypair(n): generates key-pair (pub_key, priv_key)
- encrypt_data(pub_key, data): encrypts *data* into ciphertext using pub_key

ISSN: 2233-7857 IJFGCN

- add_data(pub_key, c1, c2): adds ciphertext c1 and ciphertext c2 homomorphically
- decrypt_data(priv_key, pub_key, cipher): decrypts the cipher.

6.2. AES Algorithm

AES Algorithm is implemented in our web-application to encrypt the textual data in the system. AES is an encryption standard adopted by the US government and is a symmetric encryption algorithm that uses a single key to encrypt as well as decrypt a particular set of data [7].

AES encryption involves following steps carried out for a particular set of iterations (10, 12 or 14):

- Sub-Bytes
- Shift-Row
- Mix-Columns
- Add Round-key

Apart from encrypting data, we use the AES algorithm to encrypt certain files containing keys sensitive to the operation being carried out in our web-application. Following are the functions designed by us to implement the functionalities of the AES algorithm:

- gen_key(): generates a random 16 byte key for encryption and decryption process.
- encrypt(message, key): encrypts the *message* using *key* to generate a ciphertext.
- decrypt(key, ciphertext): decrypts the *ciphertext* using *key* to generate the original plaintext.
- encrypt_file(key, file_name): encrypts the file *file_name* using *key*.
- decrypt_file(key, file_name): decrypts the file *file_name* using *key*.

6.3. Key Management on the Server

Our application focuses on two users, managers and researchers/employees. In any case the keys need to be stored securely on the servers. There are various ways that we explored and then came to a conclusion of making use of the most suitable one. Few methods are [8]:

6.3.1. HSM (Hardware Security Module)

It is a separate hardware module in which sensitive keys are stored. The major advantage that this method provides is it's distal from the actual server. The keys are stored on the HSM and never leaves the appliance un-encrypted. Any cryptographic activity that involves the private key is carried out internally within the device. However, considering the overhead of extra hardware we didn't opt for this option in our project.

6.3.2. Encrypted file with key supplied on runtime

This method provides additional layers of security with protection against offline attacks. This method requires the user to supply the secret key used to encrypt files containing cryptographic keys before using it. Here the secret is already shared with the user while registering with the system, he has to use the same secret in the future to fetch cryptographic keys. The only way a hacker can get the cryptographic keys is by using the secret. Since the secret is not stored on the server but provided by the user on runtime and stored in RAM it gets more difficult for the hacker to penetrate and fetch the secret and hence more secure. However, the major issue with this kind of architecture is sharing the key beforehand which in-turn increases the security concerns and the inconvenience caused to the user to always provide the secret to the system before using it. Besides in our system which incorporates multiple medicines, an Employee of one medicine should not make changes to another medicine and this method could not provide us with that feature and hence didn't opt for it.

6.3.3. Encrypted File with the key not shared with the user

ISSN: 2233-7857 IJFGCN

Another solution for securely storing keys on the server is to store it in a particular file and then encrypting that file with the user password or equivalent key. The major advantages provided by such a service is getting more control while logging into the system to get results. For managers, we make use of their own passwords while for Employees we make use of the medicine name for which that particular Employee was added by the manager. We take into account the security concerns of the medicine name and hence store it in encrypted form on the server. This kind of functionality allows Employee to make changes data pertaining to the medicine for which he was added by the Manager.

7. Design

7.1. Manager

- The manager registers by entering his username, password and the medicine name. After registering two files are created - manager.txt and employee.txt. A set of public and private keys are created for paillier homomorphic encryption, and one AES key is generated. All the keys are added to the manager.txt file. The password entered by the manager will be used to encrypt the manager.txt file. Employee.txt file will consist of the public key of paillier homomorphic encryption and the AES key. The employee.txt file is encrypted using the medicine name.

