Cluster Computing (2023) 26:685–699
https://doi.org/10.1007/s10586-022-03693-1

Design and realization of a secure multiplicative homomorphic encryption scheme for cloud services

Christiana Zaraket[1] · Khalil Hariss[1] · Sandro Ephrem[1] · Maroun Chamoun[1] · Tony Nicolas[1]

Received: 10 January 2022 / Revised: 22 June 2022 / Accepted: 4 July 2022 / Published online: 4 August 2022

Abstract
Cloud technology is a modern data storing technique that gives opportunities for outsourcing of storage and computation. While storing sensitive data (such as medical records) at the cloud side can violate personal privacy, Homomorphic Encryption (HE) was presented as a special type of encryption that leverages users' privacy by allowing computation over cipher-texts at the cloud side. In our prior work, we developed and tested a new additive HE scheme (SAVHO) that has been proven to be a good competitor for the Paillier scheme. The aim of this paper is to build a new secure and efficient multiplicative HE scheme competitor for the well-known multiplicative HE scheme ElGamal. The proposed scheme is called Logarithm Operation for Randomization and Multiplicative Homomorphic Encryption scheme (LORMHE). Security and performance analyses have proven its high level of security and its efficiency in comparison with ElGamal scheme and its efficiency for real world applications.

Keywords Cloud technology · Homomorphic Encryption (HE) · Security and performance analyses · Additive HE · Multiplicative HE

1 Introduction

Cloud computing is a modern technology that has matured over the recent years. Hence, more and more companies are expected to migrate to the cloud due to several advantages provided by the concerned solution [1]. Some major advantages of such solution are the virtualization, the scalability, the disaster recovery, the high computation and storing capabilities, etc. Despite all of these advantages, some security issues remain a key barrier to organizations shifting to the cloud. So far, data encryption appears to be a reasonable answer to this issue. However, at some point, encrypted data becomes worthless unless it is decrypted. In other words, if the client needs to apply operations on its encrypted data stored at the cloud side, he must reveal his secret parameters to the cloud for making the latter able to convert the data back into plain-text before performing computations on it. Then, the result is converted back into cipher-text. Hence, the latter operation leaves classified data exposed to non trusted parties. For this reason, companies demand an encryption type that provides scalable, secure, and practical computation services that can be employed in order to save and operate securely on the data at the same time. In this scenario, the encryption type needed is the HE scheme [2], since it is a technique that allows applying mathematical operations on encrypted data without disclosing the initial data contents or the decryption key, as illustrate in Fig. 1.

There are four types of HE schemes [3] that are presented as follows: Partially Homomorphic Encryption (PHE), Somewhat Homomorphic Encryption (SWHE), Fully Homomorphic Encryption (FHE) and Switchable Homomorphic Encryption (SHE). PHE is the type that permits computation of an unlimited number of additions

✉ Christiana Zaraket
christiana.zaraket@net.usj.edu.lb

Khalil Hariss
khalil.hariss1@usj.edu.lb

Sandro Ephrem
sandro.ephrem@net.usj.edu.lb

Maroun Chamoun
maroun.chamoun@usj.edu.lb

Tony Nicolas
tony.nicolas@usj.edu.lb

[1] ESIB, Saint Joseph University, Mar Roukoz, Beirut, Lebanon

or multiplications on encrypted data, but not both of them at the same time [4]. SWHE allows a limited number of additions and multiplications to be done simultaneously on the encrypted data, beyond which the result of the calculation returns an erroneous answer [2]. FHE permits an arbitrary number of additions and multiplications to be made on the encrypted data [4]. Existing FHE crypto-systems, on the other hand, have numerous issues, such as prohibitive performance and storage costs for the bulk of practical applications. Lastly, SHE, which debuted in 2014, allows users to profit from the efficiency of PHE schemes by combining them and obtaining the same property as an FHE scheme but with greater performance [5]. This paper is concerned with analyzing PHE schemes, with a focus on designing a new secure and efficient multiplicative HE scheme, competitor to ElGamal crypto-system [6], to be used in cases where multiplication operations are required. As we mentioned previously, in our prior work we designed an additive HE scheme that we named SAVHO [7]. Security and Performance analyses of the SAVHO scheme have shown its high level of security and its efficiency in implementation, which make him a good competitor for the well known additive Paillier crypto-system [8]. Thus, our long term goal from designing both the SAVHO scheme (summarized in Table 12 in Appendix) and the LORMHE crypto-system (detailed in this paper) is to combine them together to form a SHE similar to the work done in [5]. Hence, the resultant scheme should allow to perform all types of operations over encrypted data while maintaining simultaneously the security and the efficiency in implementations.

The remainder of this paper is structured as follows:

- Section 2: a brief overview of previous work is presented.
- Section 3: an introduction to the HE and the ElGamal scheme is given.
- Section 4: a new scheme called LORMHE (Logarithm Operation for Randomization and Multiplicative Homomorphic Encryption scheme) is introduced.
- Section 5: the resistance of the LORMHE against various types of attacks is presented by applying different security tests and establishing a theoretical crypt-analysis.
- Section 6: the performance study of LORMHE is completed by establishing a comparison between LORMHE and ElGamal schemes in terms of encryption, decryption and homomorphic behavior execution time and storage overhead.
- Section 7: conclusion and future work are given.

2 Related work

Since the creation of the HE concept, researchers in academia as well as in industry have been attempting to improve in this sector by creating secure and efficient schemes practical for real-world uses.

Ron Rivest, Adi Shamir, and Leonard Adleman proposed the first HE scheme in the cryptography world in 1978 [9]. The RSA scheme was, in particular, a PHE scheme that can handle an unlimited number of multiplications over the cipher-texts. Then, in 1982, Shafi Goldwasser and Silvio Micali introduced an additive HE scheme that reached a surprising level of security [10]. The latter's shortcoming is that it can only encrypt a single bit. As a correction, Pascal Paillier devised a successful secure encryption scheme that was also additive HE [11]. Taher ElGamal developed and published a new multiplicative HE scheme a few years later, in 1985, which is still utilized in many applications today [6]. Until 2009, when Craig Gentry invented the first FHE [2], all of the schemes that were created could only handle one of two types of operations: addition or multiplication. Craig Gentry's method

