VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

J- COMPONENT

(REVIEW 1)

Information Security Analysis and Audit

TITLE-DATA SECURE SMART HOME AUTOMATION SYSTEM

PRESENTED BY - Gunjan Kumar

Reg no. - 18BIT0070

SLOT – G1

TO :

SUMAIYA THASEEN I

TITLE- DATA SECURE SMART HOME AUTOMATION SYSTEM

OBJECTIVES

1. Create Language brain using deep learning
2. Create Task brain using deep learning
3. Frame a secure encryption pathway towards main server using homomorphic encryption.
4. Handle bot attacks.

MY OBJECTIVE

Frame a secure encryption pathway towards main server using homomorphic encryption.

INTRODUCTION

Homomorphic encryption is a form of encryption allowing one to perform calculations on encrypted data without decrypting it first. The result of the computation is in an encrypted form, when decrypted the output is the same as if the operations had been performed on the unencrypted data.

Homomorphic encryption can be used for privacy-preserving outsourced storage and computation. This allows data to be encrypted and out-sourced to commercial cloud environments for processing, all while encrypted. In highly regulated industries, such as health care, homomorphic encryption can be used to enable new services by removing privacy barriers inhibiting data sharing. For example, predictive analytics in health care can be hard to apply due to medical data privacy concerns, but if the predictive analytics service provider can operate on encrypted data instead, these privacy concerns are diminished. There are various encryption techniques used as homomorphic encryption.

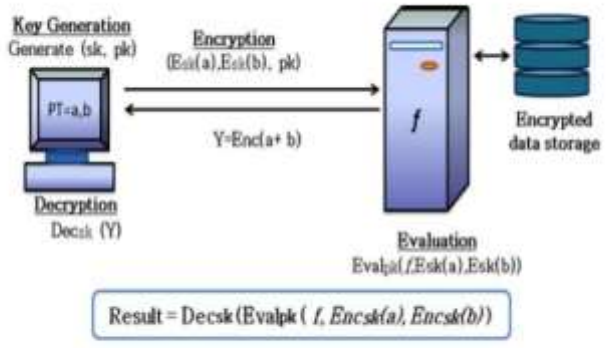

Figure 1: Homomorphic Encryption functions

LITERATURE SURVEY

1. OPTIMIZED IMPLEMENTATION OF ELLIPTIC CURVE BASED ADDITIVE HOMOMORPHIC ENCRYPTION FOR WIRELESS SENSOR NETWORKS

Author: Ugus, Osman, et al.

Year and Journal: 2009, IEEE

TECHNIQUE IMPLEMENTED AND MOTIVATION

Ugus, Osman, et al. [1] talks about several candidates for an asymmetric additive homomorphic encryption schemes. In their analysis, the authors observed that the **Elliptic Curve-ElGamal (EC-ElGamal)** encryption scheme represents the most promising one due to its superior performance and small ciphertext size.

The original ElGamal encryption scheme, is not additive homomorphic. However, the elliptic curve group is an additive group, which can be used to get an **additive homomorphic scheme**. Algorithm 1 and Algorithm 2 show the methods for EC-ElGamal encryption and decryption, respectively. Therein, E is an elliptic curve over the finite field GF(p). The order of the curve E is denoted n = #E and G is the generator point of the curve E. The secret key is defined as integer number x ∈ GF(p), while the public key is determined as Y = xG.

The function map() is a deterministic mapping function used to map values mi ∈ GF(p) to curve points Mi ∈ E such that

$$map(m_1 + m_2 + \ldots) = \underbrace{map(m_1)}_{M_1} + \underbrace{map(m_2)}_{M_2} + \ldots$$

holds. This is necessary, because the addition over an elliptic curve is only possible with points on that curve, thus, integers have to be mapped to corresponding points. For this purpose, each integer m is mapped to a curve point M, where M is the m-multiple of the generator point G, i.e. M = mG. The reverse mapping function rmap() extracts m from a given point mG. The mapping function

$$map : m \rightarrow mG \text{ with } m \in \mathbb{GF}(p)$$

exhibits the additive homomorphic property, because

$$\begin{aligned} M_1 + M_2 + \ldots &= map(m_1 + m_2 + \ldots) \\ &= (m_1 + m_2 + \ldots)G \\ &= m_1G + m_2G + \ldots \end{aligned}$$

holds, where m1, m2, · · · ∈ GF(p).

