An Efficient Solution Towards Secure Homomorphic Symmetric Encryption Algorithms

Khalil Hariss[1,2,*], *Hassan* Noura[3,**], *Abed Ellatif* Samhat[1,***], and *Maroun* Chamoun[2,****]

[1]Lebanese University, Faculty of Engineering-CRSI, Hadath, Lebanon
[2]Saint Joseph University, ESIB-CIMTI, Mar Roukoz, Lebanon
[3]American University of Beirut, Department of Electrical and Computer Engineering, Hamra, Lebanon

Abstract. In this paper, we consider Homomorphic Encryption (HE) to process over encrypted data in order to achieve user privacy. We present a framework solution to provide a high level of security for the symmetric HE algorithms. The proposed solution introduces a dynamic structure and dynamic diffusion's primitives that enhance existing symmetric HE algorithms and overcome their weaknesses. We apply this solution to a well known symmetric homomorphic approach, the PORE (Polynomial Operation for Randomization and Encryption) approach. The security analysis of the proposed solution shows that it ensures a high level of security without performance degradation. It is also evaluated against different attacks. This leads to secure and efficient HE Algorithms for practical implementations.

1 Introduction

After the significant changes in modern systems, providing an implementation with a high level of security becomes a big challenge. In this paper, we ensure users privacy by securing data processing. This can be attained by using a new kind of encryption called HE that permits third parties to process over encrypted data without the need of decrypting cipher-texts. HE is required in several real world modern applications such as Cloud Computing [1], e-Vote applications [2], Medical Applications [3], etc. As an implementation of HE in modern real world applications is given in Figure 1, an illustration of Cloud querying using HE is given in Figure 2.

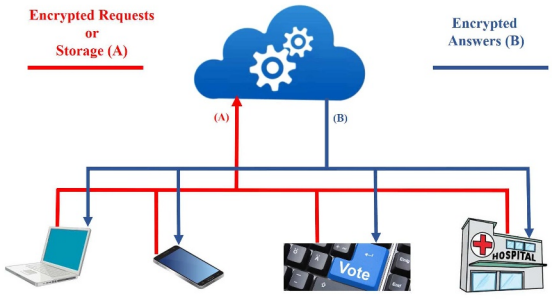


Figure 1: HE in a Cloud Based Scenario

A crypto-system that satisfies both addition and multiplication is known as Fully Homomorphic Encryption (FHE):

1. Addition

$$[E_K(x) + E_K(y)] \quad modN = [E_K(x + y, modN)] \quad modN \tag{1}$$

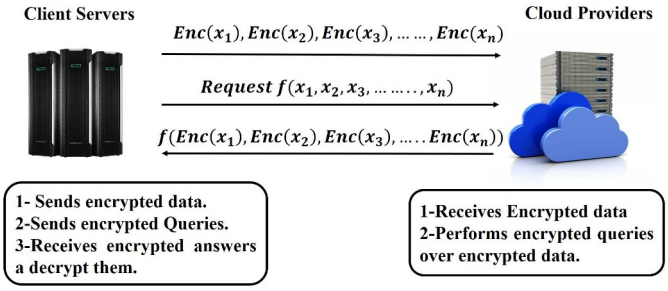


Figure 2: Cloud Querying using HE

2. Multiplication

$$\begin{aligned}&[E_K(x) \times E_K(y) modN] \quad modN \\ &= [E_K(x \times y, modN)] \quad modN\end{aligned} \tag{2}$$

where $x, y \in Z_N$, E is the encryption function and K is a secret key. Thus, the main idea of FHE is that any specified party (could be untrusted) may compute $E(x + y)$ and $E(x \times y)$ from $E(x)$ and $E(y)$ without knowing any information about x and y. Different FHE algorithms are given in the literature [4] and decomposed into asymmetric algorithms such as RSA (Rivest, Shamir, Adelman) [5], DGHV (Dijk, Gentry, Halevi, Vaikuntanathan) [6] , BGV (Brakerski, Gentry, Vaikuntanathan) [7], etc and symmetric schemes such as NOHE (Not Operation for Homomorphic Encryption) [1], DF (Domingo Ferrer) [8], etc. The asymmetric ones suffer from computational complexity and high storage overhead, while the symmetric ones suffer from weak immunity against attacks and especially chosen and known plain-texts ones, thus designing and implementing a FHE practical for real world application is a real challenge. In our previous work [9], the MORE ap-