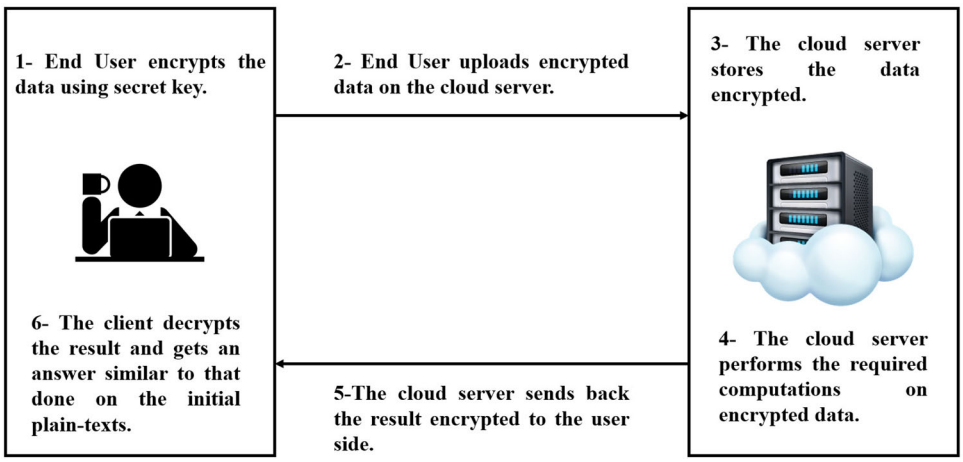


Fig. 1 A general framework for HE schemes

was built on lattices, which made it complicated and unsuitable for real-world use. Following Gentry's work, in 2010, the DGHV scheme was introduced by Dijk et al. [12]. This scheme is based on integers. It is constructed based on two steps: SWHE and bootastrapping. Its SWHE can return an erroneous result in the case of applying a huge number of additions and multiplications due to noise increase after each homomorphic operation. Bootstrapping is a refresh mechanism introduced to make it FHE by reducing the level of non wanted noise after each homomorphic operation while keeping the primitive plain-text unchanged. Although DGHV permits the application of additive and multiplicative HE operations simultaneously, it suffers from high computational complexity and storage overhead. Following DGHV, several crypto-systems were developed, each with its own set of issues. Therefore, until now, existing FHE schemes are unpractical for real world implementations. Instead of utilizing such FHE techniques, one can apply the well-known Switchable Homomorphic Encryption approach (SHE), which was developed in 2014 [5]. The latter is a hybrid of two partially HE crypto-systems. An additive HE crypto-system may be preferred if additive operations are required; Otherwise, a multiplicative HE scheme may be better. The SHE method can be utilized if both of these are necessary. Paillier and ElGamal were the only two PHE schemes combined by the SHE technique, as indicated in [5]. Implementation has shown that these two partially HE schemes combined together performed better than the existing FHE schemes while giving the requisite amount of security.

HE schemes, in all their forms, are increasingly being employed in numerous sectors. Especially those seeking security while operating on outsourced data.

Paillier's additive HE crypto-system, for example, was recently employed in e-voting in 2020, as reported in [13] by Miaomiao Zhang and Steven Romero. ElGamal multiplicative HE scheme was used as well in achieving privacy-preserving Iris identification as indicated in [14] in 2019. Furthermore, in 2021, a paper titled "Location Based Recommendation Services Using Switchable Homomorphic Encryption" discussed the importance of using SHE [15].

3 Homomorphic encryption

The term "homomorphism" is derived from the Greek language. $o\mu oc$-homos means the same and $\mu o\rho\phi\eta$-morphe meaning is structure [16]. As shown in [17], if $(E, \bigstar)$ and $(F, \bullet)$ are two magmas , then a map f from E to F is a homomorphism from $(E, \bigstar)$ to $(F, \bullet)$ if and only if:

$\forall(a, b) \in E \times E : f(a \bigstar b) = f(a) \bullet f(b)$

The definition of a homomorphism has lately been transferred to the world of cryptography. In particular, in a HE scheme, the encryption function is the homomorphism. The spaces of plain-texts $\mathcal{P}$ and cipher-texts $\mathcal{C}$, provided with specific internal composition law, represent the two magmas being mapped from and to. This paper concentrates on using the multiplicative HE crypto-system type which is a HE scheme that can only handle multiplicative operations. In other words, it must mathematically validate the following property: for n plain-texts $(m_1, \ldots, m_n) \in \mathcal{P} \times \cdots \times \mathcal{P}$:

$$g(Enc_K(m_i)) = Enc_K\left(\prod_{i=1}^{n} m_i\right) \tag{1}$$

where $Enc()$ denotes the encryption function, K the secret key and $g \in \{\prod_{i=1}^{n}, \sum_{i=1}^{n}\}$ [11].

The fundamental advantage of this homomorphic property is that any third party (trusted or not), may compute an encryption of $\prod_{i=1}^{n} m_i$ from encrypted messages without knowing the secret key or the content of the plain-texts messages m_i.

3.1 ElGamal crypto-system

A condensed description of the ElGamal scheme construction is given in Table 1 as listed in [6].

4 LORMHE scheme

Motivated by the importance of the ElGamal scheme, a new multiplicative HE scheme (LORMHE) is devised that is competent to ELGamal in terms of implementation efficiency and reduced storage overhead.

The construction steps of LORMHE scheme are presented in details in Sect. 4.1, while the homomorphic behavior of it will be proven in Sect. 4.3.

4.1 Scheme construction

Three basic functions are used in any HE scheme construction: key generation, encryption and decryption processes. Hence, the basic functions of the LORMHE scheme are listed below:

4.1.1 Key generation

The secret key is made up of a pair of random integers denoted by (a, g) such that the size of a is 45 bits and the g's one is greater or equal to 3000 bits. As will be discussed in the security analysis given in Sect. 5, these

constraints are imposed in order to achieve the required level of security.

4.1.2 Encryption process

The cipher-text c related to a message $m \in \mathbb{N}^*$ is given by the following equation:

$$c = Enc_{(a,g)}(m) = g^{log_a(\frac{m}{l})} \times r \quad (2)$$

where l and r are picked randomly in $\mathbb{N}^*$ such that they are formed of 40 bits.

4.1.3 Decryption process

Theorem 1 *Having the secret key parameters* (a, g) *and the random parameters* l *and* r*, retrieving the plain-text* m *could be done by applying the following decryption function*:

$$Dec_{(a,g)}(c) = e^{(ln(c)-ln(r))\times\frac{ln(a)}{ln(g)}+ln(l)} \quad (3)$$

Lemma 1 *The natural logarithmic function* $ln(x)$ *has the following properties*:

$ln(a \times b) = ln(a) + ln(b)$, *for* $a, b > 0$,
$ln(a^b) = b \times ln(a)$, *for* $a > 0$,
$ln(\frac{a}{b}) = ln(a) - ln(b)$, *for* $a, b > 0$.

Lemma 2 *The function* e^x *has the following property*:

$e^{ln(a)} = a$, *for* $a > 0$.

Proof of theorem 1

$$\begin{aligned}
Dec_{(a,g)}(c) &= e^{(ln(c)-ln(r))\times\frac{ln(a)}{ln(g)}+ln(l)} \\
&= e^{(ln(g^{log_a(\frac{m}{l})}\times r)-ln(r))\times\frac{ln(a)}{ln(g)}+ln(l)} \\
&= e^{(ln(g^{log_a(\frac{m}{l})})+ln(r)-ln(r))\times\frac{ln(a)}{ln(g)}+ln(l)} \\
&= e^{ln(g^{log_a(\frac{m}{l})})\times\frac{ln(a)}{ln(g)}+ln(l)} \\
&= e^{log_a(\frac{m}{l})ln(g)\times\frac{ln(a)}{ln(g)}+ln(l)} \\
&= e^{\frac{ln(\frac{m}{l})}{ln(a)}\times ln(a)+ln(l)} \\
&= e^{ln(m)-ln(l)+ln(l)} \\
&= e^{ln(m)} \\
&= m
\end{aligned}$$

4.1.4 Random parameters (l and r) dynamic generation

As given in both Eqs. 2 and 3, the two random parameters (l, r) that are generated for the encryption procedure, should be re-used for the decryption procedure. Thus, the latter mechanism will add additional burden in terms of storing, managing and sharing the random parametes (l, r) between the communicating entities especially when dealing with large number of plain-texts. To overcome the concerned challenges, a dynamic approach is designed in order to manage and facilitate the random generation and the sharing of both l and r. Hence, starting from a plain-text vector $X = [x_1, x_2, x_3, \ldots x_N]$ of size N, the dynamic mechanism is given by the algorithm below:

Algorithm 1 (l, r) *Dynamic Generation*

Input: Secret Key (g, a)*, Plain-text Vector Size* N

▷ *Random Integer* k *Generation for each Encryption Session*
$k \leftarrow Random\_Integer\_Generation(seed)$
▷ *Random Vectors Generation* L *and* R *using the secret element* g
$\{L = [l_1, l_2, ..., l_N], R = [r_1, r_2, ..., r_N]\} \leftarrow SHA256(g)$
▷ *Permutation Box* π *Generation for each Encryption Session based on* k
$\pi = [\pi_1, \pi_2, ..., \pi_N] \leftarrow PBox\_Generation(k, N)$
▷ *Generating* R_π, L_π *for each Encryption Session*
$R_\pi = \pi(R) = [r_{\pi_1}, r_{\pi_2}, r_{\pi_3}, ..., r_{\pi_N}], L_\pi = \pi(L) = [l_{\pi_1}, l_{\pi_2}, l_{\pi_3}, ..., l_{\pi_N}]$

return R_π, L_π
end

An illustration of the dynamic generation of both l and r is presented Fig. 2.

