Homomorphic Encryption over Databases

Jayakrishnan Ashok, K. N. Dheeraj, Chaitanya Subhedar, Rajeev Tiwari

Abstract*: *Human's need for storing data efficiently is perpetual. when relying on third-party services for data storage, whether storing personal or critical data, even though claimed to be completely secure, cannot be trustworthy. During security breaches or situation of personal infringement of the service provider, the need for a mechanism which can be used to protect data from any kind of unauthorised disclosures. This paper aims to showcase the capability of an encryption method that is secure and powerful in providing security to users by encrypting their data by using Homomorphism. This research work includes protecting information through encrypting data, and decryption of this data should only occur if the authorized personal requests for such needs. Else, other entities in the system will be interacting with the data in a special format which is established by the algorithm, and able to manipulate the data in the database without decrypting it at any point of time. We present a system which uses homomorphic encryption on relational database that helps to ensure the security components like integrity, confidentiality and availability of the data. The architecture introduced in this paper, is designed to fire SQL queries over encrypted data, encrypted using Paillier encryption-scheme, and perform operation directly over the encrypted text as it would perform directly on plain text.

***Index Terms*: *Homomorphism, Encryption, Homomorphic Encryption, Additive and Multiplicative Homomorphism*.**

I. INTRODUCTION

Encryption is the technique by which we are able to change our critical information into a format which is not understood to computer or human beings, especially to prevent from unwanted leaks. An encryption methodology is, plain text is modified in some means using an algorithm which will change its form, generating cipher text that can only be read after decryption. The main purpose for encrypting electronic data stored in workstation, devices or servers is to ensure security, protection of data, and to secure other digital assets this can be termed as end to end encryption, which is useful for safety of data. The Shannon model of cryptography, is one of the contemporary models used in cryptology [1-3]. It ensures security of data. As all data is encrypted into the system and is never exposed to the outside world. Data can be decrypted and then can be used, only by genuine users. It is necessary to encrypt data, so that any cyber-attack doesn't harm the integrity of data. There are some applications where encryption and decryption of data are considered an overhead, because availability of data is very crucial. The time lost in encryption and decryption is not affordable in such cases. The concept of computation on encrypted data without decryption was first introduced by Rivest [4]. In algebra, Homomorphism is a structure-preserving methodology mapping system between 2 pure mathematical algebraic structures of an equivalent or same kind (such as 2 groups, two rings, or 2 vector spaces). Homomorphism has a wide application prospect in multi-secret calculation [5], secret route, secret election [6][7], dynamic programming, mobile code security [8] and other fields which are yet to be explored. Homomorphic Encryption is an Encryption methodology through which one is able to perform mathematical operations over the encrypted data without bothering about any loss of the important information because of the computations done on data. Homomorphic Encryption is a property that an Encryption method shows when any computation on the data is done, the algorithm sets it into other values, meaning it encrypts it while preserving the end result. In simpler sense, it means when 23 is stored in server as 781 and another value 34 is stored as 456 is added or multiplied then answer would be 57 is stored as 908 and all these computations are done over encrypted data and changes are stored, when that computed value is decrypted, would result the value that the user would expect. Below is a figure, Figure 1, which explains the working of homomorphic methodology.

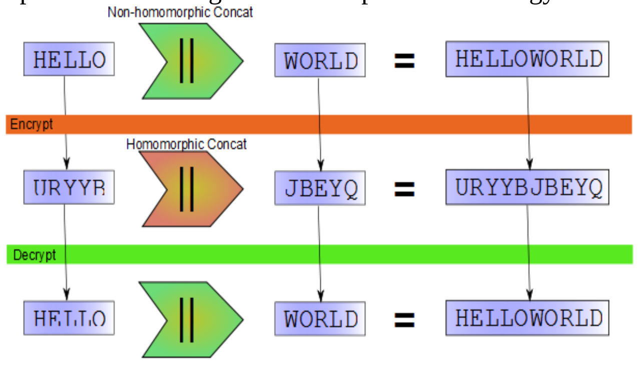


Figure 1. Describing the flow of Homomorphic Encryption

Some Applications of such crucial operations lies in financial sector where the data is the at-most critical aspect of the entire process and if exposed can lead to some serious exposure of confidentiality. But when dealing with finances we need certain experts to do some operations on the data. Putting data on cloud servers, may lead a reluctance to owner that any third eye may be watching data. So, to motivate migration towards cloud and winning the security aspect, data must be encrypted at servers and still should be open for any operations. In such situations Homomorphic encryption plays a vital role. In this research, implementation of Homomorphic encryption system using Paillier and AES Cryptosystems are shown as they show the compatibility with

Revised Manuscript Received on June 07, 2019.

