Improvised Version: Fully Homomorphic Encryption

Avinash Navlani
Assistant Professor,
School of Future Studies and Planning,
DAVV, Indore India

Pallavi
Masters in Technology,
School of Future Studies and Planning,
DAVV, Indore India

ABSTRACT
Homomorphic encryption schemes are malleable by design. In the field of homomorphic encryption schemes have made it possible to implement a variety of schemes using different techniques and programming languages. In this paper, we choose the model to increase the efficiency and security. Basically this technique is a method of encryption that combines two or more encryption technique and usually includes a combination of symmetric and asymmetric (public-key) encryption to take benefit of the strengths of each type of encryption. Fully homomorphic encryption schemes, we observe that the main bottleneck for this scheme is slow running speed and large cipher text.

Keywords
Cryptography, Homomorphic Encryption, Symmetric and Asymmetric Encryption.

1. INTRODUCTION
Information Security, is defined as safe-guarding an organisation data from unauthorised access as modification to ensure its availability, confidentiality and integrity. "The protection of information whether in storage processing as transit and against the denial of service to unauthorised users including those measures necessary to detect, document and counter such threads."

U.S.National Information System Security

The development of homomorphic encryption provides a better approach to build secure function evaluation protocols. This encryption scheme allows computation directly on encrypted data. However, the community has grown to trust the security of these schemes and, recently the work of Gentry and others demonstrate that, when carefully employed such homomorphic properties can be quite valuable. A number of recent specific applications such as data aggregation in distributed network [3], electronic voting [4], biometrics [5] and privacy preserving data mining have led to reignited interest in homomorphic schemes.One of the most significant developments in cryptography in the last few years has been the introduction of the first fully homomorphic encryption scheme by Gentry [7].

2. HOMOMORPHIC ENCRYPTION
In recent computing scenarios clients are trusted (but weak), while computationally strong servers are untrusted as we do not exhibit full control over them. How to outsource (delegate) the computation? What about privacy of the outsourced computation? What about privacy of the outsourced computation? For example, how to outsource computing on medical data, which must be kept confidential of all times? Standard solution would be to encrypt the data: this perfectly solves any privacy issues. However, requirements for standard encryption schemes (in particular, non-malleability) also do not let us achieve the wanted functionality: we cannot perform any computations on the encrypted data. Thus, problems with traditional encryption are as data needs to be decrypted whenever we have to perform any computation on the data. What if Party A does not trusts Party B, with its confidential data.

A solution to this problem is homomorphic encryption, Party B does not requires the secret key for answering this query from A. Homomorphic is a Greek word for "same structure". It permits computing on encrypted data. That is, the client can encrypt his data x and send the encryption $Enc(x)$ to the server. The server can then take the cipher text $Enc(x)$ and evaluate a function f on the underlying x obtaining the encrypted result $Enc(f(x))$. The client can decrypt this result achieving the wanted functionality, but the server learns nothing about the data that he computed on [8].

According to us, Homomorphic encryption schemes are malleable by design. In other words, in some circumstances it may be viewed as a feature that anyone can transform an encryption of m into a valid encryption of $f(m)$ without necessarily learning m. Such schemes are known as homomorphic encryption scheme.

3. FULLY HOMOMORPHIC ENCRYPTION
Fully homomorphic encryption is an encryption scheme that allows specific functions to be performed on the cipher texts and obtaining an encrypted result, which is when decrypted to obtain a result as if the operation was originally performed on the plain texts. The term homomorphic encryption has come from the fact that even after performing operation on the encrypted data, the scheme is able to retain the original property of the original data. Literally, Homomorphic means the same property. Thus homomorphic encryption is a scheme that keeps the original properties of the plain text, even though it allows operation on data.