After using the dynamic generation algorithm given above in Algorithm 1, there is no need to store the different values of l and r. In this case, the two communicating parties (one performs the encryption and the other performs the decryption) need to store for each encryption session only the specified triple (g, a, k). Thus, using (g, a, k) it is possible to generate the same vectors R_π and L_π and perform the different cryptographic operations when needed.

4.2 Symmetric or asymmetric HE scheme?

Symmetric encryption techniques are crypto-systems that use the same key for both encryption and decryption. While the asymmetric encryption system, commonly known as public key encryption, employs mathematically linked public and private keys pairs to encrypt and decrypt [18].

Table 2 summarizes the key differences between these two types of encryption schemes.

Our crypto-system is classified as symmetric encryption scheme since the same pair of keys (a, g) is used to encrypt and decrypt a given message. These systems can be utilized in a variety of real-world applications, including:

Table 1 ElGamal scheme

Key generation	$\mathbb{G}$: a cyclic group of order q with generator g where q is a prime number x: a random value in $\mathbb{Z}_q$ h: $h = g^x$ Public Key: $\mathbf{pk} = (\mathbb{G}, q, g, h)$ Secret Key: $sk = x$
Encryption procedure	For a given message $m \in \mathbb{G}$, and a random number $r \in \mathbb{Z}_q$, the cipher-text is given by: $\mathbf{c} = Enc_{pk}(m) = (c_0, c_1)$ where $c_0 = g^r$ and $c_1 = h^r m$
Decryption function	$Dec_{sk}(\mathbf{c}) = c_1 / c_0^x$
Homomorphic multiplication	For two plain-texts m_1 and m_2, $c_1 = Enc_{pk}(m_1) = (g^{r_1}, h^{r_1} m_1)$ $c_2 = Enc_{pk}(m_2) = (g^{r_2}, h^{r_2} m_2)$ $c_1 \times c_2 = (g^{r_1+r_2}, h^{r_1+r_2}(m_1.m_2)) = Enc_{pk}(m_1.m_2)$
Crypt-analysis	Theoretical Security: ElGamal scheme is indistinguishable under chosen-plain-text attack (IND-CPA) under the decisional Diffie–Hellman (DDH) assumption over $\mathbb{G}$. Security by Implementation: As given in [6], ElGamal recommends minimum 3072 bits for the q's size to be considered secure.

- Payment applications, such as card transactions, where personally identifiable information (PII) must be safe-guarded to avoid identity theft or fraudulent charges
- Validations to confirm that the sender of a message is who he claims to be
- Random number generation or hashing [21]

However, our primary goal of creating the symmetric crypto-system LORMHE is to merge it with SAVHO [7] via SHE technique [5]. SHE is reliant on two separate clouds. The user is required to divide and then transfer his private keys to the two clouds in order to do a partial decryption while operating on cipher-texts. In other words, whether the system is symmetric or asymmetric is irrelevant. As a result, we opt to profit from the symmetric schemes features while creating our scheme LORMHE.

4.3 Homomorphic behavior

Theorem 2 *To be considered as a multiplicative HE scheme, LORMHE crypto-system must verify Eq. 1.*

Remark 1 To facilitate the calculation in the proof of theorem 2, the computation of Eq. 1 has been done on two cipher-texts, namely by checking that

$Dec_{(a,g)}(c_1.c_2)$ equals $m_1 m_2$.

Proof of theorem 2 Let $c_1 = Enc_{(a,g)}(m_1) = g^{log_a(\frac{m_1}{l_1})} \times r_1$ and $c_2 = Enc_{(a,g)}(m_2) = g^{log_a(\frac{m_2}{l_2})} \times r_2$ be two different cipher-texts obtained from applying LORMHE encryption function respectively on two plain-texts m_1 and m_2. Therefore,

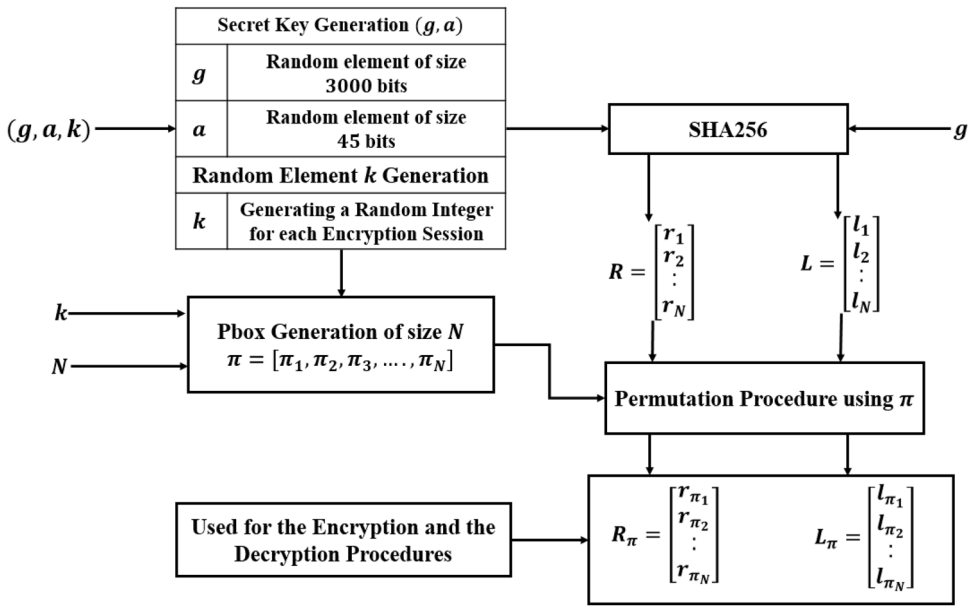


Fig. 2 L and R dynamic generation

Table 2 Symmetric vs asymmetric encryption schemes [19, 20]

Key differences	Symmetric encryption	Asymmetric encryption
Size of cipher-text	Smaller cipher-text compares to original plain-text	Larger cipher-text compares to original plain-text
Data size	Used to transmit big data	Used to transmit small data
Key lengths	128 or 256-bit key size	RSA 2048-bit or higher key size
Security	Less secured due to use a single key for encryption and decryption	Much safer as two keys are involved in encryption and decryption
Speed	Symmetric encryption is a fast technique	Asymmetric encryption is slower in terms of speed

$$
\begin{aligned}
c_1.c_2 &= g^{log_a(\frac{m_1}{l_1})} \times r_1 \times g^{log_a(\frac{m_2}{l_2})} \times r_2 \\
&= g^{log_a(\frac{m_1}{l_1})+log_a(\frac{m_2}{l_2})} \times r_1 \times r_2 \\
&= g^{log_a(\frac{m_1 m_2}{l_1 l_2})} \times r_1 \times r_2
\end{aligned}
$$