Jayakrishnan Ashok, RPA Developer, Huron Consulting Group, Bangalore.
K. N. Dheeraj, Asscoiate Software Engineer, Leanapps, Pune
Chaitanya Subhedar, MS, Scholar, Purdue University, US
Rajeev Tiwari, Associate Professor, School of Computer Science, UPES, Dehradun, India.

Homomorphic Property of Additive and Multiplicative systems. Paillier Cryptosystem was developed in late 1990s, which deals with a probabilistic calculation that utilizes the combination of public and private key on text to carry out encryption and decryption processes respectively. The calculation is performed with a secret key is that is generated, by utilizing two prime numbers that used alongside while encryption and decryption process that is created using some mathematical considerations. AES is a component of Rijndael cipher developed by two cryptographers, Vincent Rijmen and Joan Daemen. Rijndael that uses numerous key and block sizes that change on each round of encryption. The remainder of this paper is organised as follows: In Section II, the notable works in the domain has been discussed. In Section III, we discuss the problem that has to be solved. In Section IV, we present our work, section V describes the experimental observations with tables and pictorial representations, Section VI showcases the Results and the paper concludes with section VII and future work is discussed in section VIII.

II. Literature Review

Data security is a methodology which allows to secure digital information by allowing secure and authorized access to the content which can be present in Database, Website or Personal computers[9]. Cryptographic systems can be put to use in any domain, including networks, databases, infrastructure. In this paper, we have discussed the most notable works done in the domain. Joshi & Karkade[10] discussed the importance of Network security in Information security. They focused on how it is implemented in infrastructure, the policies adopted by a system administrator to prevent unauthorized access. Rao et al., surveyed techniques and methodologies about data security and data privacy to be established in cloud environments, for data storage and transfer in cloud in order to establish trust between cloud provider and cloud consumer [11]. Kester et al., researched about the behaviour of digital images after being encrypted in cloud [12]. They concluded that in considering the first 1000-pixel values between image and ciphered image, there was no pixel expansion observed. The entropy value and arithmetic mean value being 6.9882 and 87.4561 respectively. Hence, they found the quality remains the same. Gentry introduced fully homomorphic encryption scheme [13], which allows anyone to work on encrypted data without decrypting at any point. A three-step process is done by the authors, beginning with an introduction of encryption scheme which evaluates its own decryption circuit and calls its bootstrappable. Mid-way through the paper, a public key cryptography using ideal lattices which is almost bootstrappable. Finally introducing a scheme which reduces the depth of decryption which enables a bootstrappable encryption scheme. Makkaoui et. Al. [14], worked on hybridizing the homomorphic encryption scheme to support all homomorphic properties. They inspected the possibilities through which algebraic structures could be used to hybridize the existing schemes which supports limited homomorphic properties. Cheon and Kim[15], introduced a hybrid homomorphic encryption scheme that combines Public Key encryption (PKE) and Somewhat homomorphic encryption (SHE), which reduced the storage requirements for Homomorphic applications. They concluded that their scheme is useful in cloud environments where small bandwidth, low bandwidth and efficient computations are required. Song et. Al. [16], introduced a hybrid cloud computing scheme which is homomorphic in nature which is obtained through combination of Pallier algorithm (additively homomorphic) and RSA encryption algorithm (multiplicatively homomorphic) which establishes the fact that large key size is not required to achieve homomorphic functionality. Their result set showcased encryption using 512, 1024, 2048 bit keys and text length less than 64KB, its able to generate key, encrypt, process and decrypt text. For text size 64KB or key size 4096 bits, the encryption scheme worked with elevated processing time. Their result concluded the practicality of key size lesser than 4096 bits and concluded that homomorphic methods are not to be used if key size is 4096 bits or text size greater than 64KB. Hayward and Chiang [17] explored the possibilities of parallelizing Gentry’s Fully homomorphic encryption methodology by backing the entire process with a private cloud built using Openstack. The design of algorithm followed was master-slave model with a test setting comprising of 1,2,4 & 8 nodes. Sum, Variance and Product of 20 random 8-bit integers were calculated on each setting. The stats showed that with increasing in the number of nodes, the time(ms) for Variance and Product decreased but while calculating Sum, the time taken increased after a short decline in magnitude. Thus, the research concluded that not all operations are to be parallelised. Rangasami and Vagdevi[18] introduced a new Homomorphic encryption (HE) scheme which is a mixed HE scheme in which key matrices are generated. Their base idea is to use BGV scheme as a primary source and on top to that add byte level automorphism to the data.They compared their methodology with RSA and Algebra Homomorphic Encryption Scheme based on Updated ElGamal(AHEE) where files of varied sizes where input and the size of cipher text, encryption time,& decryption time was observed. Their research was that as the size of data increases, the time for encryption and decryption increases which also affects the transmission time in applications. She and Zhu[19] proposed two methodologies by which they are making RSA algorithm adaptable to fully homomorphic schemes. Method one makes use of Pascal triangle formula to convert the algorithm to make it fully homomorphic and method two utilizes the modulo arithmetic to convert the legacy RSA algorithm derivation in a form where the resultant algorithm becomes fully homomorphic. Zhang et. Al.[20], introduced a fusion algorithm based on Elliptical cryptographic algorithm which makes the advanced meter services (AMI) safe and reliable. They concluded that their algorithm could validate the authenticity of the cipher text that it was proposed, refuse the illegal cipher text that an attacker may propose, and achieve message encryption and digital signing simultaneously.Wu and Ming[21] studied a methodology which is combination of IDEA algorithm (based on DES)[22] and RSA with which a key management and encryption/decryption engine module is studied.