The mapping function is not security relevant, i.e. it neither increases nor decreases the security of the EC-ElGamal encryption scheme. Note that the reverse mapping function is equivalent to solving the discrete logarithm problem over an elliptic curve, which represents a computational drawback of this scheme. The reader device must solve rmap() using a brute force approach, because it does not know the sum of the sensed data values m beforehand. However, rmap() is only performed on the powerful reader device and the maximum length of the final aggregation is assumed to be small enough in realistic WSNs (e.g., 3 byte) allowing a successful brute force approach resulting in m.

ALGORITHM / PSEUDOCODE

Algorithm 1 EC-ElGamal encryption

Require: public key Y, plaintext m
Ensure: ciphertext (R, S)
1: choose random $k \in [1, n-1]$
2: $M := map(m)$
3: $R := kG$
4: $S := M + kY$
5: **return** (R, S)

Algorithm 2 EC-ElGamal decryption

Require: secret key x, ciphertext (R, S)
Ensure: plaintext m
1: $M := -xR + S$
2: $m := rmap(M)$
3: **return** m

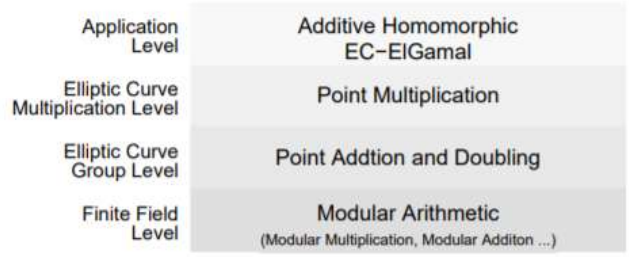

Figure 3: EC-ElGamal abstraction levels

2. A SIMPLE FULLY HOMOMORPHIC ENCRYPTION SCHEME AVAILABLE IN CLOUD COMPUTING.

Author: Li, Jian, et al.

Year and Journal: 2012, IEEE

TECHNIQUE IMPLEMENTED AND MOTIVATION

In this paper Marten van Dijk, Craig Gentry, Shai Halevi and Vinod Vaikuntanathan presented a **fully homomorphic encryption scheme, the DGHV scheme** [2], which uses many of the tools of Gentry's construction, but which does not require ideal lattices. Instead, they show that the somewhat homomorphic component of Gentry's ideal lattice-based scheme can be replaced with a very simple somewhat homomorphic scheme that uses integers. The scheme is therefore conceptually simpler than Gentry's ideal lattice scheme, but has similar properties with regards to homomorphic operations and efficiency.

A homomorphic public key encryption scheme has four algorithms: the usual **KeyGen**, **Encrypt**, **Decrypt**, and additional treatment algorithm **Evaluate**. The algorithm **Evaluate** takes as input a public key pk, a circuit C, a tuple of ciphertexts and outputs another ciphertext c.

When the noise r is sufficiently smaller than the secret key p, this simple scheme is both additively and multiplicatively homomorphic for shallow arithmetic circuits.

ALGORITHM / PSEUDOCODE:

KeyGen: The key is an odd integer, chosen at random in some prescribed interval.

Encrypt: To encrypt a bit $m \epsilon \{0,1\}$, set the ciphertext as an integer whose residue mod p has the same parity as the plaintext. Namely, set $c = pq + 2r + m$, where the integers q, r are chosen at random in some other prescribed intervals, such that 2r is smaller than p/2 in absolute value.

Decrypt : (p, c): Output (c mod p) mod 2.