*e-mail: khalil.hariss@net.usj.edu.lb
**e-mail: hn49@aub.edu.lb
***e-mail: Samhat@ul.edu.lb
****e-mail: maroun.chamoun@usj.edu.lb

proach was described and a dynamic platform (dynamic structure and dynamic diffusions primitives) is applied over it, giving birth to a new homomorphic encryption algorithm called the Enhanced MORE. Security analysis and performance of the Enhanced MORE in [9] has shown a high resistance to several attacks. In this paper, We extend the solution shown in [9] to PORE approach [10] to provide a new HE algorithm candidate that is based on different mathematical rules compared to MORE. In addition, the proposed solution called "Enhanced PORE" can overcome the different weaknesses of original PORE. Furthermore, a security analysis demonstrates that the proposed solution ensures a high level of security without performance degradation as the "Enhanced MORE". This leads to providing a second secure and efficient HE algorithm candidate for practical application.

The rest is organized as follows, Section 2 explains PORE approach. Section 3 describes the dynamic implementation listed in [9] over the PORE approach. The security analysis and performance evaluation of the resultant algorithm (Enhanced PORE) are given in Section 4 with a comparison to PORE, MORE, Enhanced MORE, in term of execution time and storage overhead. Conclusions are presented in Section 5.

2 MORE And PORE Approaches

In this section, we give a brief explanation of the MORE Approach and a detailed one for the PORE approach.

Table 1: MORE Approach

Secret Key	Invertible matrix K in Z_N
Public Parameters	No Public Parameters
Plain-text Space	set of x in Z_N.
Encryption Process	$Enc(x)= K \begin{bmatrix} x & 0 \\ 0 & r \end{bmatrix} K^{-1}$, r random
Cipher-text Space	set of matrices $C = [c_{ij}]$, $c_{ij} \in Z_N$
Decryption Process	$\begin{bmatrix} x & 0 \\ 0 & r \end{bmatrix} = K^{-1} Enc(x) K$.
Fully Homomorphism	verified by a matrix calculations

2.1 MORE Approach

The MORE approach is investigated in [9, 10], based on invertible matrix equation, and summarized in Tab.1.

2.2 PORE Approach

PORE Approach stands for Polynomial Operations for Randomization and Encryption [10]. It is a FHE algorithm that satisfies both addition and multiplication properties. The proposed algorithm is based on the following operations:

2.2.1 Encryption Parameters

The symmetric key (v_1, v_2) is selected from secret large integers v_1 and v_2 $mod(N)$. Using this key, the public polynomial $PP(v)$ of variable v is computed to calculate the public parameters b and c:

$$\begin{aligned} PP(v) = (v - v_1)(v - v_2) = v^2 - (v_1 + v_2)v + v_1 v_2 = v^2 + bv + c. \\ where, b = -(v_1 + v_2) \quad mod(N) \\ and, c = (v_1 v_2) \quad mod(N) \end{aligned} \tag{3}$$

b and c are known by the third party to perform homomorphic computations over encrypted data.

2.2.2 Encryption Process

The encryption of a plain-text x_i in Z_N is done as follows:
- The sender picks a large random integer $r_i mod(N)$ for a given plain-text x_i and should solve a linear system with two unknowns a_i and d_i:

$$\begin{aligned} a_i v_1 + d_i = x_i \\ a_i v_2 + d_i = r_i \end{aligned} \tag{4}$$

- The encryption of x_i is $E(x_i)$=(a_i, d_i) and calculated as follow:

$$\begin{aligned} a_i = (\frac{x_i - r_i}{v_1 - v_2}) mod(N) \\ d_i = (\frac{r_i v_1 - x_i v_2}{v_1 - v_2}) mod(N). \end{aligned} \tag{5}$$

As we know the division in a ring structure is not supported, we will develop during this work a new algorithm that makes the encryption possible.