Their study concluded that the algorithm worked effectively with encrypted database. He and Wang[23] studied the application of cryptography over RDBMS which was evaluated for its security and processing capability of huge data. They introduced the concept of security dictionary, which is used to store all of the information that is used to manage the objects in a database, and proposed several secure management methods such as password based encryption, public key based encryption, Encryption based on user supplied keys and Group Encryption. Singh & Kaur[24] interrogated the need and importance for database encryption and they made an in-depth review of various studies done in the cryptography field. They reviewed algorithms like DES, Fast comparison encryption (FCE), RSA, AES etc.

Munir[25] proposed a new security model for Database as a Service model which is available as a SaaS service from cloud providers. He described a security Architecture comprising of 4 phases with approaches to implement the required security levels in a cloud environment. Zhang et. Al.[26], proposed an authentication scheme with fault tolerance for Databases. The proposed scheme was having capability to check for alterations by any individual/Bot by checking the digit signature of the person who created the record. The Architecture is capable to correct a single error within each record due to a shared hash architecture, and is ideal for applications where security and Fault tolerance is required along with very less computational overhead. Floissac & L'Hyver[27] tested various attacks on the variants of AES algorithm(128,196,256). They considered various attacks such as Differential Fault Attacks (DFA) and C. H. Kim and J.- J. Quisquater's attack. They adapted two methodologies where the first one consisted in extending the original attack and the second one in reproducing the attack on an anterior round. Nadehara et. Al.[28], presented a work which covered the best methods to implement AES encryption algorithm which accelerates performance in embedded processors. The instructions implemented with the help of a 2-Kbit inverse table allowed table lookups and multiplications at Galois Field (GF (2^8)) and generated results in a four-way parallel, and also enabled a 640-Mbps encryption rate on a 1-GHz processor which proved sufficient for secure home network equipment at a minimal cost.
Here is a tabular representation of the notable works done in the field.

Table 1: Notable works

SN No:	Source	Authors	Findings
1	Network Security with Cryptography	Joshi, Mukund R., and Renuka Avinash Karkade	Focused on infrastructure , best policies and practices to be followed in a Network for best security.
2	Data Security in Cloud Computing	Rao, Ch Chakradhara, and A V Ramana	Investigated various methodologies and practices to be followed in a cloud environment for data storage and transfer.
3	Fully Homomorphic Encryption Using Ideal Lattices	Gentry, Craig	Invented fully homomorphic encrypto scheme which allows everyone to directly work on encrypted data without decrypting it at any point.
4	Can Hybrid Homomorphic Encryption Schemes Be Practical?	Makkaoui, Khalid El, Hssane, Abderrahim Beni, and Ezzati, Abdellah	Using Algebraic structures hybridize homomorphic encryption schemes to support all homomorphic properties.
5	A Hybrid Scheme of Public-Key Encryption and Somewhat Homomorphic Encryption	Cheon, Jung Hee, and Jinsu Kim	Concentrated on reducing storage requirements for homomorphically communicating applications
6	The modification of RSA algorithm to adapt fully homomorphic encryption algorithm in cloud computing	Sha, P., and Zhu, Z.	Developed methodologies in making RSA algorithm adaptable to homomorphic properties.