3. PERFORMANCE ANALYSIS OF HOMOMORPHIC CRYPTOSYSTEM ON DATA SECURITY IN CLOUD COMPUTING

Author: Beyene, Mebiratu, and K. Raja Shekar
Year and Journal: 2019, IEEE

TECHNIQUE IMPLEMENTED AND MOTIVATION

In this paper Beyene, Mebiratu, and K. Raja Shekar talks about **SDC fully homomorphic system [3]** . The DGHV scheme [2] support addition and multiplication on ciphertexts, however, **neither of them** has **referred** to the **cipertext retrieval algorithms**. As to DGHV scheme, c = pq + 2r + m, when the client wants to retrieve some contents mindex, he encrypts the keywords cindex = pq + 2r + mindex and delivers cindex to the server. On receiving cindex, server reads the ciphertexts, utilizing the cipertext retrieval algorithm, and then he gets the result. Analyzing the scheme, for both q and r are changing at random, the cipertext retrieval algorithm Retrieval may be R = (ci - cindex) mod p mod 2, only when R = 0, gets the desired result, therefore p, the private key should be transferred to server, which seems terribly insecure. As to Gen10 scheme of the form c = pq + m, the cipertext retrieval algorithm R may be R = (ci - cindex) mod q, and q should be transferred to server, which seems much more secure than DGHV scheme, yet once q contains in the server, utilizing c mod q, the plaintext leaks out. Therefore, both the **DGHV scheme and the Gen10 scheme are insecure in the cloud computing, especially in the aspect of ciphertext retrieval,** so the proposed SDC scheme, derived from Gentry cryptosystem, may be a good candidate for a simple symmetric homomorphic encryption scheme described below.

Obviously, the ciphertext retrieval algorithm R in the proposed **SDC scheme avoids the fatal weakness of the DGHV scheme**. In the SDC scheme, transferring q to the server merely, the server can complete the process of ciphertext retrieval successfully, without plaintext leak out, because the process of decryption uses the private key p while the retrieval process uses the integer q, which is entirely different. Thus satisfies both the demand of ciphertext retrieval and data security.

ALGORITHM / PSEUDOCODE:

KeyGen(p): The key is a random P-bit odd integer p.

Encrypt (p, m): To encrypt a bit $m \in \{0,1\}$, output the ciphertext $c = m + p + r * p * q$, where r is a random R-bit number and q is a constain Q-bit big integer.

Decrypt (p, c): Output (c mod p).

Retrieval(c): $\boldsymbol{R} = (c_i - c_{index})$ mod q. When the client wants to retrieve contents m_{index}, he encrypts the keywords $c_{index} = m_{index} + p + r * p * q$ and delivers c_{index} to the server. On receiving c_{index}, server reads the ciphertexts, computing $\boldsymbol{R} = (c_i - c_{index})$ mod q, once $\boldsymbol{R} = 0$, ciphertext retrieval succeeds, and c_i is the desired result.

4. NON-INTERACTIVE EXPONENTIAL HOMOMORPHIC ENCRYPTION ALGORITHM

Author: Chen, Liang, et al.
Year and Journal: 2012, IEEE

TECHNIQUE IMPLEMENTED AND MOTIVATION

Chen, Liang, [4] have proposed the **Non-interactive Exponential Homomorphic Encryption Scheme** in this paper. They described the problem of non-interactive evaluation of encrypted functions (EEF) using a very simple and well formulated example:

- Alice (the originating host) has an algorithm to compute a function f.
- Bob (the remote host) has an input x and is willing to compute f(x).
- Alice wants Bob to learn nothing ―substantial‖ about f.
- Bob should not need to interact with Alice during the computation of f(x).