Decrypting this result yields the following:

$$
\begin{aligned}
&Dec_{(a,g)}(c_1.c_2) \\
&= e^{[(ln(c_1.c_2)-ln(r_1)-ln(r_2))\frac{ln(a)}{ln(g)}+ln(l_1)+ln(l_2)]} \\
&= e^{[(ln(g^{log_a(\frac{m_1 m_2}{l_1 l_2})}\times r_1 \times r_2)-ln(r_1)-ln(r_2))\frac{ln(a)}{ln(g)}+ln(l_1)+ln(l_2)]} \\
&= e^{[(ln(g^{log_a(\frac{m_1 m_2}{l_1 l_2})})+ln(r_1)+ln(r_2)-ln(r_1)-ln(r_2))\frac{ln(a)}{ln(g)}+ln(l_1)+ln(l_2)]} \\
&= e^{[log_a(\frac{m_1 m_2}{l_1 l_2})ln(g)\frac{ln(a)}{ln(g)}+ln(l_1)+ln(l_2)]} \\
&= e^{[log_a(\frac{m_1 m_2}{l_1 l_2})ln(a)+ln(l_1)+ln(l_2)]} \\
&= e^{[\frac{ln(\frac{m_1 m_2}{l_1 l_2})}{ln(a)}\times ln(a)+ln(l_1)+ln(l_2)]} \\
&= e^{[ln(m_1 m_2)-ln(l_1 l_2)+ln(l_1)+ln(l_2)]} \\
&\quad + + = e^{[ln(m_1 m_2)-ln(l_1)-ln(l_2)+ln(l_1)+ln(l_2)]} \\
&= e^{[ln(m_1 m_2)]} \\
&= m_1 m_2
\end{aligned}
$$

Applying the same steps to a tuple of cipher-texts leads to verifying Eq. 1. Hence, This result ensures that the LORMHE scheme fulfills the multiplicative HE feature for a tuple of cipher-texts.

5 LORMHE security analysis

This section investigates the robustness of the LORMHE scheme against various types of attacks. Such investigations are carried out in two ways:

1. Theoretically: a deep mathematical analysis of its algebraic structure is performed that studies its resistance to known plain-text/cipher-text attacks.
2. Practically: specific security tests are applied over the new scheme as listed [23] that highlight its robustness in defending the statistical and the related key attacks and its verification to some mandatory properties.

5.1 Theoretical crypt-analysis: resistance to known plain-text/cipher-text attacks

As mentioned in [7], a scheme's resistance to known plain-text/cipher-text assaults is investigated by assuming the presence of an attacker with a tuple pairs of known plain-texts and their associated cipher-texts. This attacker is attempting to disclose the secret key of the scheme in question.

We recall that LORMHE's secret key is given by (a, g) and its encryption function is given by $c = Enc_{(a,g)}(m) = g^{log_a(\frac{m}{r})} \times r$. Hence, a attack model can be presented as given in the upcoming section

5.1.1 Oracle model

Starting with an attacker that has a t tuple pairs of cipher-texts/plain-texts denoted by $(m_1, c_1), \ldots, (m_t, c_t)$, in order to reveal the secret key and parameters of the scheme, he can do the following steps:

1. Building a System Using the Known Pairs of Plain-text/Cipher-text $(m_i, c_i)_{(1 \le i \le t)}$: the attacker is able to construct the following system

$$
\begin{aligned}
c_1 &= g^{log_a(\frac{m_1}{l_1})} \times r_1 \\
c_2 &= g^{log_a(\frac{m_2}{l_2})} \times r_2 \\
&\vdots \\
c_t &= g^{log_a(\frac{m_t}{l_t})} \times r_t
\end{aligned} \tag{4}
$$

2. Linearizing the Obtained System: the attacker can linearize the system given Eq. in 4 by applying the

natural logarithm function to each cipher-text as follows: $\forall i \in [1,t]$,

$$ln(c_i) = ln(g)log_a(\frac{m_i}{l_i}) + ln(r_i) = ln(g)\frac{ln(m_i) - ln(l_i)}{ln(a)} + ln(r_i)$$

. In order to make the prior system's writing easier, the attacker can assume that $\forall i \in [1,t]$, $A_i = ln(c_i)$, $B_i = ln(m_i)$, $X = ln(g)$, $Y = \frac{1}{ln(a)}$, $Z_i = ln(l_i)$ and $T_i = ln(r_i)$. Therefore, the new linear version of the system is given as follows:

$$\begin{cases} A_1 = XY(B_1 - Z_1) + T_1 \\ A_2 = XY(B_2 - Z_2) + T_2 \\ \vdots \\ A_t = XY(B_t - Z_t) + T_t \end{cases} \quad (5)$$

where, $\forall i \in [1,t]$, X, Y, Z_i and T_i are the unknown values while A_i and B_i are the known ones. As a result, the attacker is confronted with a system that has t equations and $2t + 2$ unknowns.

3. Mathematical Problem Formulation: the attacker is working with a system with infinite solutions, and disclosing the secret key is not feasible. As a conclusion, the security of the concerned scheme is based on the mathematical hardness of solving an indeterminate linear system [22].

5.2 Security tests

In this section, the immunity of the LORMHE scheme against attacks is investigated by implementation. Different security tests are implemented as given in [23] in order to determine the required level of the security parameters sizes for achieving a secure implementation that can resist against related key attacks and statistical attacks and assures the presence of avalanche effect. Security tests were implemented under Mathematica on a machine having the following technical specifications (Table 3):

A dynamic Graphical User Interface (GUI) was developed on Mathematica. This GUI allows the user to change the different security parameters sizes and to instantly evaluate their effects on the tests values. These values were found to be most optimal around a fixed value of the size of a, l and r, and getting closer to their ideal values for an increasing size of g (we recall that g is a part of the secret key and used during the encryption procedure given in Eq. 2). As result, the a, l and r values were easily identified and set, as illustrated in Table 4. Then, the main purpose of the following security tests is to determine the required size of the parameter g in order to achieve a secure implementation for the LORMHE scheme.

Table 3 Machine specification

Component	Specification
CPU	Intel(R) Core(TM) i7-8550U
CPU clock speed	1.99 Ghz
CPU cores	8
RAM size	16 GB
Operating system	Windows 10 Pro

5.2.1 Resistance against related key attacks

The Key Sensitivity (KS) test is used to evaluate the resilience of a given scheme against related key attacks. This test permits to measure the percentage of change at the bit level that may be made on a cipher-text owing to a slight change in the secret key while retaining the same plain-text. The following formula is used to calculate the KS test :

$$KS_m = \frac{1}{T}\sum_{i=1}^{T} E_K(m)_i \oplus E_{K'}(m)_i \quad (6)$$

Where K represents the original key parameter and K' the new one obtained by flipping the least significant bit (LSB) of one random byte of the original key. The KS test ideal value is 50%, indicating an important change as a reaction to the slight perturbation. In Table 5, the mean KS values for the LORMHE scheme are calculated through Eq. 6, for 10,000 iterations by varying the size of the security parameter g from 100 to 4000 bits. According to Table 5, the KS value has gotten close to the ideal one and more stable starting size of $g = 3000$ bits, with a mean value of 48.1898%. Consequently, the KS test demonstrates that the LORMHE scheme is resistant against related key attacks at g size $\geq$ 3000 bits.

5.2.2 Presence of avalanche effect

The avalanche effect is an advantageous characteristic for any encryption scheme. It describes a significant change in a system's output when only a slight modification in the input is made. A way to evaluate the avalanche effect in an encryption scheme is the Plain-text Sensitivity (PS) test. The formula used in such types of tests is given as follows :

Table 4 Security parameters size in bits

Parameter	Size in bits
a	45
l	40
r	40

Table 5 KS mean values variation according to g size in bits

Size of g (in bits)	KS test	Standard deviation
100	44.5533	0.398091
400	45.9923	0.327308
700	47.897	0.376824
1000	48.8514	0.403112
1200	47.6211	0.374563
1400	47.2786	0.476707
1600	48.2459	0.395617
1800	47.9673	0.514739
2000	48.8253	0.430068
2200	48.7609	0.380718
2400	48.0016	0.440969
2600	48.6763	0.414523
2800	48.2091	0.42613
3000	48.1898	0.463203
3200	49.052	0.474593
3400	48.9634	0.333262
3600	49.242	0.468724
3800	49.0586	0.494062
4000	49.3066	0.626656