7	Cryptography and relational database management systems	J. He and M. Wang	Studied the application of Cryptography over RDBMS and its capabilities to process huge data.
8	Database security using encryption	Singh, P., & Kaur, K.	Compared various encryption algorithms.
9	Security model for cloud database as a service (DBaaS)	Munir, K.	Proposed a new architecture for security in Database as a Service model in Cloud
10	An Efficient Authentication Scheme with Fault Tolerance for Database Systems	Zhang, C. N., Chunren Lai, and Honglan, Zhong.	Proposed a new methodology by which authorization of individuals can happen by check digital signature of the person who created each record.

III. Problem Formulation

To access data which is stored in encrypted format, the encryption key is made available to the owner. The owner has to retrieve data and then decrypt it to further process/manipulate data. Considering the overheads of encryption and decryption of data at rest or at motion, it consumes time and speed with which applications need to work which can be very crucial. Homomorphic encryption allows user to make changes to the data without decrypting it. This removes the overheads discussed above and adds an extra layer of security to the system. Implementation of such Encryption Algorithm is complex and is considered challenging when used in software applications. This research showcases an application which uses Paillier encrypto system which can be considered as a deployable software application in corporate

IV. Proposed Work

In the field of Mathematics, the operations performed on data includes Exponents, Addition, Subtraction, Multiplication, Division considering a general basic need to get the desired output. In the scope of this research, calculation of Exponents was not considered, but focus was maintained on Additive and Multiplicative Homomorphism, as using Additive Homomorphism we can achieve Addition and Subtraction of the data and using Multiplicative we can achieve Multiplication and Division.

Due to the procedure of encrypting data, we can eliminate the overhead of decryption followed by performing the operation needed and encrypting it back again. Yet not changing the result of the operation. Thus, maintaining security and also improving the performance of the operation that would otherwise have some overheads.

This research introduces a very basic application, which demonstrates how Paillier homomorphic encryption can be used for Partial Homomorphic encryption. Paillier Encryption Scheme can be represented as:

$$C = g^{m}.r^{n} \bmod n^{2} \qquad (1)$$

Where C is the cipher text, m is the plain text, r is the private key and n is a multiple of two random primes of equal length and g is a random number. Homomorphic addition for plaintext m1 and m2 with cipher text C1 and C2 respectively would be:

$$C_1 = g^{m}{}_{1}.r^{n} \bmod n^{2}$$

$$C_2 = g^{m}{}_{2}.r^{n} \bmod n^{2}$$

Multiplying (2) and (3) we get

$$C_1.C_2 = g^{m_1 + m_2}.r^{2n} \bmod n^{2}$$

Comparing (4) with (1) gives:

$$C = C_1.C_2$$

$$m = m_1 + m_2$$

From (4) we can see that on decrypting the cipher obtained from (2) we can get the sum of the plaintexts.

To get results similar to Multiplicative homomorphism using paillier encryption scheme we have done the following operations:

$$(C_1)^{m_2} = g^{m_1 \cdot m_2}.r^{m_2 n} \bmod n^{2} \qquad (2)$$

Comparing (1) and (5)