2.2.3 Decryption Process

Having the secret key (a_i, d_i), the receiver can recover the plain-text by applying this decryption process:

$$x_i = a_i v_1 + d_i. \tag{6}$$

2.2.4 Homomorphic Properties

Given two different cipher-texts $E(x_1) = (a_1, d_1)$ and $E(x_2) = (a_2, d_2)$, Homomorphic properties are discussed as follow:

1. **Addition**

$$\begin{aligned} E(x_1) + E(x_2) = (a_1 + a_2, d_1 + d_2) = E(x_1 + x_2) \\ because, a_1 v_1 + d_1 + a_2 v_1 + d_2 \\ = (a_1 + a_2)v_1 + d_1 + d_2. \end{aligned} \tag{7}$$

2. **Multiplication**
We recall the two public parameters
$b = -(v_1 + v_2) mod(N)$ and $c = (v_1 v_2) mod(N)$
known by the third party introduced in (3).
Departing from a cipher-text (A, D) =
$((a_1 + d_1))(a_2 + d_2) - a_1 a_2 (1 + b) - d_1 d_2, d_1 d_2 - a_1 a_2 c)$.
By applying the PORE decryption process listed in Equation (6) on (A, D) we obtain:
$Av_1 + D = ((a_1 + d_1)(a_2 + d_2) - a_1 a_2 (1 + b) - d_1 d_2)v_1 +$
$d_1 d_2 - a_1 a_2 c = a_1 a_2 {v_1}^2 + (a_1 d_2 + d_1 a_2)v_1 + d_1 d_2$
$= (a_1 v_1 + d_1)(a_2 v_1 + d_2) = x_1 x_2 mod(N)$.
We can conclude that the PORE homomorphic Multiplication is given by: $E(x_1) \times E(x_2) = (A, D) =$

$$\begin{aligned} ((a_1 + d_1))(a_2 + d_2) - a_1 a_2 (1 + b) - d_1 d_2, \\ d_1 d_2 - a_1 a_2 c). \end{aligned} \tag{8}$$

2.2.5 PORE Security Analysis

PORE suffers from vulnerabilities because it is built on linear transformations and it has two public parameters $b = -(v_1 + v_2) mod(N)$ and $c = (v_1 \times v_2) mod(N)$.

3 Dynamic Implementation

A dynamic implementation similar to the one applied in [9] over MORE approach, is used in this paper to enhance the PORE approach. A detailed explanation of this framework is explained in the next subsections.

3.1 Enhanced PORE Implementation

The proposed solution employs a dynamic structure in addition to dynamic diffusion primitives. Figure 3 shows the different steps of Enhanced PORE algorithm that are explained below.

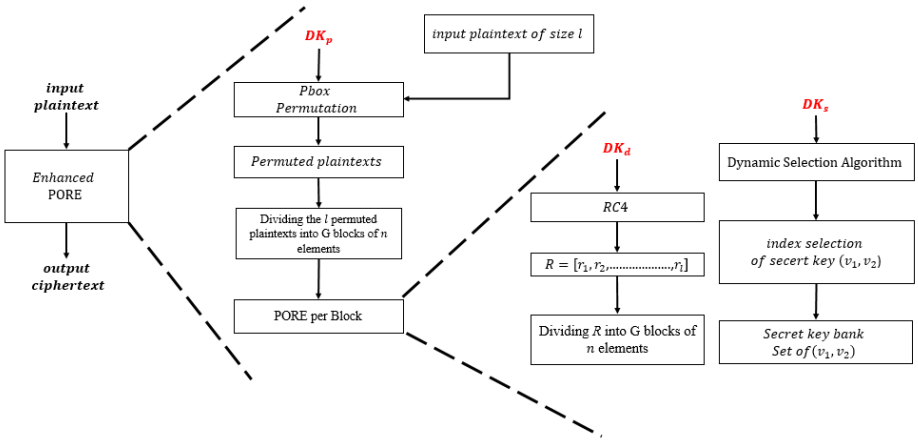


Figure 3: Enhanced PORE Flow Chart

3.1.1 Dynamic Key Generation

The two end hosts should agree on two secret parameters: a secret key and an initial vector (IV). Using a secure hash algorithm, a Dynamic Key (DK) of 64 bytes is created. Three different keys are picked and used to form three different cipher layers as follows (Figure 3) :