The concept of Fully Homomorphic Encryption (FHE) was introduced by Rivest, Adleman, and Dertouzous soon after the invention of RSA algorithm in 1978, originally as a method to allow expensive computation to be performed by any untrusted third party. Rivest et.al [9] named it privacy encryption and showed RSA algorithm [10] to be multiplicative homomorphic encryption scheme-i.e., given a RSA public key pk=(N,e) and cipher texts $C_i=M_i^e mod\ N$, one can efficiently compute $\Pi_i C_i=(\Pi_i M_i)^e \bmod N$, which is eventually a cipher text that encrypts the product of the original plaintexts. With this invent, they asked a basic question: What can one do with a scheme that is fully homomorphic. The answer to this question is that one can compute arbitrary number of computations on the encrypted data without the need to decrypt it anywhere. This is a desirable feature in modern computing system where security and privacy of various user data is an area of concern and with the inception of cloud computing, this technique has gain lots of importance in terms of research and development. The application and prospects of homomorphic encryption is enormous as it can be used to develop secure voting systems,

collision resistant hash function, and searchable encrypted database and enable widespread use of cloud computing by ensuring the confidentially of the data stored and processed on the server.

4. IMPROVING FULLY HOMOMORPHIC ENCRYPTION SCHEMES

Reason behind choosing this model is the security and efficiency. Most interesting thing in this technique is the combination of two different techniques. Basically this technique is a method of encryption that combines two or more encryption technique and usually includes a combination of symmetric and asymmetric (public-key) encryption to take benefit of the strengths of each type of encryption. Symmetric encryption has the performance advantage and therefore is the common solution for encrypting and decrypting performance-sensitive data, such as an online data stream. However, symmetric encryption has a downside the cryptographic key needs to be known to both the sender and receiver of encrypted data, and the exchanging of the key over an insecure channel may cause security risks. On the other hand, asymmetric or public-key encryption provides better security in that the cryptographic key required for decrypting data does not have to be shared with other parties. Here we show a small description of the proposed architecture shown in figure, as proposed by Authors. In the proposed architecture user input secret integers and secret key. This secret integer is encrypted by proposed encryption algorithm with the help of secret key to produced cipher text. At the same time secret key is also encrypted by the public RSA key and combine cipher integers and cipher key and send to other end. At other end receiver receive cipher text and cipher key. Initially receiver first decrypt cipher key by RSA private key and then decrypt cipher text by proposed algorithm with the help of secret key to get original secret integer.