$$C = (C_1)^{m_2}$$

$$m = m_1.m_2$$

To decrypt the cyphertext, we use two primes of same length p and q such that,

$$n = pq,$$

$$\alpha = \text{lcm}(p-1 , q-1),$$

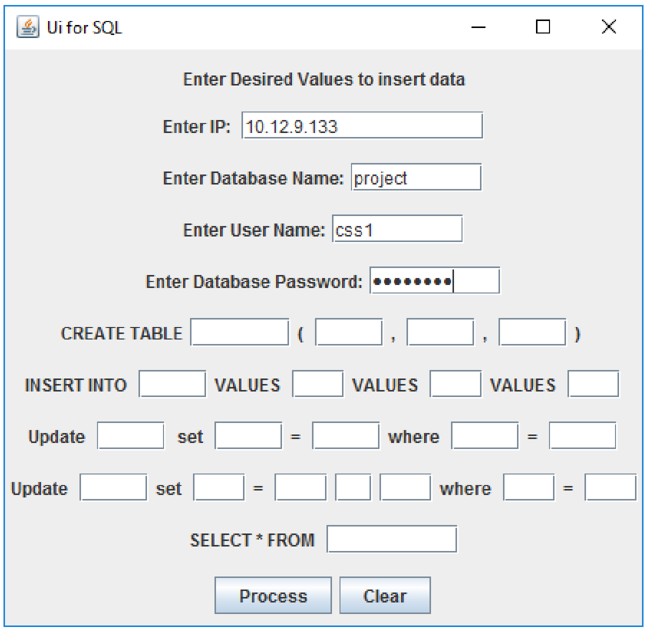
$$L(x) = (x-1)/n$$

$$\mu = (L(g^{\alpha} \bmod n^2))^{-1} \quad (3)$$

The plain text will be:

$$m = L(c^{\alpha} \bmod n^2).\mu \bmod n \quad (4)$$

The operations involving cipher text are arithmetic operations and can be performed using simple SQL queries to the database, other operations are used to generate the private key and will be performed on the user side.
As, (3) depends on the value of g and g is a random number, a plaintext can have more than one ciphertexts. To, keep the ciphertext for column names constant, AWS encryption scheme is used, which returns same ciphertext for a fixed set of plain text and key.
To visualize the working of the encryption scheme, GUI was created which lets user select different arithmetic operations, which can be performed on the database as shown in figure 2.



Enter Desired Values to insert data

Enter IP: 10.12.9.133

Enter Database Name: project

Enter User Name: css1

Enter Database Password: ••••••••

CREATE TABLE (, ,)

INSERT INTO VALUES VALUES VALUES

Update set = where =

Update set = where =

SELECT * FROM

Figure:2. UI for user input.

The GUI lets user execute SQL commands on a database which can be set up on the host machine or any remote machine which is in the same network as the host machine.

The application when executed creates and stores a private key which is used for encryption and decryption process, the plaintext entered by the user is replaced by the ciphertext and a SQL query consisting cipher text is executed

V. Experimental Observations

Table below displays two different views of the homomorphically encrypted database. Left side showcases the state of Database in the Encrypted format (seen by any third party), while right side depicts the same data in a decrypted manner (person who has key for decryption). Below is a table 2, which compares the Encrypted view to decrypted view of the data.

Working on encrypted database consumes time, which can be considered as an overhead according to the processes. To test the time taken/time consumed for each method different tests were performed. Three scenarios were considered while observing the working of the application, They, were 1. Working with normal Encrypted database. 2. Working with Database encrypted using Homomorphic method. 3. Non-encrypted database.

Considering update query, the time taken for Normal encrypted database is given below.

Encrypted View	Decrypted View
CREATE TABLE student (id , name , sap)	
XKnOVnXu0jaQmTqlMfjzbw== | N6+UdBsQSx8Cj4TyzrJqRA== | NLU7JVj1SwP+QQs6GTWWcg	id | name | sap
INSERT INTO student VALUES 1 VALUES abc VALUES)2334	
XKnOVnXu0jaQmTqlMfjzbw== | N6+UdBsQSx8Cj4TyzrJqRA== | NLU7JVj1SwP+QQs6GTWWcg 394218228264377245041586531... | 349074952628899133189032868... | 44293735297014787873895453	id | name | sap 1 | abc | 50002334
Update student set id = id + 2 where name = wxy	
XKnOVnXu0jaQmTqlMfjzbw== | N6+UdBsQSx8Cj4TyzrJqRA== | NLU7JVj1SwP+QQs6GTWWcg 394218228264377245041586531... | 349074952628899133189032868... | 44293735297014787873895453 441107628948859250648306698... | 256757303401237542587174153... | 35499778625861239932939657	id | name | sap 1 | abc | 50002334 4 | wxy | 50002375

Table 2 : Comparison between Encrypted and Decrypted view

```
Time for homomorphic encrypted update: 745458362 ns
```

The time taken for Update command with non-encrypted database is given below.

```
Time for plaintext update: 519158644 ns
```

Even though the time interval taken for the Homomorphically encrypted methodology consumes more time (~2 seconds) to Plain text, the security which it offers is un-comparable. The results showcase that when the architecture is put to work, the time taken for the entire application to process data and store it in an encrypted manner is worth the effort. The data received from the database after decryption resembles the actual output that is expected, which proves the accuracy of encryption and decryption of Paillier encrypto-system. For the same, several other tests were performed on the database to validate the accuracy and performance of the algorithm.