1. DK_p: Permutation DK (23 bytes).
2. DK_d: Diffusion DK (16 bytes).
3. DK_s: Selection DK (23 bytes).

3.1.2 Permutation Box

Using DK_p, a Permutation Box (PBox) is generated and applied over the input plain-text. PBox creation is done similar to [11], where a key dependent permutation technique is employed that preserves homomorphic properties. The homomorphic behavior is shown by using a PBox called π of dimension N defined by: $\pi=[p_i]_{1\leq i\leq N}$.
Two plain-texts X and Y of dimension N are given: $X = [x_i]_{1\leq i\leq N}$ and $Y = [y_i]_{1\leq i\leq N}$.
After permutation $\pi(X) = [x_{p_i}]_{1\leq i\leq N}$ and $\pi(Y) = [y_{p_i}]_{1\leq i\leq N}$.
Suppose that $\odot$ is a law defined over the plain-texts by:
$X\odot Y = [x_i]_{1\leq i\leq N}\odot[y_i]_{1\leq i\leq N} = [x_i\odot y_i]_{1\leq i\leq N} = [z_i]_{1\leq i\leq N} = Z$.
$\pi(X \odot Y) = \pi(Z) = [z_{p_i}]_{1\leq i\leq N} = [x_{p_i} \odot y_{p_i}]_{1\leq i\leq N}$.
And $\pi(X) \odot \pi(Y) = [x_{p_i}]_{1\leq i\leq N} \odot [y_{p_i}]_{1\leq i\leq N}$
$= [x_{p_i} \odot y_{p_i}]_{1\leq i\leq N}$.
Since $\pi(X\odot Y)=\pi(X)\odot\pi(Y)$, We can deduce the homomorphic behavior of π.

3.1.3 Dynamic Block Encryption

As shown in Figure 3, the permuted plain-texts are decomposed into a G blocks, where $G = \left\lceil \frac{l}{n} \right\rceil$, and n is the block size. Each block of dimension n is encrypted with PORE approach using an encryption key (v_1, v_2) chosen dynamically from a secret key bank based on a dynamic selection algorithm.

3.1.4 Secret Pseudo-Random Sequence Generation

Based on equation (4), the encryption of a plain-text x_i requires a random integer r_i. Thus, we should generate blocks of pseudo-random integers for the encryption of plain-texts blocks. DK_d can be employed as a seed for a cipher algorithm (like $RC4$) to build a secret sequence R of l random integers. Similar to the permuted plain-texts, the sequence R is decomposed into G blocks where $G = \left\lceil \frac{l}{n} \right\rceil$.
To perform encryption using Enhanced PORE each block of random integers is used to encrypt a block of plain-texts as shown in Figure 3.

3.1.5 Secret Key Bank Generation

As stated above, a key is chosen dynamically from a secret key bank during the dynamic encryption of each block. Based on equation (5), any key (v_1, v_2) is built such that $(v_1 - v_2)$ is invertible by the multiplicative law in Z_N. We propose a generation algorithm to create a shared secret key bank having the following form $(v_1^k, v_2^k, (v_1^k - v_2^k)^{-1})$, where $1 \leq k \leq H$. H is the secret bank length ($H < G$) .
The two algorithms 1 and 2 represent the generation of k^{th} secret key of the bank (v_1^k, v_2^k). The generation of H keys requires the repetition of these two algorithms H iterations. The two end hosts start by generating a secret sequence s of length αH. The secret parameter ${v_1}^k$ is generated based on the pseudo code of algorithm 1.

Algorithm 1 v_1^k Generation

```
procedure (v1^k, s)=v1^k_GENERATION(DK_d,k,α,H)
    s ← RC4(DK_d, α, H)
    i ← k
    while s_i = 0 do
        i = i + 1
    end while
        v1^k = s_i
return (v1^k, s)
end procedure
```

Once ${v_1}^k$ is chosen as in algorithms 1, ${v_2}^k$ is chosen such that $({v_1}^k - {v_2}^k)$ is invertible by the multiplicative law. The invertible $({v_1}^k - {v_2}^k)$ generation is based on the pseudo code of algorithm 2. In the Pseudo-Code of algorithm 2, after generating the secret sequence s listed in algorithm 1, the two end hosts pick from it a parameter $u(k, j)$ to build an invertible multiplicative element $({v_1}^k - {v_2}^k)$ in Z_N. Now $({v_1}^k - {v_2}^k)$ and ${v_1}^k$ are ready, ${v_2}^k$ is simply calculated.
In general the number of invertible multiplicative elements in Z_N is limited. This leads to a limited number of encryption keys (v_1^k, v_2^k), but our key generation dynamic approach strengthens this implementation, because in each session the H secret keys are generated dynamically in different order based on DK.