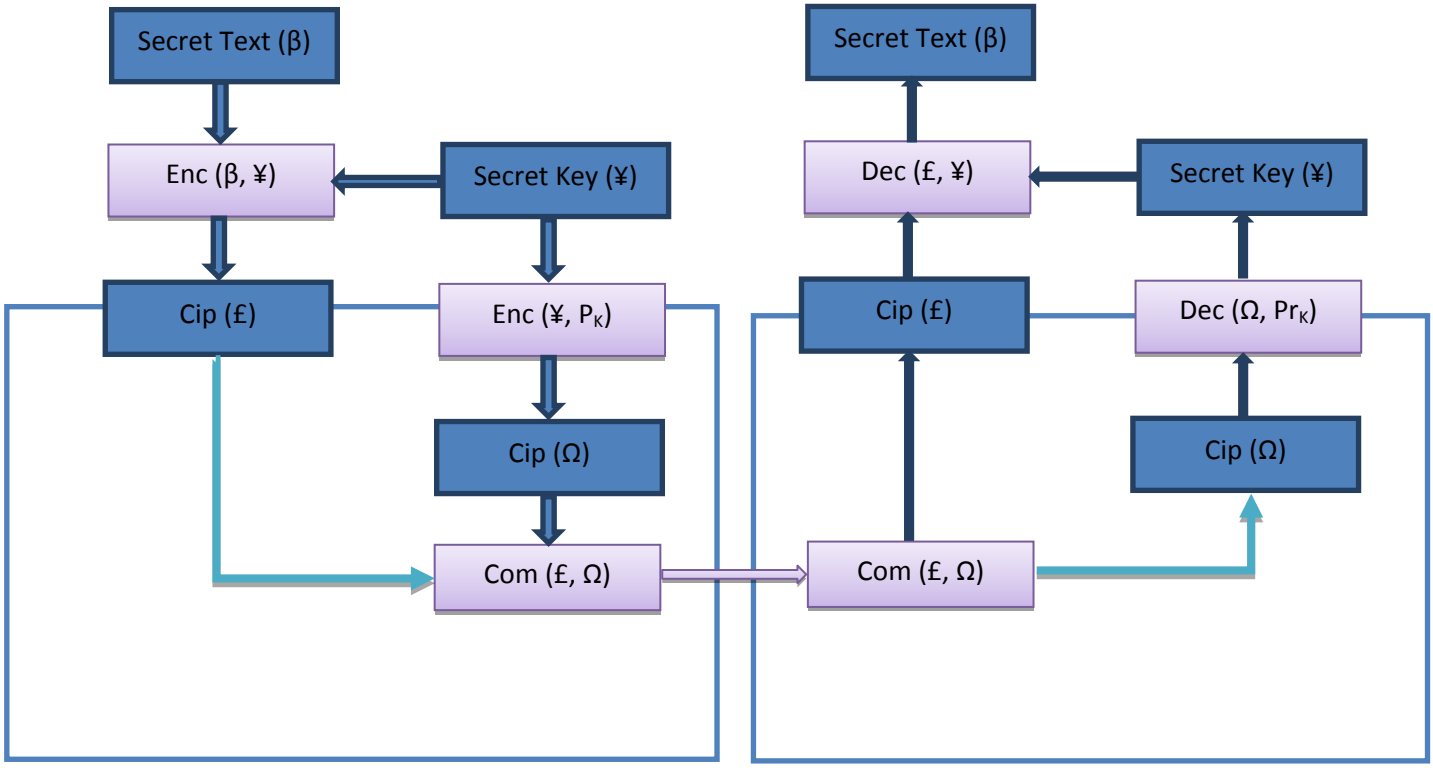


Fig: Architecture of Proposed Encryption and Decryption

These are the following parameters:

- β is representing for arbitrary bit length of secret input
- ¥ is representing for arbitrary bit length of secret key
- μ is representing for summation of all the ASCII value of characters in arbitrary length key
- α is representing for generated Pseudo Random Number
- π is representing for the containing bits in matrix
- £ is representing for cipher text of arbitrary bit length of secret input
- Ω representing for cipher text of arbitrary bit length of secret key

Proposed work takes two things from user. First is input and another is secret key of arbitrary length. With the help of secret key, user is producing a pseudo random number by using following steps.

With this help of this pseudo number, proposed work will generate it's new secret key.

1. Input secret key of (¥) arbitrary length key.

2. Generate a Pseudo Random Number (α)

   a) Add all the ASCII value of characters in arbitrary length key

      $\mu = \sum \text{ASCII}(¥)$

   b) Calculate modulo of 4 μ

      $\alpha = \mu \pmod 4$

3. If the length of β is not multiple of 16 then making it multiple of 16 by padding '\0' at the end

4. Key Generation (¥)
   a) Input secret key of (¥) arbitrary length key.
   b) If length (¥) < 16, "XOR" all the characters with the characters at position α ahead to it.
   c) Append the results with last key
   d) Go to step b).
   e) At last, select first 16 characters of the resultant key.

Fig: Re-generating Secret Key using Pseudo Number

With the help of this secret key, proposed work will encrypt the input contents. This is shown in figure 3.

5. Repeat the step for all characters of (β)
   a) Input secret text (β) of arbitrary length.
   b) Take 16 characters of β
   c) Arrange β and ¥ in matrix of 4X4 in row major order.
   d) π = β XOR ¥
   e) Column shifting of π (select α and swap it with next to next column)
   f) Transpose of matrix π
   g) Row Shifting of matrix π (Select α and swap it with next to next row)
   h) Results is a cipher text (£) of first 16 characters of β.
   i) Repeat a to g till β != NULL.

5. RESULTS

For the tests, a laptop with a Core i5 m520@2.4GHz CPU and 2GB of Ram memory was used. The proposed system has run hundred times approximately. In each time, same integer value are respectively encrypted and decrypted by **"Proposed system"** by copying them. Size of the selected key was very in each time. Finally, the outputs of the proposed system are execution time which is noted in numeric form and showing in table 1 to 4. Table 1 shows the concrete parameters used by authors in his work. Those parameters were chosen to mitigate certain kinds of attacks against the cipher text.