Table 6 PS mean values variation according to g size in bits

Size of g (in bits)	PS test	Standard deviation
100	44.3227	0.189231
400	47.4448	0.21161
700	46.9989	0.187662
1000	46.9177	0.256999
1200	47.0588	0.208095
1400	47.0438	0.200046
1600	47.7795	0.186868
1800	47.6773	0.177178
2000	48.4738	0.28389
2200	48.4866	0.275553
2400	47.7509	0.169353
2600	47.3986	0.150028
2800	47.0364	0.193498
3000	48.7488	0.20586
3200	48.9309	0.222752
3400	48.4948	0.184889
3600	48.1875	0.190877
3800	48.5284	0.191281
4000	48.2725	0.19753

$$PS_m = \frac{1}{T}\sum_{i=1}^{T} E_K(m)_i \oplus E_K(m')_i \tag{7}$$

Where K is the key parameter, m is the original plain-text message and m' the new plain-text message, obtained by flipping one random bit of m. The PS is the bit-wise difference of the two differently obtained cipher-texts. To reach a high level of security, a scheme must have around 50% as PS value. The result of applying the PS test, through Eq. 7, for 10,000 iterations on the LORMHE scheme according to the variation of the g size from 100 to 4000 bits is shown in Table 6. An analysis of the presented table shows that starting at g size equal to 3000 bits, the PS value approaches 50%, with a mean value of 48.7488%. Consequently, the PS test ensures that the LORMHE scheme presents the avalanche effect for size of g $\geq$ 3000 bits.

5.2.3 Resistance against statistical attacks

In order to ensure a high level of resistance against statistical attacks, a scheme must satisfy the two following properties given below:

1. Uniformity: tested by the entropy and the distribution tests.
2. Independence: tested by the correlation and the difference tests.

Hence, in order to validate the two properties highlighted above, different security tests are implemented as follows:

1. Entropy test: the entropy value quantifies the level of uncertainty in a random variable. The entropy of a given message m is defined by the following formula :

$$H(m) = -\sum_{\substack{i=0 \\ P(x_i)\neq 0}}^{2^M-1} P(x_i) log_2(P(x_i)) \tag{8}$$

where $p(x_i)$ is the probability of occurrence of symbol x_i, and 2^M the total number of states of information. The ideal value of entropy, equal to M, is encountered when the random variable satisfies the uniformity property. In this case, the ideal value that should be encountered is 8 ($2^8 = 256$). In Table 7, the mean entropy values for the LORMHE scheme are calculated through Eq. 8, for 10,000 iterations by varying the size of the security parameter g from 100 to 4000 bits. By examining Table 7, it is obvious that the mean entropy value approaches the ideal value 8 starting from g size equals to 3000 bits with a mean value of 7.86275 and continues to grow reaching a maximal value of 7.89824. As a result, the entropy test shows that the

LORMHE scheme verifies the uniformity property for size of $g \geq 3000$ bits.

2. Difference test: the difference test quantifies the bit-level percentage difference between the plain-texts and their corresponding cipher-texts. The difference test values are obtained using the following formula:

$$D(m,c) = \frac{1}{T}\sum_{i=1}^{T} m_i \oplus c_i \tag{9}$$

where (m, c) denotes a plain-text message and its matching cipher-text c, and T is their bit length. To be considered secure, a scheme must get around 50% as result in this type of test. Such result indicates that there is no clear relationship at the bit level between cipher-texts and plain-texts. In Table 8, the mean difference values for the LORMHE scheme are calculated through Eq. 9, for 10,000 iterations by varying the g size as done in the previous tests. Table 8 shows that for g size equal to or greater than 3000 bits, the difference value approaches the ideal one, with a mean value of 49.365%. Thus, the difference test demonstrates that the LORMHE scheme verifies the independence property for size of g $\geq$ 3000 bits.

3. Correlation test : in order to be secure, an encryption scheme must present a low correlation between the plain-texts and their matching cipher-texts as there should be no clear relation that can be used to deduce one from the other. The correlation value is obtained using the following equation :

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sqrt{D(X) \times D(Y)}} \tag{10}$$

where $cov(X,Y) = E[(X - E(X))(Y - E(Y))]$, $E(X) = \frac{1}{n}\sum_{k=0}^{n} x_k$ and $D(X) = \frac{1}{n}\sum_{k=0}^{n}[x_k - E(X)]^2$. The ideal value of correlation is 0 and it is encountered when there's no linear relationship between the two variables X and Y, namely the plain-texts and their corresponding cipher-texts. In Table 9, the mean correlation values for the LORMHE scheme are calculated through Eq. 10, for 10,000 iterations by varying the size of the security parameter g from 100 to 4000 bits. By analyzing Table 9, it is clear that all the correlation values are near to zero, regardless of g size. Therefore, the LORMHE scheme ensures low correlation between its plain-texts and cipher-texts.

As conclusion, the LORMHE scheme provides resistance to statistical attacks as long as the g size is equal or greater than 3000 bits. To validate the LORMHE scheme's resilience to statistical attacks for g size $\geq$3000, distribution test is implemented while taking g size equal to 3000 as follows:

Distribution test: the distribution test is based on examining the distribution of cipher-text byte values

Table 7 Entropy mean values variation according to g size in bits

Size of g (in bits)	Entropy test	Standard deviation
100	7.50457	0.0049004
400	7.65458	0.00508553
700	7.71679	0.00537651
1000	7.757	0.00590522
1200	7.77149	0.00619845
1400	7.78281	0.00527091
1600	7.81018	0.00549545
1800	7.81645	0.0052038
2000	7.82715	0.00619482
2200	7.83685	0.00498642
2400	7.8484	0.00537628
2600	7.84697	0.00492603
2800	7.85868	0.00498562
3000	7.86275	0.00622701
3200	7.87968	0.00507043
3400	7.87995	0.00462881
3600	7.87912	0.00548008
3800	7.89071	0.00569883
4000	7.89824	0.00677862

Table 8 Difference mean values variation according to g size in bits

Size of g (in bits)	Difference test	Standard deviation
100	44.7356	0.194872
400	46.9036	0.219269
700	47.7236	0.321364
1000	47.783	0.164255
1200	46.4988	0.21326
1400	48.4669	0.233102
1600	48.7972	0.181802
1800	48.9061	0.240816
2000	49.6077	0.186999
2200	47.6616	0.222265
2400	47.7825	0.268626
2600	48.3809	0.198188
2800	48.6269	0.167038
3000	49.365	0.220737
3200	49.4472	0.167383
3400	49.9223	0.224698
3600	49.68	0.190333
3800	49.8475	0.20918
4000	50.4255	0.188776

frequency counts. Such distribution should be uniform to guarantee that the cipher-texts will never leak information on the plain-texts distribution and therefore deduce that the scheme in question is secure. Figure 3 depicts the frequency counts form of a message before and after the LORMHE method is applied to it. As a result, for a 10,000 Bytes plain-texts message length generated using a Gaussian distribution (mean: = 128, standard deviation: = 32), one can see that the frequency counts distribution of the corresponding cipher-texts (after taking the security parameter $g = 3000$ as required for achieving a secure level of security) has a peak at a byte value of around 65. Further investigations and calculations revealed that this was caused by the double precision representation on 64 bits of the cipher-texts using the IEEE 754 standard. Effectively, as all cipher-text were in more or less the same range, resulting in a more or less constant exponent value. This exponent being coded on 11 bits will result in one or two bytes in the cipher-text having a low variance, hence, a peak in that distribution. This is not a security problem as it reveals no specific information about the plain-texts, only the exponent of the cipher-texts which does not limit the search space enough to be exploited in case of possible attack. Disregarding that peak, the distribution of byte-values frequency is close to a uniform distribution. Therefore, the scheme can be seen satisfying the uniformity property.