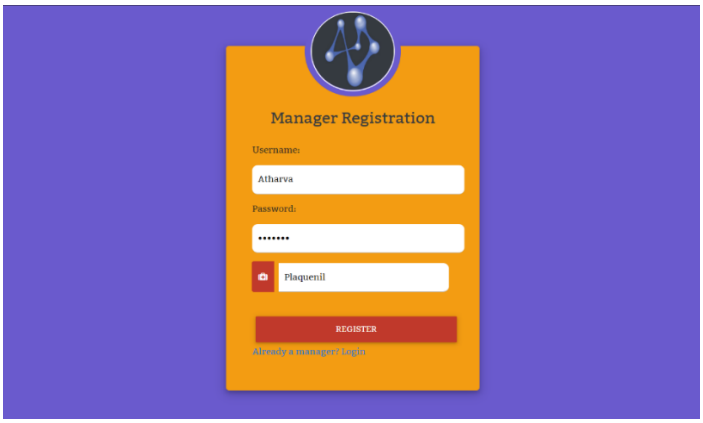
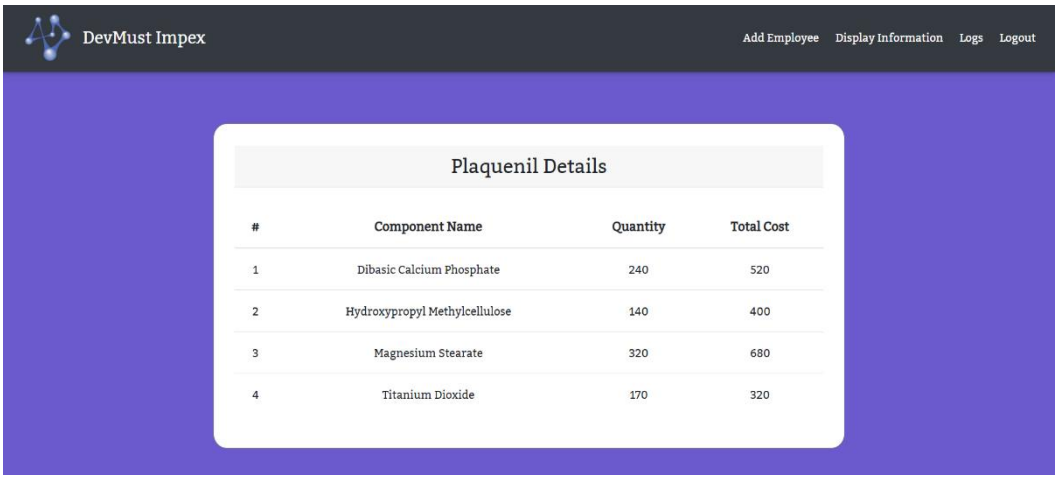


Figure 3. Manager Registration Page

- When the manager registers/logs in, his password will be used to decrypt the file containing all the keys. The details of the component name, quantity and cost will be displayed to the manager. For this, the private keys will be fetched from the keys file. The encrypted values will be fetched from the database. The private keys will then be used to decrypt the encrypted component name, quantity and cost, and then it will be displayed to the manager in a table form.



ISSN: 2233-7857 IJFGCN

Figure 4. Display of Medicine Components

- The manager has the option to add employees in his department. The manager can add an employee by entering the employee's name and email address on this page. Upon submitting this form, the email address mentioned in the form will be used to send the employee a mail where in his login credentials will be mentioned. The employee can then use these details to login.