The protocol for non-interactive EEF proposed by Chen, Liang [4] is shown in figure 7 as follows:

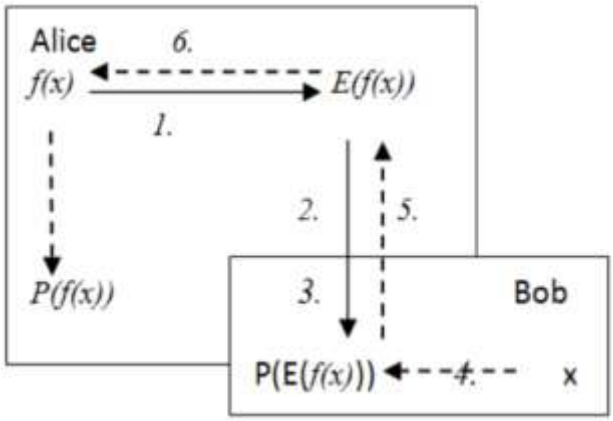

Figure 7: Encryption/Decryption of NEHE

1. Alice will encrypt f.
2. Alice will creates a program P(E(f)) which implements E(f).
3. Alice will send P(E(f)) to Bob.
4. Bob will execute P(E(f)) at x.
5. Bob will send P(E(f))(x) to Alice.
6. Alice will decrypt P(E(f))(x) and obtain f(x).

ALGORITHM / PSEUDOCODE:

The following **exponential homomorphic encryption algorithm E() is established on the RSA**.

a) Produce the public and private key:

Step 1. Choose three large prime numbers p, q and r. let n = pq, N= nr = pqr.

Step 2. Randomly selected the encrypting key e, and let e be co-prime with (p-1) (q-1).

Step 3. Calculate the multiplicative inverse of e, named d, by using the Euclid extension algorithm to meet the equation $ed \equiv 1 \bmod (p-1)(q-1)$.

Step 4. Then $d \equiv e^{-1} \bmod (p-1)(q-1)$.

N is public, and e,d,n are private keys.
Three prime numbers p, q and r are no longer needed, but can not be leaked.
$K \in z+$, $\omega^k < n$ and, ω is co-prime with n.

b) Encryption algorithm E:

$\theta = E(\omega^k) = (\omega^k \omega^{ed-1}) \bmod N$

c) Decryption algorithm D:

$D(\theta) = \theta \bmod n$

d) The proof of correctness of EHA:

$D(E(\omega^k)) = ((\omega^k \omega^{ed-1}) \bmod N) \bmod n$

Since N = nr, and then

$D(E(\omega^k)) = ((\omega^k \omega^{ed-1}) \bmod N) \bmod n = (\omega^k \omega^{ed-1}) \bmod n = \omega^k \omega^{(p-1)(q-1)} \bmod n = \omega^k \times 1 = \omega^k$

e) Security analysis of EHA:

It is based on the difficulty of factoring. Because e, d are not public, and factorization of open large number N = pqr is difficult, the opponent cannot factorize N, so ed cannot be deduced. And thus k cannot be deduced, i.e. k is not leaked.

5. COMPARATIVE STUDY OF HOMOMORPHIC ENCRYPTION METHODS FOR SECURED DATA OPERATIONS IN CLOUD COMPUTING

Author: Rangasami, Kanagavalli, and S. Vagdevi
Year and Journal: 2017, IEEE

TECHNIQUE IMPLEMENTED AND MOTIVATION

In this paper Rangasami, Kanagavalli, and S. Vagdevi talks about **Algebra Homomorphic Encryption Scheme Based On Updated ElGamal (AHEE). [5]** This is the modified form of the digital signature standard DSS presented by the NIST in America. The security of the AHEE is IND-CPA which is the highest level of the security of AHEE. Additive homomorphism of this algorithm refers the same k for encryption but uses the random number of k in E1() which makes AHEE able to resist plaintext attack. The AHEE is the subset of the fully homomorphism. AHEE has been proved to be secure. This description of fully homomorphism is advanced by Rivest, Adleman and Dertouzos.

ALGORITHM / PSEUDOCODE:

Step 1: select any two prime numbers say p and q

Step 2: calculate the product of those two prime numbers. Say N = p * q. where p and q being confidential and N is public.