Table 9 Correlation test mean values according to g size in bits

Size of g (in Bits)	Correlation test	Standard deviation
100	− 0.104726	0.0121138
400	0.0625372	0.0181244
700	0.0501442	0.0103746
1000	0.041004	0.0125713
1200	0.0457267	0.0117055
1400	0.0396739	0.00977117
1600	0.0357391	0.0103434
1800	0.0222172	0.0131584
2000	0.0343072	0.0105156
2200	0.0224815	0.0109613
2400	0.0288109	0.0126543
2600	0.0229197	0.0124308
2800	0.013325	0.0106455
3000	0.0197972	0.0128093
3200	0.0179161	0.0119241
3400	0.0204749	0.0128511
3600	0.0130659	0.0125206
3800	0.00953406	0.012797
4000	0.0149596	0.010473

5.3 Analysis and conclusion

To summarize, all of the security tests driven in the previous sections ensure that the newly created HE scheme LORMHE is highly secure for particular sizes of the security parameters a, g, l, and r shown in the Table 10.

6 Performance analysis

The performance analysis of the LORMHE crypto-system is presented and explained in this section. It is accomplished by comparing the execution timings of its various functions and storage overhead to those of the ElGamal

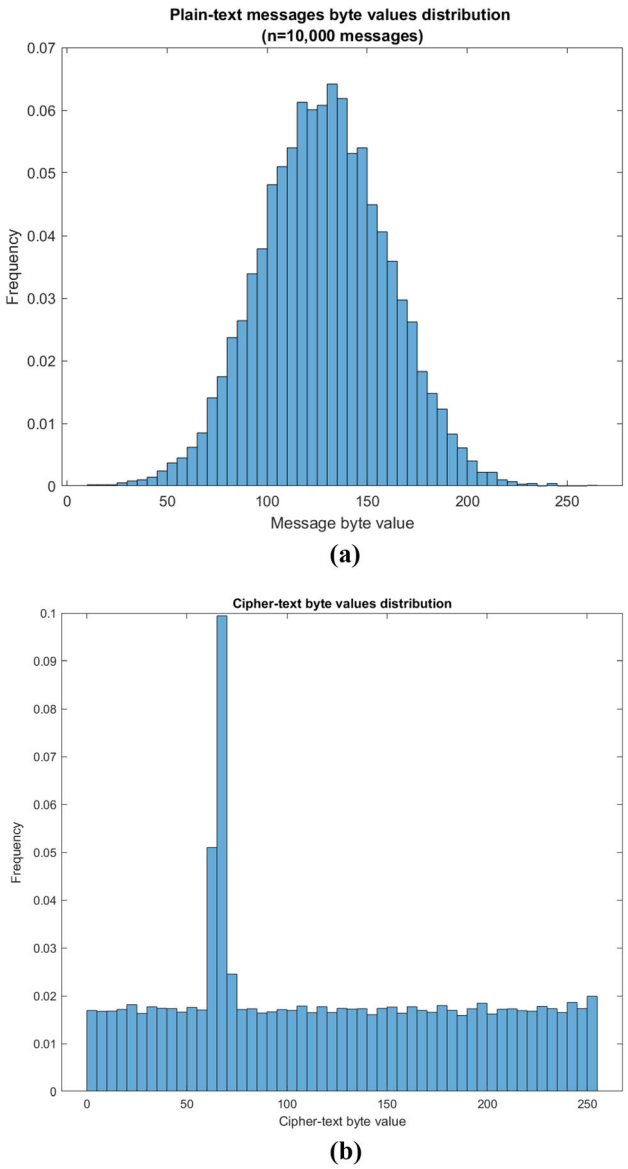


Fig. 3 Distribution test: **a** Plain-texts, **b** LORMHE cipher-texts

scheme. The two schemes are securely implemented in the latter part as follows:

- LORMHE scheme : as discussed in Sect. 5, a secure implementation of the LORMHE crypto-system is obtained when the secret key parameter size of g is equal to or greater than 3000 bits, the a size is 45 bits and the l and r sizes are 40 bits.
- ElGamal scheme : as mentioned in Sect. 3.1, a secure implementation for this scheme requires minimum 3072 bits for the key size q.

6.1 Implementation performance study and comparison

Different implementations are done using the same working environment described in Sect. 5.2 (Table 3). Various comparisons and analyses are carried out by changing the plain-text message size from 10 bytes to 100 bytes in steps of 10, and determining the mean metric of each implementation for 50 iterations as follows:

Table 10 Security parameters size in bits

Security parameter	Size in bits
a	45
g	$\geq$ 3000
l	40
r	40

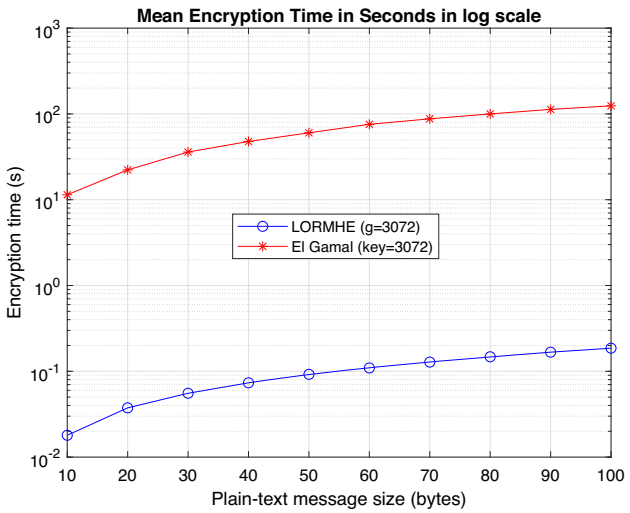


Fig. 4 Comparison of mean encryption execution time between LORMHE and ElGamal schemes

6.1.1 Cryptography functions

The encryption and decryption procedures are the two major cryptography functions for both the LORMHE and ELGamal schemes.

Figures 4 and 5 provide a comparison of the mean execution times for the encryption and decryption functions for both the LORMHE and ElGamal schemes using logarithmic scales. After examining the two figures, it is apparent that the LORMHE crypto-system is quicker than the ElGamal scheme. As a result, our new multiplicative HE scheme outperforms the ElGamal crypto-system.

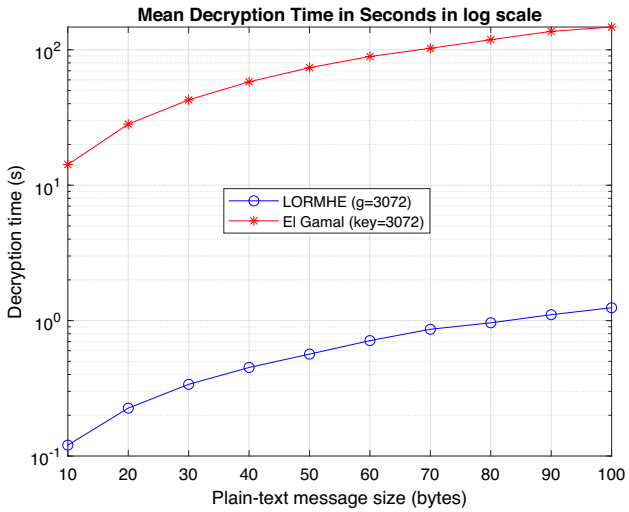


Fig. 5 Comparison of mean decryption execution time between LORMHE and ElGamal schemes

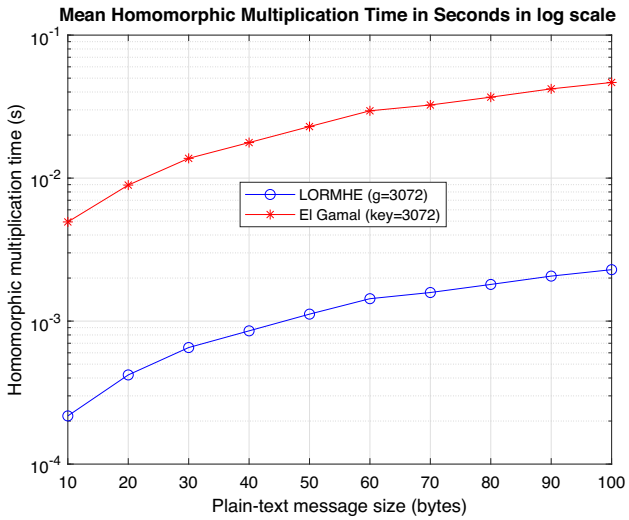


Fig. 6 Comparison of mean homomorphic multiplication execution time between LORMHE and ElGamal schemes