VI. RESULTS

In this paper, 3 tests were performed to evaluate the efficiency and robustness of our architecture build using Paillier Encrypto-system. They are Uniqueness, Homomorphic Evaluation and Decryption Accuracy. For performing these tests, values of p and q were fixed.

p= 8.73988690050758E+76

q= 6.4336310270582E+76

a. Uniqueness

This test was performed with the motive to observe the uniqueness of the cipher text that will be created for the same plain text and fixing values of p and q (prime numbers used for encryption) and by varying random number r. Below is the table representing the values after calculation.

Number	Random r	Cipher text	Conclusion
10	1.1008E+154	846510095893784 266089882691049 649270439150876	Unique
10	8.196E+153	240801595641083 597599840432248 452467630076927	Unique
10	9.9939E+153	324410409822871 919750444629752 615793122086464	Unique
15	2.9882E+153	164319771178844 475170245799148 823528512939444	Unique
15	5.0134E+153	560682007301010 764013730914799 951956160583869	Unique
15	1.9101E+153	581570189776171 575642351131695 931431765882195	Unique

Table 3 : uniqueness of Cipher text by varying r.

As observed from the cipher text, the algorithm maintains uniqueness in creating cipher text while varying values of

b. Homomorphic Evaluation

This test is performed to evaluate the decryption capabilities of the algorithm for numbers varying in its magnitude. The results have represented below table 4.

Numbers	Cipher	Decrypted Text	Conclusion
20+30	8.6699E+306	50	TRUE
35+45	1.155E+307	80	TRUE
80+90	5.1046E+306	170	TRUE
1029384757 + 1293847509776	578901891748263 769545585375232 605632461835798	1.11306E+11	FALSE
12398979872 + 1209821371	223383466782366 193016258413712 344558934566732	13608801243	TRUE

Table 4 : Decryption of varying Encrypted text

As observed, addition of very large numbers the accuracy of the algorithm is very low. This concludes the algorithm is not suitable for situation where addition are large numbers are needed.

c. Decryption Evaluation

This test is performed to understand the decryption capability of the algorithm by varying random number r. The result is represented in below table 5. As observed, the decrypted version is as expected and the values hold intact. This concludes that the algorithm is efficient is retrieving the encrypted value.

Cipher	Decrypted	Expected Number	Conclusion
846510095893784 266089882691049 649270439150876	10	10	TRUE
240801595641083 597599840432248 452467630076927	10	10	TRUE
324410409822871 919750444629752 615793122086464	10	10	TRUE
164319771178844 475170245799148 823528512939444	15	15	TRUE
560682007301010 764013730914799 951956160583869	15	15	TRUE

Table 5 : Decryption of encrypted text by varying r.

As a final test, we observed the time taken for decryption as we vary the values of random number r. The results are represented below table 6.

Run-Time for different values of r.	Time taken for any encryption (ns)
1.1008E+154	743094329
9.9939E+153	742688971
8.196E+153	737101741
2.9882E+153	733465260
5.0134E+153	736344693
1.9101E+153	726553264

Table 6 : Variation of Time taken when random number r is changed.

As observed, we concluded that as the value of r increases the time taken to decrypt the cipher text is increasing.

VII. CONCLUSION

Through this paper it has been shown how the encrypted data can be passed over to databases which is present locally or remotely. Also, the research showcased Homomorphic Property of the Encryption Algorithm by doing different operations over the encrypted data like performing arithmetic operations +, -, * etc. It provides a method that would help in making a system where the data can be stored encrypted and operations can be done on the encrypted data and the value decrypted would retain the same value as expected if the operations were performed on plain text.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

In this System the Database used is entirely encrypted that the Column names and database name is even encrypted at the time of creation using RSA algorithm. Which means that any modification done on the Database is unknown to anyone how tries to attack it as any data regarding tuple values or database is unknown. Limitations in the System are that it isn't dynamic for certain functions like Create table system doesn't support for more than 3 column values. Also, for very large values, the decryption process is not retrieving the expected output. Thus, scope for better algorithms are present which can be developed for the system to enhance its performance and memory consumption. As noticed, the time taken for decryption process to increasing with the increasing value of r. thus compromising on the magnitude of r, can expose a security threat. When during a Brute-Force attack on the system, for smaller values of r may crack. Huge Development scope is present through which the system can be made to work more dynamically in the cloud for n number of applications and is generating encrypted data along with making them secure for accepting any kind of data.