Step 3: select random number x and a root g of GF(p). where g and x are smaller than p.

Step 4: calculate $y = g^x \bmod p$. use this y for the encryption.

Step 5: encryption will be performed in following two steps:

1. Select random integer number r and apply following homomorphic encryption.
   $E_1(M) = (M+r*p) \bmod N$.
2. Select random integer number k, and the encryption algorithms are:
   $E_g(M) = (a,b) = (g^k \bmod p, y^k E_1(M) \bmod p)$

Step 6: Decrypted algorithm D_g () is $M=b\times (a^x)^{-1} \pmod p$.

Figure 8: AHEE Homomorphic scheme

Multiplicative: Eg (M1M2) = Eg (M1)·Eg (M2), or M1.M2=Dg (Eg (M1)·Eg (M2)).

Additive: Eg (M1+M2) = Eg (M1) ⊕Eg (M2), or

M1+M2=Dg (Eg (M1) ⊕Eg (M2)).

Above equations show multiplicative and additive Homomorphic Encryption properties of Algebra Homomorphic Encryption Scheme Based On Updated ElGamal (AHEE).

6. THE HOMOMORPHIC OTHER PROPERTY OF PAILLIER CRYPTOSYSTEM

Author: Sridokmai, Tanyaporn, and Somchai Prakancharoen**.**
Year and Journal: 2015, IEEE

TECHNIQUE IMPLEMENTED AND MOTIVATION

In this paper Sridokmai, Tanyaporn, and Somchai talks about the **Paillier cryptosystem**, [6] invented by and named after Pascal Paillier in 1999, which is a **probabilistic asymmetric algorithm for public key cryptography**. The problem of computing *n*-th residue classes is believed to be computationally difficult. The decisional composite residuosity assumption is the intractability hypothesis upon which this cryptosystem is based.

The scheme is an **additive homomorphic cryptosystem**; this means that, given only the public key and the encryption of m1 and m2 , one can compute the encryption of m1+m2 this makes it very useful for privacy preserving application.

Paillier scheme is the most known, and maybe the most efficient partially homomorphic encryption scheme available.

ALGORITHM / PSEUDOCODE:

Key generation

1. Choose two large prime numbers p and q randomly and independently of each other such that $\gcd(pq, (p-1)(q-1)) = 1$. This property is assured if both primes are of equal length.[1]
2. Compute $n = pq$ and $\lambda = \operatorname{lcm}(p-1, q-1)$. lcm means Least Common Multiple.
3. Select random integer g where $g \in \mathbb{Z}^*_{n^2}$
4. Ensure n divides the order of g by checking the existence of the following modular multiplicative inverse: $\mu = (L(g^\lambda \bmod n^2))^{-1} \bmod n$,

   where function L is defined as $L(x) = \frac{x-1}{n}$.

   Note that the notation $\frac{a}{b}$ does not denote the modular multiplication of a times the modular multiplicative inverse of b but rather the quotient of a divided by b, i.e., the largest integer value $v \geq 0$ to satisfy the relation $a \geq vb$.

- **The public (encryption) key is (n, g).**
- **The private (decryption) key is (λ, μ).**

If using p,q of equivalent length, a simpler variant of the above key generation steps would be to set $g = n + 1, \lambda = \varphi(n)$, and $\mu = \varphi(n)^{-1} \bmod n$, where $\varphi(n) = (p-1)(q-1)$

Encryption

1. Let m be a message to be encrypted where $0 \leq m < n$
2. Select random r where $0 < r < n$ and $r \in \mathbb{Z}_n^*$ (i.e., ensure $gcd(r, n) = 1$)
3. Compute ciphertext as: $c = g^m \cdot r^n \bmod n^2$

Decryption

1. Let c be the ciphertext to decrypt, where $c \in \mathbb{Z}_{n^2}^*$
2. Compute the plaintext message as: $m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$