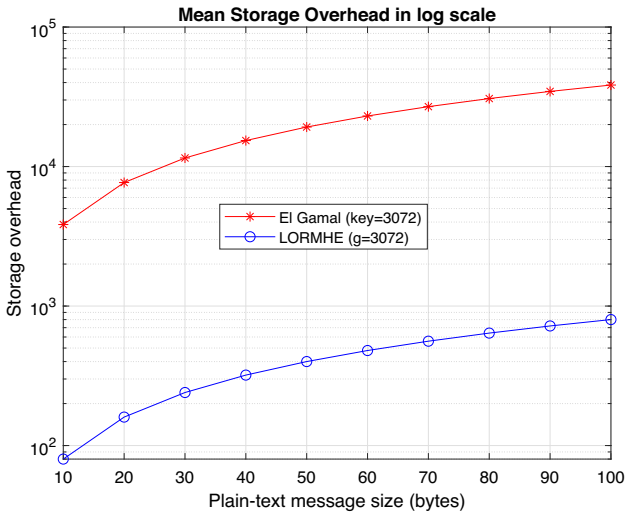


Fig. 7 Comparison of storage overhead between LORMHE and ELGamal schemes

6.1.2 Multiplication on encrypted data

Figure 6 represents the mean execution time for the homomorphic multiplication property applied to both schemes using logarithmic scale. It is clear that the LORMHE scheme is also, in this case, faster than the ELGamal crypto-system.

6.1.3 Storage overhead

The evolution of storage overhead for both encryption schemes in terms of the length of the plain-text messages is shown in the logarithmic scale in Fig. 7 below. After a deep examination of this graph, it is clear that the LORMHE scheme's storage overhead outperforms the ELGamal scheme's.

6.2 Results analysis

In this section, the LORMHE and ElGamal schemes were implemented and represented in their secure condition ((g, a, r, l) sizes equal respectively to (3072,45, 40,40) bits for LORMHE and q size equals to 3072 bits for ElGamal). Analyzing the results shows that the LORMHE scheme performing is better than the ElGamal scheme in terms of execution time for encryption, decryption and homomorphic multiplication property. Furthermore, the LORMHE was proven to be more storage efficient as it presented a lower storage overhead compared to ElGamal. In conclusion, the LORMHE scheme is a strong rival to the well-known HE ElGamal scheme. In Table 11 listed below, the minimum (for $Mess\_Leng = 10$ Bytes) and maximum (for $Mess\_Leng = 100$ Bytes) values of each implementation's mean metric are listed. $\epsilon = \frac{highest\_value}{lowest\_value}$ values are also supplied to show

Table 11 Implementations minimum and maximum mean metrics in linear scale

Scheme	LORMHE	ELGamal	ϵ	Interpretation
Minimum Encryption Execution Times in (s)	0.0179688	11.4219	$\frac{11.4219}{0.0179688} = 635.6518$	The LORMHE scheme encrypts a 10 Bytes message about 636 times faster than the ELGamal scheme
Maximum Encryption Execution Times in (s)	0.185938	124.188	$\frac{124.188}{0.185938} = 667.90005$	The LORMHE scheme encrypts a 100-Byte message about 668 times quicker than the ELGamal method
Minimum Decryption Execution Times in (s)	0.120313	14.2031	$\frac{14.2031}{0.120313} = 118.05125$	The ELGamal scheme decrypts a 10 Bytes message about 118 times slower than the LORMHE scheme
Maximum Decryption Execution Times in (s)	1.24453	147.266	$\frac{147.266}{1.24453} = 118.33061$	The ELGamal scheme decrypts a 100-Byte message about 118 times slower than the LORMHE scheme
Minimum Homomorphic Multiplicative Execution Times in (s)	0.000217188	0.0049373	$\frac{0.0049373}{0.000217188} = 22.73284$	For 10-Byte messages, multiplying homomorphically using LORMHE is about 23 times quicker than using ELGamal
Maximum Homomorphic Multiplicative Execution Times in (s)	0.00228906	0.0466464	$\frac{0.0466464}{0.00228906} = 20.377972$	Multiplying homomorphically with LORMHE takes 20 times as long as ELGamal for 100-Byte messages
Minimum Storage Overhead in (Bytes)	80	3840	$\frac{3840}{80} = 48$	The amount of storage required for a 10-Byte message encrypted by ELGamal is approximately 48 times greater than that required for a 10-Byte message encrypted by LORMHE
Maximum Storage Overhead in (Bytes)	800	38400	$\frac{38400}{800} = 48$	The amount of storage required for a 100-byte message encrypted by ELGamal is roughly 48 times larger than that required for a 100-byte message encrypted using LORMHE

numerically the efficacy of the LORMHE scheme in comparison to ELGamal's cryptosystem.

7 Conclusion and future work

The interest for HE is at the moment at its peak. It enables data analysis while keeping its content securely encrypted. One of the biggest challenge in this sector is the design of a scheme that will be at the perfect intersection of security and efficiency. In this paper, a new multiplicative HE scheme, called LORMHE, has been designed. A crypt-analysis study of the latter proved that it is secure and resistant against several types of attacks. Furthermore, performance study has revealed that the new scheme is a good substitute for the well-known ELGamal crypto-system.

Future work resides in exploiting the SAVHO and the LORMHE scheme homomorphic properties in order to design a new secure and efficient SHE scheme competent to the Paillier/ElGamal ones as well to existing and well known FHE crypto-systems like the BGV scheme [24].

Appendix: An overview of the SAVHO crypto-system

See Table 12.

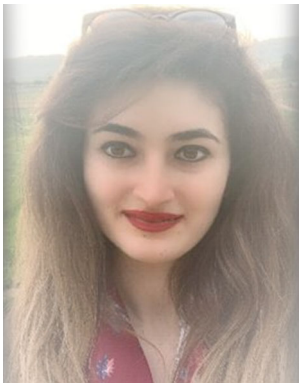
References

1. Gajbhiye, A., Shrivastva, K.M.P.: Cloud computing: need, enabling technology, architecture, advantages and challenges. In: 2014 5th International Conference-Confluence The Next Generation Information Technology Summit (Confluence), IEEE, pp. 1–7 (2014)
2. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis. Stan-ford University.crypto.stanford.edu/craig (2009)
3. Acar, A., Aksu, H., Uluagac, A.S., Conti, M.: A survey on homomorphic encryption schemes: theory and implementation. ACM Comput. Surv. (2018). https://doi.org/10.1145/3214303
4. Morris, L.: Analysis of Partially and Fully Homomorphic Encryption. Rochester Institute of Technology, pp. 1–5 (2013)
5. Lim, H.W., Tople, S., Saxena, P., Chang, E.C.: Faster secure arithmetic computation using switchable homomorphic encryption. IACR Cryptol. ePrint Arch. **2014**, 539 (2014)
6. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**, 469–472 (1985)
7. Zaraket, C., Hariss, K., Chamoun, M., Nicolas, T.: Cloud based private data analytic using secure computation over encrypted data. J. King Saud Univ. (2021). https://doi.org/10.1016/j.jksuci.2021.06.014
8. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Advances in Cryptology - EUROCRYPT 99, pp. 223–238. Springer, Heidelberg (1999)
9. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**, 120–126 (1978). https://doi.org/10.1145/359340.359342
10. Gururaja, H.S., Seetha, M., Koundinya, A.K., Shashank, A.M., Prashanth, C.A.: Comparative study and performance analysis of encryption in RSA, ECC and Goldwasser-Micali cryptosystems. Int. J. Appl. Innov. Eng. Manag. **3**, 111–118 (2014)