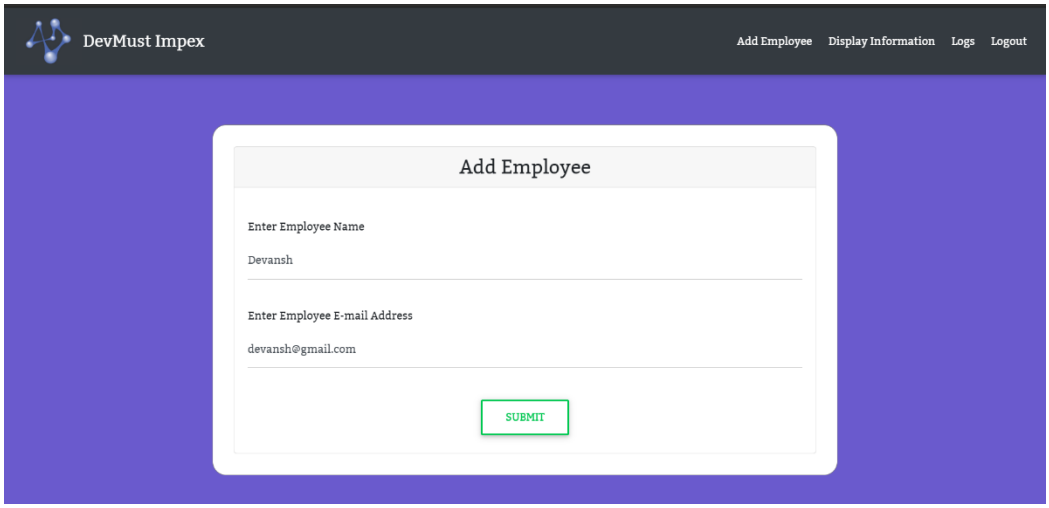


Figure 5. Graphical Interface to add Employees

7.2. Employee

The employee has access to this page where he can enter the details of the medicine component used, such as component name, quantity and cost of the component. The component name entered will be encrypted using AES Encryption Method. The component cost and quantity will be encrypted using Homomorphic Encryption Method. The component cost and quantity values can be further used for analysis such as calculating the total expenditure, to keep track of the quantity of components used. Homomorphic Encryption enables performing operations on the encrypted information for such analysis.

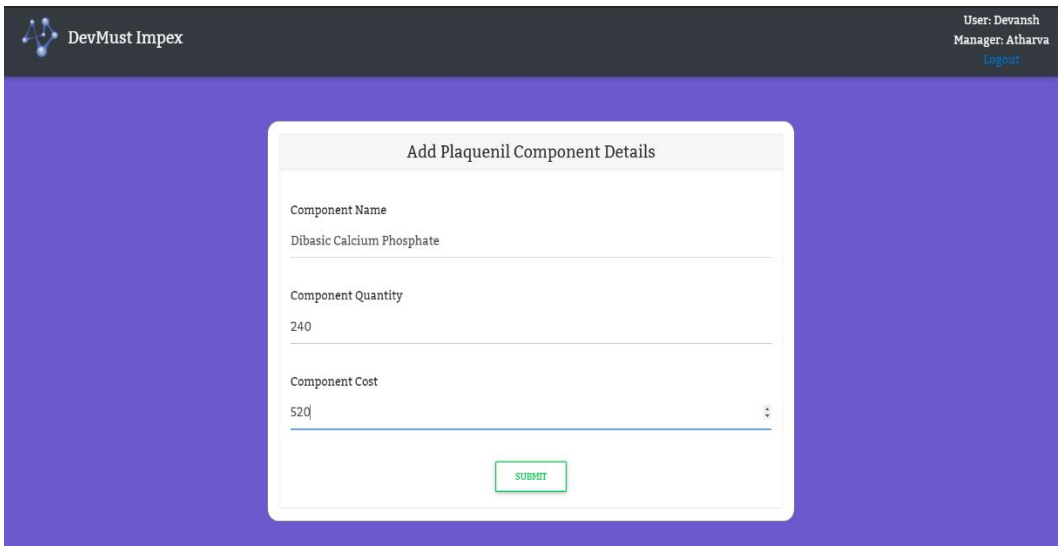


Figure 6. Graphical Interface to enter Medicine details

There are 2 possible cases when the employee submits the data:

- If the database does not contain an instance of the encrypted name, a new entry will be created in the database and the encrypted values of the component name, quantity and cost will be stored.
- If the database contains an instance of the encrypted name, the new component cost and quantity values will be Homomorphically added to the already existing component cost and quantity values in the database.

8. Experimentation and Results

While evaluating the performance of a Homomorphic algorithm various challenges are faced. Paillier algorithm uses random numbers while encryption hence becomes difficult comparing performances for encrypting different files. Various researches have been conducted pertaining to measuring the performance of Paillier Cryptosystem and comparing with other available cryptosystems, but there have been no empirical results regarding that whatsoever [4]. One of such properties on the basis of which we ensured proper performance of Paillier is Avalanche effect.

Avalanche effect [9] is a property exhibited by cryptographic algorithms that measure the dispersal of output bits in the cipher text, by changing a few bits of the input plain text. It gives a good idea about the strength of an algorithm. A secure cryptographic algorithm is the one in which at least 50% of the bits change with change in input i.e. there is a minimum of 50% probability of bits being changed in the cipher text.