PERFORMANCE ANALYSIS

Characteristics	**EC-ElGamal** (paper-1)	**DGHC** (paper-2)	**SDC** (paper-3)	**NEHE** based on RSA (paper-4)	**AHEE** (paper-5)	**Paillier** (paper-6)
Salient feature	Superior performance and small ciphertext size	Conceptually simple and is both additively and multiplicatively homomorphic	The server can complete the process of ciphertext retrieval successfully, without plaintext leak out	It provides high level of security.	The security of the AHEE is IND-CPA which is the highest level of the security of AHEE	Most known, and the most efficient partially homomorphic encryption scheme available.
Homomorphic encryption type	Additive	Fully	Fully	Fully	Multiplicative	Additive
Speed	**medium**	**low**	**high**	**high**	**high**	**medium**
Versality	**low**	**high**	**low**	**low**	**low**	**high**
Applications	In Hybrid systems	Active networks, e-commerce based on mobile agent, computing grid	Secure multi-party computation, electronic voting and mobile cipher	Active networks, e-commerce based on mobile agent, computing grid	To secure internet, Banking and credit card transaction	e-voting system, threshold scheme
Privacy of data	Is ensured in communication and storage processes	Is ensured in communication and storage processes	Is ensured in communication and storage processes	Is ensured in communication and storage processes	Is ensured in communication and storage processes	Is ensured in communication and storage processes
Platform	Cloud Computing	Cloud Computing	Cloud Computing	Cloud Computing	Cloud Computing	Cloud Computing

FURTHER DETAILED PERFORMANCE ANALYSIS FROM DATA TAKEN FORM THE RESEARCH PAPERS

TABLE 4: encryption time for a given data size

Byte	SDC FHE (Mic	Paillier (Micro sec)	RSA (Micro sec)
860	0.04	2.46	0.094
1316	0.13	4.827	0.169
3394	0.28	10.34	0.319
6790	0.66	21.407	0.717
13582	1.56	44.231	1.277
17591	1.98	56.94	1.713
18974	2.32	60.43	1.862
27166	3.3	90.12	2.588
54334	6.6	189.066	5.302

TABLE 5: decryption time with a given data size

Byte	SDC FHE (Mic ro	Paillier (Micro sec)	RSA (Micro sec)
860	0.0001	1.668	0.416
1316	0.00012	2.755	0.608
3394	0.001	5.09	1.512
6790	0.005	10.648	2.813
13582	0.009	23.708	5.699
17591	0.012	28.458	7.387
18974	0.014	31.606	7.541
27166	0.021	37.351	11.392
54334	0.032	74.304	21.103

Table 4 and 5 shows that **SDC FHE algorithm takes less time for encryption as well as decryption** as compared with Pallier and RSA algorithm for given different file size, especially when the file size is increasing the change is growing fast.[2][3]

Table 7: decryption throughput with a given data size

	Encryption throughput		
Byte	SDC FHE (Micro sec)	Paillier (Micr o sec)	RSA (Micr o sec)
860	17551.02041	349.593	9148.94
1316	10045.80153	272.633	7786.98
3394	11825.78397	328.24	10639.5
6790	10210.52632	317.186	9470.01
13582	8689.699296	307.07	10635.9
17591	8861.964736	308.939	10269.1
18974	8174.924601	313.983	10190.1
27166	8207.250755	301.443	10496.9
54334	8133.832335	287.381	10247.8

TABLE 6: Encryption throughput with a given data size

	Decryption throughput		
Byte	Derived SDC FHE	Paillier	RSA
860	**8600000**	**515.58753**	**2067.307692**
1316	10966666.67	477.67695	2164.473684
3394	3394000	666.79764	2244.708995
6790	1358000	637.67844	2413.793103
13582	1509111.111	572.88679	2383.225127
17591	1465916.667	618.13901	2381.345607
18974	1355285.714	600.32905	2516.111921
27166	1293619.048	727.31654	2384.655899
54334	1697937.5	731.23923	2574.705018