Table 12 SAVHO scheme

Encryption parameters	1. The dimension of the cipher-text, denoted by n, is considered as a security parameter 2. The secret key is given by a $n \times n$ random invertible matrix denoted by $P = (p_{i,j} \in \mathbb{Z})$
Encryption function	For a given message $m \in \mathbb{Z}$, decomposed into $S = [S_0, \ldots, S_{n-1}] \in \mathbb{Z}^n$ such that $\sum_{i=0}^{n-1} S_i = m$, and two random vectors R and $N \in \mathbb{Z}^n$ generated such that $(N_i + S_i) > 0$ and $(N_i + \frac{S_i}{2}) > 0$ $(i \in [0, n-1])$ the cipher-text vector elements is given by: $c_i = \sum_{j=0}^{n-1} p_{ij}(N_j S_j + R_j)$ where $0 \le i \le n-1$
Decryption Function	Retrieving m is based on applying the following formulas: 1. $S_i = \sqrt{2(\sum_{j=0}^{n-1} q_{ij} c_j - R_i) + N_i^2 + S_i^2} - N_i$ 2. $m = \sum_{i=0}^{n-1} S_i$ Remark: $0 \le i \le n-1$ and $Q = [q_{ij} \in \mathbb{Z}]$ represents the inverse matrix of the secret key P.
Homomorphic addition and average	As demonstrated in [7], SAVHO scheme validates the following formulas: For L cipher-texts $c^i = [c_j^i]$ $(1 \le i \le L$ and $0 \le j \le n-1)$ $c_{add} = \sum_{i=1}^{i=L} c^i$ and $c_{Average} = \frac{\sum_{i=1}^{i=L} c^i}{L}$
Crypt-analysis	Theoretical Security: SAVHO scheme is resistant against known plain-text/cipher-text attacks. Security by Implementation: As given in [7], SAVHO recommends $n \ge 400$ to be considered secure.

11. Zaraket, C., Chamoun, M., Nicolas, T.: Calculating the average using Paillier's cryptosystem. In: BDCSIntell, CEUR-WS, pp. 113–117 (2019)
12. Hariss, K., Chamoun, M., Samhat, A.E.: On DGHV and BGV fully homomorphic encryption schemes. In: 2017 1st Cyber Security in Networking Conference (CSNet), pp. 1–9 (2017). https://doi.org/10.1109/CSNET.2017.8242007
13. Zhang, M., Romero, S.: Design and implementation of an e-voting system based on Paillier encryption. In: Arai, K., Kapoor, S., Bhatia, R. (eds) Advances in Information and Communication. FICC 2020. Advances in Intelligent Systems and Computing, vol 1129. Springer, Cham. https://doi.org/10.1007/978-3-030-39445-5-59 (2020)
14. Ding, Y., Tian, L., Han, B., Wang, H., Wang, Y., Zheng, J. X. (2019). Achieving privacy-preserving iris identification via El Gamal. Comput. Mater. Continua **61**(2), 727–738 (2019)
15. Jain, M., Singh, P., Raman, B.: SHELBRS: location based recommendation services using switchable homomorphic encryption (2021). arXiv preprint arXiv:2105.14512
16. Sarton, G.: A Greek-English Lexicon. Henry George Liddell, Robert Scott, Henry Stuart Jones, Roderick McKenzie. IACR Cryptol. ePrint Arch. (1926)
17. Pierce, R.S.: The associative algebra. In: Associative Algebras, pp. 1–20. Springer (1982)
18. Simmons, G.J.: Symmetric and asymmetric encryption. ACM Comput. Surv. **11**(4), 305–330 (1979)
19. Parms, J.: Symmetric vs. Asymmetric Encryption-What are differences?. SSL2BUY Wiki-Get Solution for SSL Certificate Queries (2020)
20. Hariss, K., Noura, H.: Towards a fully homomorphic symmetric cipher scheme resistant to plain-text/cipher-text attacks. Multimed. Tools Appl. **81**, 14403–14449 (2022). https://doi.org/10.1007/s11042-022-12043-7
21. Smirnoff, P., Turner, D.M.: Symmetric Key Encryption-why, where and how it's used in banking. Accessed 20 Nov 2019
22. Shukla, K.N.: The linear indeterminate equation-a brief historical account. Hist. Math. **15**, 83–94 (2015)
23. Noura, H., Chehab, A., Sleem, L., Noura, M., Couturier, R., Mansour, M.M.: One round cipher algorithm for multimedia IoT devices. Multimed. Tools Appl. **77**, 18383–18413 (2018). https://doi.org/10.1007/s11042-018-5660-y
24. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, Association for Computing Machinery, New York, NY, USA, pp. 309–325 (2012). https://doi.org/10.1145/2090236.2090262



Christiana Zaraket is currently an instructor at Saint Joseph University of Beirut. She received her Ph.D degree in Computer Engineering and Telecommunications at Saint Joseph University in 2021. Her main research interest is in Homomorphic Encryption (HE) and Applied Mathematics. Currently, she is focusing on designing new secure and performant Fully Homomorphic Encryption (FHE) schemes for Cloud Services.

Khalil Hariss is currently an instructor at Université Saint-Joseph in Beirut (ESIB, Faculty of Engineering). He received his Ph.D degree in Cryptography at both Université Saint-Joseph and Université Libanaise. His main research interest is in Homomorphic Encryption (HE) and Blockchain technology. Currently, he is focusing on designing new HE schemes that provide simultaneously the required efficiency and level of security, Implementing HE in real-world applications, and finally integrating blockchain technology with HE in order to achieve a fully secure system especially for cloud environments.

Sandro Ephrem graduated with an engineering degree in Software Engineering from Ecole Supérieure d'Ingénieurs de Beyrouth (ESIB), Saint Joseph University of Beirut in 2021. He has worked as a data scientist with multiple startups mainly on machine learning in fraud detection, health, and finance. His mains interests and research fields are deep learning, cryptography, and finance.

Maroun Chamoun is a professor at the Faculty of Engineering of Saint Joseph University (USJ) in Beirut. He holds a degree in Computer Engineering from Saint Joseph University and a Master in Intensive Calculus from the joint program between the Lebanese University (UL) and Saint- Joseph University of Beirut. He holds a PhD in "Computer Science and Networks" from Telecom ParisTech in Paris. He teaches "ethical hacking", "Malware Analysis", "operating systems", and "language theory and compilers". He is a member of the research center CIMTI (Center for Computer Science, Modeling and Information Technologies) where he was the director between 1998 and 2015. His research interests include Cybersecurity mainly in Virology and Threat Detection, Cryptography mainly Homomorphic Schemes, Operating Systems mainly in scheduling and system protection, Compilers and Computer Languages mainly in correctness proof of compilers. He has more than 20 journal and conference publications in the domain of Cybersecurity. Beside Cybersecurity, he is interested in detection problems using Machine and Deep Learning as well as Natural Language Processing (NLP) such as Arabic cyberbullying detection and detection of information related to crime from Arabic texts.

Tony Nicolas is an Associate professor at the Faculty of Engineering of Saint Joseph University (USJ) in Beirut. He holds a Master in Parallel Computing from Joseph Fourier University (France). He holds a PhD in "Applied Mathematics" from Rouen University (France). He is a member of the research center CIMTI (Center for Computer Science, Modeling and Information Technologies). Recently, his research interests include Cryptography mainly Homomorphic Schemes. He has more than 10 journal and conference publications in the domain of Acoustic Control and Parallel Computing.