VIII. FUTURE WORK

Throughout this research, the length of the cipher text is considered too large to be used with ordinary Java data types, so the code should be modified to produce smaller cipher text length. So, the Algorithm can be further synthesized to generate lesser length of the cipher value for better memory management. This System currently, is not smart enough to relate appropriate Encryption and Decryption function as per the input data type. Once done, it can be tested and deployed over to a Cloud Environment. Generic Encryption and Decryption function can be created to accept any data value and can encrypt and decrypt it. The Encryption methodology discussed in this paper can be programmed to handle further mathematical operations like exponents, modulo etc. The Encryption method discussed in this paper only provides security from other authorized / unauthorized people not able to see the original data. But doesn't prevent any unauthorized operations thus corrupting the data. In order to prevent the encrypted data being hampered or corrupted we must have to employ mechanism like Hamming Code, Parity Bits, Checksums, Cyclic redundancy checks (CRCs), Cryptographic hash functions etc. that ensure data integrity to prevent unauthorized people from hampering the data.

ACKNOWLEDGMENT

We are really grateful to our College and Department of School of Computer Science, University of Petroleum and Energy Studies for all the support, guidance, Infrastructure that was provided to us which helped us in carrying out the project and completing this research work.

REFERENCES

1. M. Jessa: "Correlation in Pseudo chaotic Sequtmces Generated in the Set of Natural Numbers", Proceedings of the 6Ih International Specialist Workshop on Nonlinear Dynamics of Electronic Systems NDES'98, Budapest 1998, pp.169-172
2. M. Jessa: "Maximal Cycle Length of Pseudo chaotic Sequences Generated by Piecewise Linear Maps", Proceedings of the 5Oth Intenational Symposium on Circuits and Systems ISCAS'99, Orlando 1999, vol. 5,pp. 450-453.
3. D. E. Knuth: The Art of Computer Programming, Vol. 2, 2"d edn., Addison Wesley, 1981
4. R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphism. Foundations of Secure Computation, pages 168–177, 1978
5. Stepanov M., Bezzateev S., Jung T.C. Privacy Homomorphism for Delegation of the Computations[C]. 6th International Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking, NEW2AN 2006. St. Petersb :Springer-Verlag, 2006: 474-480.
6. Cramer R., Gennaro R., Schoenmakers B. A secure and optimally efficient multiauthority election scheme[C]. in Advances in Cryptology (EUROCRYPT '97). New York: Springer, 1997: 103–118.
7. M. Yokoo, K. Suzuki. Secure Multi-agent Dynamic Programming based on Homomorphic Encryption and its Application to Combinatorial Auctions[C], Proceedings of the First International joint Conference on Autonomous Agents and Multiagent systems (AAMAS), 2002:112-119.
8. Jansen W., Karygiannis T. Mobile agent security[J]. NIST Special Publication 800-19, NIST, 2000: 1-38.
9. Kumari, Sarita. "A Research Paper on Cryptography Encryption and Compression Techniques." International Journal of Engineering and Computer Science, 4 Apr. 2017, pp. 20915–20919., doi:10.18535/ijecs/v6i4.20
10. Joshi, Mukund R., and Renuka Avinash Karkade. "Network Security with Cryptography." International Journal of Computer Science and Mobile Computing, Vol. 4, no. 1, Jan. 2015, pp. 201–204.
11. Rao, Ch Chakradhara, and A V Ramana. "DATA SECURITY IN CLOUD COMPUTING." International Journal of Current Trends in Engineering & Research (IJCTER), vol. 2, no. 4, Apr. 2016, pp. 84–92. ISSN - 2455–1392 Retrieved from https://www.ijcter.com/
12. Kester, Quist-Aphetsi, et al. "A Novel Cryptographic Encryption Technique for Securing Digital Images in the Cloud Using AES and RGB Pixel Displacement." 2013 European Modelling Symposium, 2013, doi:10.1109/ems.2013.51.
13. Gentry, Craig. "Fully Homomorphic Encryption Using Ideal Lattices." Proceedings of the 41st Annual ACM Symposium on Symposium on Theory of Computing - STOC '09, 2009, doi:10.1145/1536414.1536440.
14. Makkaoui, Khalid El, et al. "Can Hybrid Homomorphic Encryption Schemes Be Practical?" 2016 5th International Conference on Multimedia Computing and Systems (ICMCS), 2016, doi:10.1109/icmcs.2016.7905580.
15. Cheon, Jung Hee, and Jinsu Kim. "A Hybrid Scheme of Public-Key Encryption and Somewhat Homomorphic Encryption." IEEE Transactions on Information Forensics and Security, vol. 10, no. 5, 2015, pp. 1052–1063., doi:10.1109/tifs.2015.2398359.
16. Song, Xidan, and Yulin Wang. "Homomorphic Cloud Computing Scheme Based on Hybrid Homomorphic Encryption." 2017 3rd IEEE International Conference on Computer and Communications (ICCC), 2017, doi:10.1109/compcomm.2017.8322975.
17. Hayward, Ryan, and Chia Chu Chiang. "Parallelizing Fully Homomorphic Encryption." 2014 International Symposium on Computer, Consumer and Control, 2014, doi:10.1109/is3c.2014.192.
18. Rangasami, K., & Vagdevi, S. (2017). Comparative study of homomorphic encryption methods for secured data operations in cloud computing. 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT). doi:10.1109/iceeccot.2017.8284566.
19. Sha, P., & Zhu, Z. (2016). The modification of RSA algorithm to adapt fully homomorphic encryption algorithm in cloud computing. 2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS). doi:10.1109/ccis.2016.7790289
20. Zhang, X., Tong, W., Jin, X., & Dai, S. (2014). Research on Fusion Algorithm of Elliptic Curve Cryptography in Advanced Metering Infrastructure Communication. 2014 Seventh International Symposium on Computational Intelligence and Design. doi:10.1109/iscid.2014.96