To measure the performance of the algorithm it is compulsory to compute the throughput time for encryption and decryption methods. The throughput is calculated as the total user data in bytes encrypted/encryption time (Byte/Micro second). As the throughput increases, power consumption (the delay) decreases. Table 6 and 7 shows the encryption and decryption throughput for given user data for **Paillier and RSA has the lowest throughput but SDC FHE have the highest throughput** for given user is measured by dividing the total cipher text in binary and total decryption time in each algorithm.[2][3]

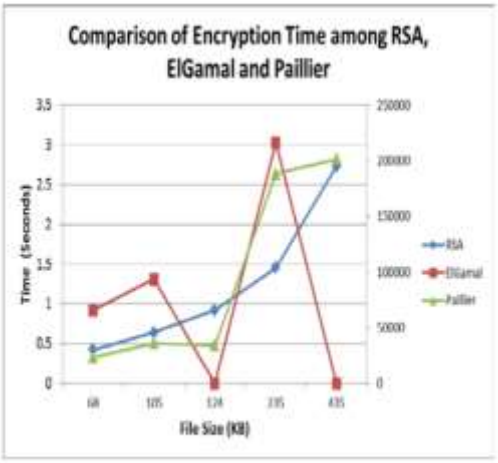

Figure 1: Comparison of Encryption Time among RSA, ElGamal and Paillier

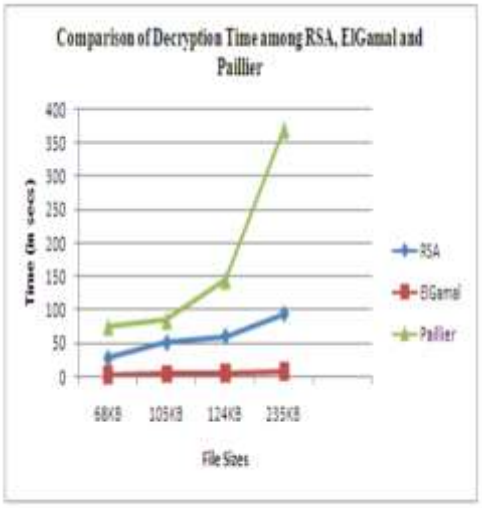

Figure 2: Comparison of decryption Time among RSA, ElGamal and Paillier

Figure 1 shows the comparison of encryption time in seconds among RSA, ElGamal, and Paillier. The secondary Y-axis represents the encryption time of Paillier because of huge time difference of Paillier as compared to RSA and ElGamal. **RSA showed better performance over ElGamal and Paillier in term of encryption time and ElGamal showed better performance over RSA and Paillier in term of decryption time** as shown in Figure 2.[4]

From the analysis, we can be concluded that SDC FHE encryption/decryption algorithm has better performance based on execution speed relative to the other homomorphic cryptographic algorithms. However it is tough to implement SDC in practical. **Only the paillier encryption is simple to implement and stable version is available.** While RSA is also not stable for homomorphic encryption hence it is not being used. Also further research is going on in order to arrive at the stable version of SDC as well.

REFERENCES

[1] Ugus, Osman, et al. "Optimized implementation of elliptic curve based additive homomorphic encryption for wireless sensor networks." *arXiv preprint arXiv:0903.3900* (2009).

[2] Li, Jian, et al. "A simple fully homomorphic encryption scheme available in cloud computing." *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*. Vol. 1. IEEE, 2012.

[3] Beyene, Mebiratu, and K. Raja Shekar. "Performance Analysis of Homomorphic Cryptosystem on Data Security in Cloud Computing." *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE, 2019.

[4] Chen, Liang, et al. "Non-interactive exponential homomorphic encryption algorithm." *2012 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*. IEEE, 2012.

[5] Rangasami, Kanagavalli, and S. Vagdevi. "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)*. IEEE, 2017.

[6] Sridokmai, Tanyaporn, and Somchai Prakancharoen. "The homomorphic other property of Paillier cryptosystem." *2015 International Conference on Science and Technology (TICST)*. IEEE, 2015.