3.1.6 Dynamic Key Selection algorithm

The DK selection is given in Figure 4 and can be summarized by:
Based on DK_s and a stream cipher algorithm like $RC4$, another permutation box $\Delta = \{\delta_i, i = 1, 2, 3, ..., G\}$ is generated. (G is the number of blocks, and $\delta_i \in \{1, 2, 3, ..., H\}$).

Algorithm 2 $(v_1^k - v_2^k)$ Generation

procedure $(v_1^k - v_2^k)$=$(v_1^k - v_2^k)$_*Generation*(*k*,*s*)
  $(v_1^k - v_2^k)^{-1} \leftarrow 0$
  $j \leftarrow 0$
  while $(v_1^k - v_2^k)^{-1} = 0$ **do**
    $u(k, j) \leftarrow s_{3\times(k-1)+1+j}$
    $product \leftarrow 1 + N \times u$
    $i \leftarrow 2$
    while $i \leq N - 1$ **do**
      if $mod(product, i) = 0$ **then**
        $(v_1^k - v_2^k)^{-1} \leftarrow i$ **break**
      else
        $i = i + 1$
      end if
    end while
    $j = j + 1$
  end while
  $(v_1^k - v_2^k) \leftarrow mod(\frac{product}{(v_1^k - v_2^k)^{-1}}, N)$
return $(v_1^k - v_2^k)$
end procedure

For the k^{th} block, the index δ_k is chosen from Δ and based on it, a secret key $(v_1{}^{\delta_k}, v_2{}^{\delta_k})$ from the secret key bank is chosen.

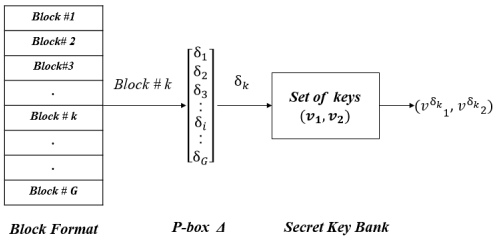


Figure 4: DK Selection Algorithm

3.1.7 Decryption Process

As any symmetric scheme, the decryption process is the inverse of the encryption. Having DK and IV, all secret parameters can be generated. The decryption process is based on the following steps:

1. **First Step** Based on DK_s and DK_d, the receiving end host can pick for each block number k the key $(v_1{}^k, v_2{}^k)$, such that $1 \leq k \leq G$.
2. **Second Step** After retrieving the decryption key for each block, the receiving end host applies the equation (6), to perform the decryption process.
3. **Third Step** The receiving end host generates the inverse secret permutation vector π^{-1} by using DK_p and the following transformation:

$$\pi^{-1}[\pi[i]] = i, where \quad i \in \{1, 2, 3, ..., l\} \qquad (9)$$

4 Security Analysis and Performances

To evaluate the performance of the Enhanced PORE, the same security analysis given in [9] is implemented. A simulation under Matlab is done where a set of plain-texts in Z_{256} is taken . A comparison between the Enhanced PORE, PORE, MORE and Enhanced MORE is done in terms of execution time and performances. In the upcoming results, the Enhanced PORE cipher-text is the couple (a, d) where a is known as first cipher and d the second one.

4.1 Resistance Against Statistical Attacks

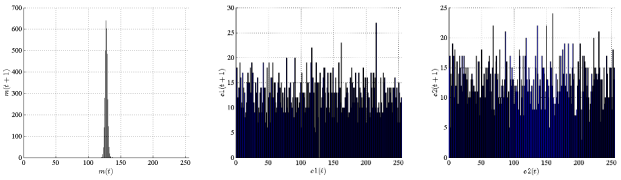
To resist against statistical attacks, the proposed scheme should ensure the Uniformity and Independence criterion.

4.1.1 Uniformity Property

The Uniformity criterion can be examined by applying the two different tests given in Tab. 2.