21. Wu, X., & Ming, X. (2010). Research of the Database Encryption Technique Based on Hybrid Cryptography. 2010 International Symposium on Computational Intelligence and Design. doi:10.1109/iscid.2010.105.
22. R. Chen and D. Yuan, "Key technique researching of database encryption engine", Computer Engineering and Design, 2007(14): 182-184.
23. J. He and M. Wang, "Cryptography and relational database management systems," in Database Engineering and Applications, 2001 International Symposium on., 2001, pp. 273-284; doi:10.1109/ideas.2001.938095
24. Singh, P., & Kaur, K. (2015). Database security using encryption. 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE). doi:10.1109/ablaze.2015.7155019
25. Munir, K. (2015). Security model for cloud database as a service (DBaaS). 2015 International Conference on Cloud Technologies and Applications (CloudTech). doi:10.1109/cloudtech.2015.7336974
26. Zhang, C. N., Chunren Lai, & Honglan Zhong. (n.d.). An Efficient Authentication Scheme with Fault Tolerance for Database Systems. Third International Conference on Information Technology and Applications (ICITA'05). doi:10.1109/icita.2005.62
27. Floissac, N., & L'Hyver, Y. (2011). From AES-128 to AES-192 and AES-256, How to Adapt Differential Fault Analysis Attacks on Key Expansion. 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography. doi:10.1109/fdtc.2011.15
28. Nadehara, K., Ikekawa, M., & Kuroda, I. (n.d.). Extended instructions for the aes cryptography and their efficient implementation. IEEE Workshop onSignal Processing Systems, 2004. SIPS 2004. doi:10.1109/sips.2004.1363041

AUTHORS PROFILE

Jayakrishnan Ashok, working as RPA Developer, Huron Consulting Group, with standard Machine learning algorithms and Natural language processing techniques. Currently into exploring RPA. Interested in ML,AI, Cryptography, intelligent systems and secured platforms.

K.N. Dheeraj, Associate Software Engineer in Lean Apps, Pune. He is working in Data Science and Full stack Development. Exploring Securing systems and cryptography.

Chaitanya Subhedar, Research Scholar, MS, Purdue University, US. He has his interest in computer graphics, Algorithms, Cryptography and competitive programming. Exploring intelligent system architecture and secured communication.

Rajeev Tiwari, He has done doctorate in CSE. He has his interest and work in field of cloud computing, security of cloud platforms. He has more than 30 publication in his name.