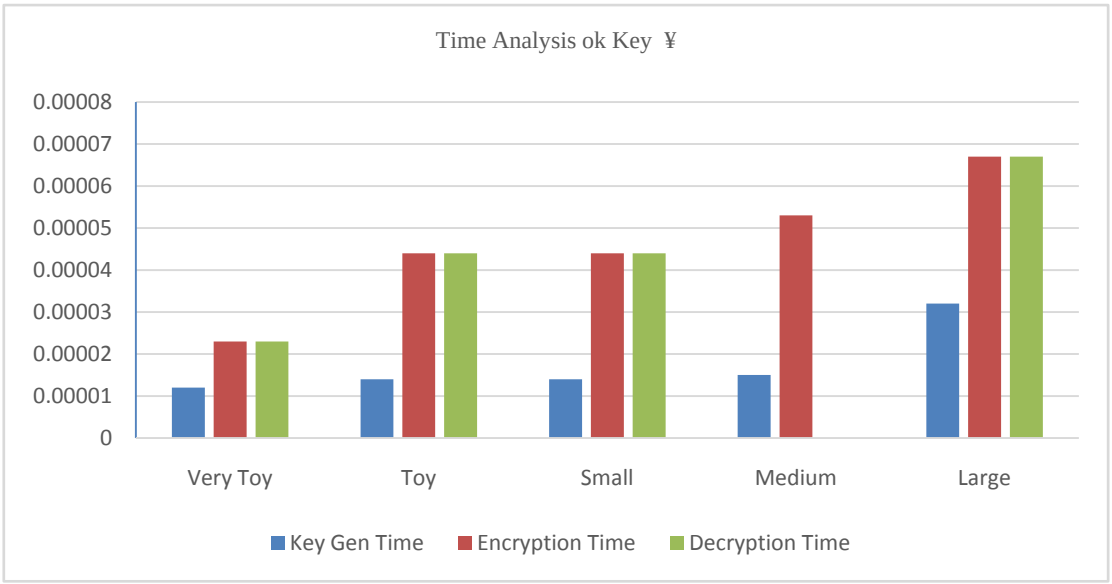
Table 1: Parameter Value

Key Character	¥	β	µ	A	£	Ω
Very Toy	48	16	8	2	16	96
Toy	64	32	9	2	32	128
Small	80	64	10	2	64	160
Medium	96	112	10	2	112	192
Large	128	144	10	2	144	256

Table 2 shows the obtained times for the key generation and encryption and decryption time of key by proposed algorithm described by authors. **"The Proposed Crypto System"** has been implemented on a number of integers varying sizes of a wide range. Table 3 shows, generating pseudo random number time by proposed algorithm. Table 4 is showing encryption and decryption time of integers by proposed algorithm.

Table 2: Execution Times in Nano Second

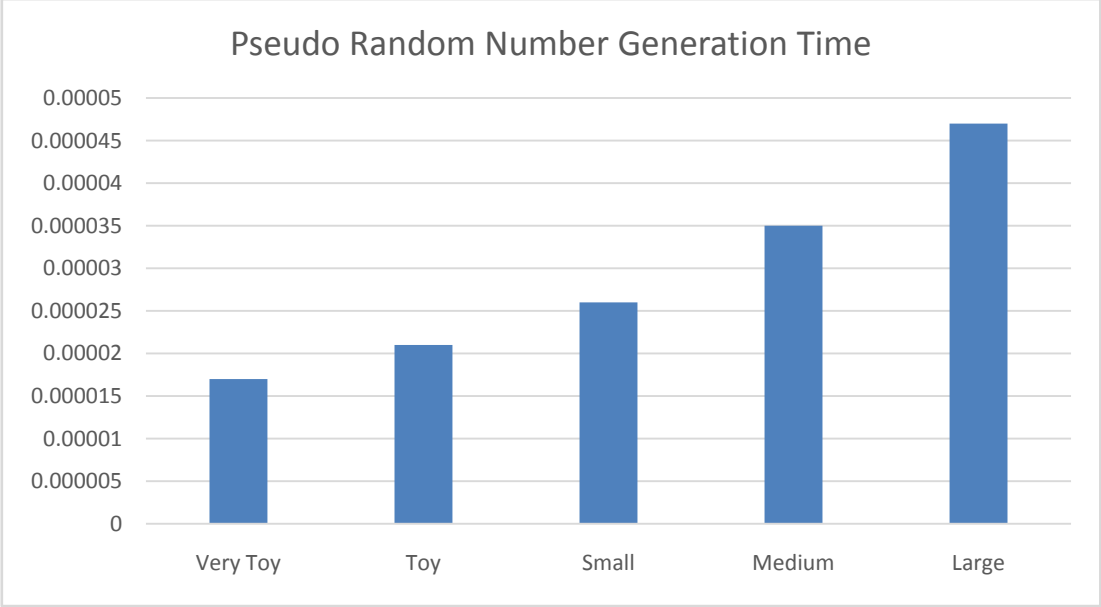
Key Character	¥	Key Gen Time	Encryption Time	Decryption Time
Very Toy	48	0.000012ns	0.000023ns	0.000023ns
Toy	64	0.000014ns	0.000044ns	0.000044ns
Small	80	0.000014ns	0.000044ns	0.000044ns
Medium	96	0.000015ns	0.000053ns	0.000053ns
Large	128	0.000032ns	0.000067ns	0.000067ns