Table 2: Uniformity Property

Required Test	Procedure	Purpose
Distribution Test	Plotting distribution	Uniform distribution
Entropy Test	$\sum_{i=0}^{2^M-1} p(m_i) log_2 \frac{1}{p(m_i)}$	Uncertainty



(a) (b) (c)

Figure 5: Distribution Test: (a)- Original message, (b)-Enh PORE Cipher 1 (c)-Enh PORE Cipher2

Distribution Test: a Gaussian plain-text distribution with a mean value equal to 128 and standard deviation equal to 16 is taken in Figure 5 (a), and the distribution of the obtained set of cipher-texts is illustrated in Figure 5 (b), (c). Comparing the different results of Figure 5, the cipher-text distribution after applying the encryption process is close to uniform distribution. As a conclusion the Enhanced PORE can strongly resist against any statistical attack.

Entropy Test: The entropy of a source message m is given in Tab. 2, where $p(m_i)$ represents the probability of occurrence of symbol m_i and 2^M is the total states of information source. A truly random source entropy is equal to M. In our implementation the cipher values are in Z_{256}, the ideal value of the entropy should be equal to 8 ($2^8 = 256$). The entropy values for 10000 cipher-texts or iterations has shown that the mean values are close to 8 for both ciphers ($mean1 = 7.936$ and $mean2 = 7.9321$) with a low standard deviations ($Std1 = 0.005153$ and $Std2 = 0.0116$). The resultant cipher-texts of our scheme are considered a truly random source.

4.1.2 Independence Property

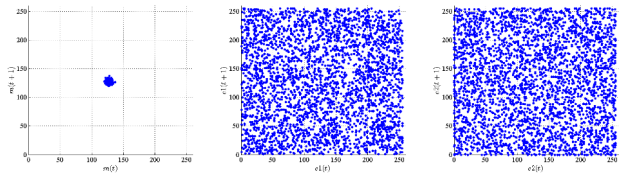
To examine Independence Property, we need to validate the three different tests given in Tab. 3.

Recurrence Test: Figure 6 shows the correlation between $x_i(t)$ and $x_i(t+1)$ for the original and the encrypted data respectively. Figure 6 (a) represents the correlation among a set of plain-texts with mean value equal to 128 and a low standard deviation equal to 16. Figure 6 (b) and (c) shows the variation between $x_i(t)$ and $x_i(t + 1)$ for the Enhanced PORE. The cipher-text space presents a high level of randomness, and no clear pattern is shown after the encryption process.

Correlation Test: Correlation is calculated as given in Tab. 3, where $E(x)$ represents the Mean value, and $D(x)$ represents the Variation. The correlation values for our algorithm for 10000 iterations have given mean values close to zero ($mean1 = 0.0002386$ and $mean2 = 8.854 \times 10^{-5}$)

Table 3: Independence Property

Required Test	Procedure	Purpose
Recurrence Test	Variation between a data $x_i = x_{(i,1)}, ...x_{(i,m)}$ and its delayed version $x_i(t) = x_{(i,t)}, ...x_{(i,mt)}$	Randomness
Correlation Test	$\rho_{x,y} = \frac{cov(x,y)}{\sqrt{D(x) \times D(y)}}$, where $cov(x,y) = E[\{x - E(x)\}\{y - E(y)\}]$	Independence
Difference Test	Difference at the bit level between cipher-texts and plain-texts	Independence



(a) (b) (c)

Figure 6: Recurence Test: (a)- Original message, (b)-Enh PORE Cipher 1 (c)-Enh Pore Cipher 2

with low standard deviations($Std1 = 0.01779$ and $Std2 = 0.01756$), which means that the cipher-texts of the proposed algorithm do not reveal any information about the plain-texts.

Difference Test: To evaluate this difference, in our simulation we calculate the difference at the bit level between 10000 cipher-texts and plain-texts. The different simulations have shown mean values close to 50 ($Mean1 = 50.004$ and $Mean2 = 50.0005$) with low standard deviations ($Std1 = 0.312$ and $Std2 = 0.3177$). The Enhanced PORE algorithm satisfies the difference property and presents a high level of independency between the cipher-texts and the plain-texts.

4.2 Resistance Against Several Kinds of Key Attacks

The main purpose of this section is to show that our encryption algorithm can resist against several types of key attacks.

4.2.1 Weak Keys

In this dynamic implementation, the proposed key derivation function produces a set of dynamic sub-keys with a high degree of randomness. Indeed, Different cipher layers such as the permutation layer and the diffusion layer are related to the dynamic key to achieve the desirable cryptographic performance. Suppose, for example, a weakness exists in any dynamic key, it will not alter the previous and the next processed data. As a conclusion, the used dynamic approach provides a good resistance degree against the weak keys.