The following experiment was conducted by fixing an input plain text with 'n' bits, encrypting it and observing the cipher text. Next the input plain text is modified by changing a few bits. This modified plain text is then encrypted and its cipher text is compared with the original cipher text [10]. The avalanche score is calculated using

Avalanche score = (Number of flipped bits in cipher text / Total number of bits)

Table 1: Avalanche Score for Various Set-ups for Paillier Cryptosystem

	Number of bits changed in plain text			
Sr. no.	1	2	3	4
1	0.478	0.533	0.518	0.495
2	0.500	0.478	0.539	0.484
3	0.479	0.527	0.524	0.502
4	0.524	0.486	0.523	0.498
5	0.513	0.493	0.498	0.513
Average	0.498	0.503	0.52	0.489

ElGamal cryptosystem is more suited for multiplicatively Homomorphic applications. Even though by making changes to parameters, ElGamal provides Additive Homomorphic functionality, it doesn't work for large input values. Its counterpart Paillier cryptosystem is best suited for additive Homomorphic applications. The scope of the project is to make the process of manufacturing a medicine abstract and hence requires additive homomorphism. For our project we prefer Paillier over ElGamal because of its convenience in providing additive homomorphism [11].

ISSN: 2233-7857 IJFGCN

9. Conclusion

We were able to implement Homomorphic operations as intended in a working prototype. As discussed above, we designed an application that allows employees of a particular pharmaceutical company to update sensitive data on the server cloud/database without decrypting any part of the encrypted data that is stored on the cloud/database. Since keys play an important role in carrying out these Homomorphic operations it was important to secure these keys which were unique to all the users accessing the application. For that, we designed a secure key generation and storage functionality which stored the keys on the server in its encrypted form and were only accessible based on login credentials which in itself were unique, hence adding an additional layer of security to the application.

10. Future Scope

Through our research paper we try to propose a possible application of Homomorphic Encryption in an environment pertaining to Pharmaceutical Companies. Though our proposed idea does not purely incorporate Homomorphic properties in all its functions for example we made use of the AES algorithm to encrypt the textual content and not Paillier algorithm, because the encryption function of the Paillier algorithm makes use of a random prime integer. As a result of which a certain text gets encrypted to different values every time a new entry or updation of data (i.e. Component name and component quantity) is carried out. Also Homomorphic algorithms provide no functionality regarding appending characters to textual data in its encrypted form. Further research could be carried out on how to solve these issues. Considering all the issues stated above we still hope our research and the proposed methodology may form the initial basis for future research pertaining to this domain.

References

[1] Roberts, Shawn Josette. "The necessity of information security in the vulnerable pharmaceutical industry." Journal of Information Security 5.04 (2014): 147

[2] Homomorphic Encryption: An Overview -https://www.sciencedirect.com/topics/computer-science/homomorphic-encryption.

[3] Nassar, Mohamed, Abdelkarim Erradi, and Qutaibah M. Malluhi. "Paillier's encryption: Implementation and cloud applications." 2015 International Conference on Applied Research in Computer Science and Engineering (ICAR). IEEE, 2015..

[4] Islam, Naveed, William Puech, and Robert Brouzet. "How to secretly share the treasure map of the captain?." Multimedia on Mobile Devices 2010. Vol. 7542. International Society for Optics and Photonics, 2010.

[5] V. Sidorov and W. K. Ng, "Towards Performance Evaluation of Oblivious Data Processing Emulated with Partially Homomorphic Encryption Schemes," 2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), New York, NY, 2016, pp. 113-115.

[6] Das, Debasis. "Secure cloud computing algorithm using homomorphic encryption and multi-party computation." 2018 International Conference on Information Networking (ICOIN). IEEE, 2018.

[7] Shao, Fei, Zinan Chang, and Yi Zhang. "AES encryption algorithm based on the high performance computing of GPU." 2010 Second International Conference on Communication Software and Networks. IEEE, 2010.

ISSN: 2233-7857 IJFGCN

[8] Where to store a server side encryption key?, https://security.stackexchange.com/questions/12332/where-to-store-a-server-side-encryption-key

[9] Liu, Jian-dong, Shu-hong Wang, and You-ming Yu. "TDHA-A One-Way Hash Algorithm Based on Extended Integer Tent Maps with Dynamic Properties." 2008 International Symposium on Electronic Commerce and Security. IEEE, 2008.

[10] What is the avalanche effect in cryptography? How can we measure it ?, https://www.researchgate.net/post/What_is_the_avalanche_effect_in_cryptography_How_can_we_measure_it

[11] Additive ElGamal encryption algorithm, https://crypto.stackexchange.com/questions/9000/additive-elgamal-encryption-algorithm

ISSN: 2233-7857 IJFGCN