Graph 1: Time Analysis of Key in Nano Second

Table 3: Time Analysis of Pseudo Random Number Generation

Key Character	¥	Pseudo Random Number Generation Time
Very Toy	48	0.000017ns
Toy	64	0.000021ns
Small	80	0.000026ns
Medium	96	0.000035ns
Large	128	0.000047ns



Graph 2: Time Analysis of Pseudo Random Number Generation

Table 4: Time Analysis of Encryption and Decryption of Integers

Key Character	¥	β	Encryption Time(ms)	Decryption Time(ms)
Very Toy	48	48	0.000031	0.000031
Toy	64	64	0.000037	0.000037
Small	80	80	0.000043	0.000043
Medium	96	96	0.000045	0.000045
Large	128	128	0.000053	0.000053

6. CONCLUSION

In this paper, a fully homomorphic encryption scheme is described that uses only simple integer arithmetic. The primary open problem is to improve the efficiency of the scheme. The choice of dot net as the programming language in this project facilitated the coding in some aspects, for instance, operations with lists are simple to implement using dot net native resources. The fully homomorphic encryption supports arbitrary number of addition and multiplication on the cipher texts, in this paper the number of addition and multiplication are decreased thus it reduces the time.

7. REFERENCES

[1] E.Mykletun, J.Girao and D.Westhoff. Public key based Cryptoschemes for Data Concealment in Wireless Sensor Networks. In IEEE, Int. Conference on Communication ICC, Istanbul, Turkey June 2006.

[2] Aggelos Kiayias, Moti Yung, Tree-Homomorphic Encryption and Scalable Hierarchical Secret Ballot Elections. Financial Cryptography 2010: pp, 257-271.

[3] An application of the Goldwasser-Micali Cryptosystem to Biometric Authentication, Information Security and Privacy, LNCS 4586, pp. 96-106, 2007.

[4] C.Gentry, Fully homomorphic encryption using ideal lattices. Symposium on Theory of Computing (STOC), 2009, pp. 169-178.

[5] Vinod Vaikuntanathan, Lecture on Computing on Encrypted Data on Sep 09'13.

[6] Paillier Pascal. "Public-key Cryptosystems Based on Composite Degree of Residuosity Classes". EUROCRYPT Springer. pp. 223-238, 1999, doi: 10,1007/3-540-48910-X_16.

[7] T. ElGamal A public key cryptosystem and a signature scheme based on discrete algorithms. In Advances in Cryptology- CRYPTO '84, Volume 196 of Lecture Notes in Computer Science, pages 10-18. Springer – Valag, 1985.

[8] S.Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In STOC, pages 365-377. ACM, 1982.

[9] R.Rivest, A.Shamir and L.Aldeman. A method for obtaining Digital Signatures and Public Key Crypto systems, Communication of the ACM 21 (2) : 120-126, 1978.

[10] R.Rivest, L.Adleman and M.Dertouzos. On data banks and privacy homomorphism. In Foundations of Secure Computation, p.p.- 169-177. Academic Press, 1978.