4.2.2 Key Sensitivity

The Key Sensitivity (KS) refers to a big change in the cipher-text due to a slight change in the encryption key. Let all the elements of K'_w be equal to those of K_w, except a random Least Significant Bit (LSB) of a random byte, and T_b being the length of the original and cipher packets (in bits), the sensitivity is calculated as follows:

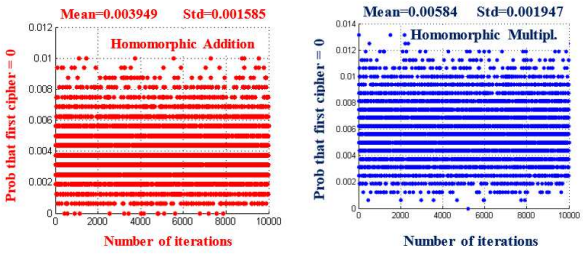
$$KS_w = \frac{\sum_{k=1}^{T} E_{K_w} \oplus E_{K'_w}}{T} \times 100\%, \quad w = 1, 2, \ldots, 1000. \tag{10}$$

A good cryptosystem should give a key sensitivity close to 50. KS test is done for 10000 iterations; the mean values are also close to 50 ($mean1 = 50.0025$ and $mean2 = 49.999$) with a low standard deviations ($Std1 = 0.3105$ and $Std2 = 0.3151$). As a conclusion, the resultant algorithm provides a high resistance against related key attacks.

4.3 Enhanced PORE Zero Homomorphic Test

The PORE encryption equations given in (5) impose that during the random pick of r_i, it should always be different from the plain-text x_i; otherwise the encryption is useless because the second cipher d_i will be equal to x_i (i.e first cipher a_i should always be different from zero). A constraint is added to solve this limitation and indicates that r_i is always different than a_i. The remaining problem can simply be discovered by focusing on homomorphic operations of equations (7) and (8). As an example in the homomorphic addition listed in (7), it is sure that $a_1 \neq 0$ and $a_2 \neq 0$ but $a_1 + a_2 \neq 0$ is not guaranteed. The same problem exists in equation 8. To evaluate the effect of this vulnerability in Enhanced PORE, this test is proposed: two different vectors of plain-texts are taken, then encrypted using homomorphic Enhanced PORE. The two resultant cipher-texts are added and multiplied using homomorphic operations, and the probability that the first cipher is equal to zero is calculated. The test is done for 10000 iterations.

The analysis of Figure 7, shows that the probability of the first cipher being equal to zero is negligible with a low standard deviations for homomorphic addition and multiplication. The investigated problem does not decrease the security performance of the Enhanced PORE since its occurrence is rare.



(a) (b)

Figure 7: Zero PORE Homomorphic Test: (a) Zero test for addition (b) Zero test for multiplication

4.4 Performance Analysis

The performance of any crypto-system resides in its low storage overhead and latency. The performance evaluation of the two resultant algorithms is studied in the next subsections.

4.4.1 Storage Overhead

The new dynamic approach did not affect the storage overhead of the resultant algorithm (i.e the storage overhead of the PORE its Enhanced version is the same). The encryption of m bytes of plain-text using PORE or Enhanced PORE gives $2 \times m$ bytes of cipher-text.

4.4.2 Execution Time

Different implementations are done under MATLAB using Toshiba Laptop having the following specifications: Processor Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz, 2301MHz, 2 Core(s), 4 Logical Processor(s). In this work, we implemented the Enhanced PORE and the PORE at the block level (i.e for each plain-text block different encryption key is chosen during the encryption form the secret key bank), the same implementation can be done for the PORE and the Enhanced PORE at the byte level (i.e for each byte or plain-text in Z_{256} a different secret key is chosen from the secret key bank) which improves the security performances. The execution time of MORE and Enhanced MORE are taken from [9]. The execution time is studied by varying plaintexts vector size from 800 bytes to 8000 bytes with a step equal to 800, and measuring for each plaintext size the mean execution time for 10000 iterations as in [9]. The execution time is shown in Figure 8, where the PORE Approach at the block level is taking the lowest execution time, then comes the Enhanced PORE at the block level and its execution time is still much smaller than the PORE at the byte level, MORE and Enhanced MORE. The PORE Approach at the byte level is taking the highest execution time.

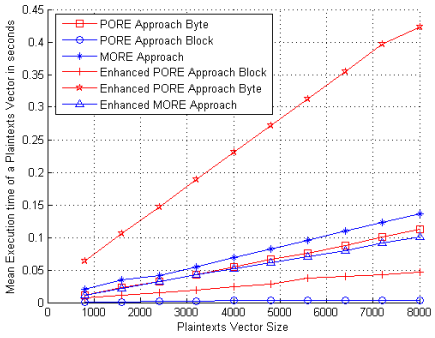


Figure 8: Execution Time

4.5 Enhanced MORE and Enhanced PORE comparison

The Enhanced MORE can be applied in a non trusted cloud scenario because it does not have any public parameters, while the Enhanced PORE should be applied in a trusted cloud scenario due to public parameters b and c. In term of storage overhead, the cipher-text size of the Enhanced MORE is related to the matrix dimension as explained in [9] (Given a plain-text of m bytes and a matrix of dimension $n \times n$ the output cipher is $m \times n$), while the Enhanced PORE is fixed (plain-text of m bytes will output a cipher-text of $2 \times m$ bytes).

5 Conclusion

Homomorphic encryption becomes an efficient solution for different modern systems and applications for preserving users privacy. Indeed, in this paper, we extend the previous solution of [9], design and realize a dynamic solution explained into the PORE approach towards overcoming its original vulnerability. According to the presented security analysis, Enhanced PORE with its dynamic approach has shown a high degree of security. Benefiting from its dynamic implementation, comes with using short encryption sessions and a dynamic $p-boxes$ selection at the message level. A comparison between the Enhanced PORE and the Enhanced MORE is given in term of latency, security and storage overhead, which indicates that similar cryptographic and efficiency performances are obtained between them. Therefore, the main goal of this paper is to provide a new HE algorithm candidate.

Acknowledgment

This work has been partially funded with support from the Lebanese University.

References

[1] K. Hariss, H. Noura, A.E. Samhat, M. Chamoun, *Design and Realization of a Fully Homomorphic Encryption Algorithm for Cloud Applications*, in *Risks and Security of Internet and Systems*, edited by N. Cuppens, F. Cuppens, J.L. Lanet, A. Legay, J. Garcia-Alfaro (Springer International Publishing, Cham, 2018), pp. 127–139

[2] S.M. Anggriane, S.M. Nasution, F. Azmi, *Advanced e-voting system using Paillier homomorphic encryption algorithm*, in *2016 International Conference on Informatics and Computing* (2016), pp. 338–342

[3] P. Raj, G.C. Deka, *Handbook of Research on Cloud Infrastructures for Big Data Analytics, Chap.19*, 1st edn. (IGI Global, Hershey, PA, USA, 2014)

[4] P. Martins, L. Sousa, A. Mariano, *A Survey on Fully Homomorphic Encryption An Engineering Perspective*, in *ACM Comput. Surv.* (ACM, New York, NY, USA, 2017), Vol. 50, pp. 83:1–83:33

[5] R.L. Rivest, A. Shamir, L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems* (1978)

[6] M. van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan, *Fully Homomorphic Encryption over the Integers*, in *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques* (Springer-Verlag, Berlin, Heidelberg, 2010), EUROCRYPT'10, pp. 24–43

[7] Z. Brakerski, C. Gentry, V. Vaikuntanathan, *(leveled) fully homomorphic encryption without bootstrapping* (2014)

[8] J. Domingo-Ferrer, *A Provably Secure Additive and Multiplicative Privacy Homomorphism*, in *Proceedings of the 5th International Conference on Information Security* (Springer-Verlag, London, UK, UK, 2002), ISC '02, pp. 471–483

[9] K. Hariss, H. Noura, A.E. Samhat, *Fully enhanced homomorphic encryption algorithm of more approach for real world applications* (2017)

[10] A. Kipnis, E. Hibshoosh, *Efficient methods for practical fully-homomorphic symmetric-key encryption, randomization, and verification* (2012)

[11] P. Zhang, Y. Jiang, C. Lin, Y. Fan, X. Shen, *P-Coding: Secure Network Coding against Eavesdropping Attacks*, in *2010 Proceedings IEEE INFOCOM* (2010), pp. 1–9