



An efficient fully homomorphic symmetric encryption algorithm

Khalil Hariss^{1,2} · Hassan Noura^{3,4} · Abed Ellatif Samhat¹

Received: 13 January 2018 / Revised: 2 October 2019 / Accepted: 26 November 2019 /

Published online: 11 January 2020

© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

In this paper, we consider Homomorphic Encryption (HE) to process over encrypted data to achieve users privacy. We present a framework solution that provides a high level of security for the symmetric HE algorithms. The proposed solution introduces a dynamic structure and a dynamic diffusion primitives that enhance existing symmetric HE algorithms and overcome their weaknesses. Domingo Ferrer is a well known symmetric HE scheme that relies on polynomial computations but at the same time suffers from some vulnerabilities and especially sensitivity to known plain-text attack. We apply the concerned dynamic framework over the Domingo Ferrer encryption scheme to overcome its main weaknesses. Security analysis of the new encryption scheme that we called Enhanced Domingo Ferrer has shown that the latter became immune to several types of attack especially known plain-text attack. Crypt-analysis has also shown that this new implementation will be secure also with the lowest possible storage overhead. Implementation of the new scheme has shown an acceptable execution time. All the new specifications listed previously make the scheme a good candidate for efficiently preserving users privacy in a big variety of real-world modern applications.

Keywords Fully homomorphic encryption · Secure multimedia processing · Dynamic diffusion and permutation primitives · Polynomial resultant · Known plain-text attack

1 Introduction

Encryption is one of the most important techniques that preserve data confidentiality and users privacy. In some critical cases, such as cloud computing, where computation over

✉ Hassan Noura
hn49@aub.edu.lb

¹ Faculty of Engineering - CRSI, Lebanese University, Hadath, Lebanon

² Engineering School, ESIB, Saint Joseph University, Mar Roukoz, Beirut, Lebanon

³ Department of Electrical and Computer Engineering, American University of Beirut, Beirut, Lebanon

⁴ Department of Computer Sciences, Arab Open University, Beirut, Lebanon

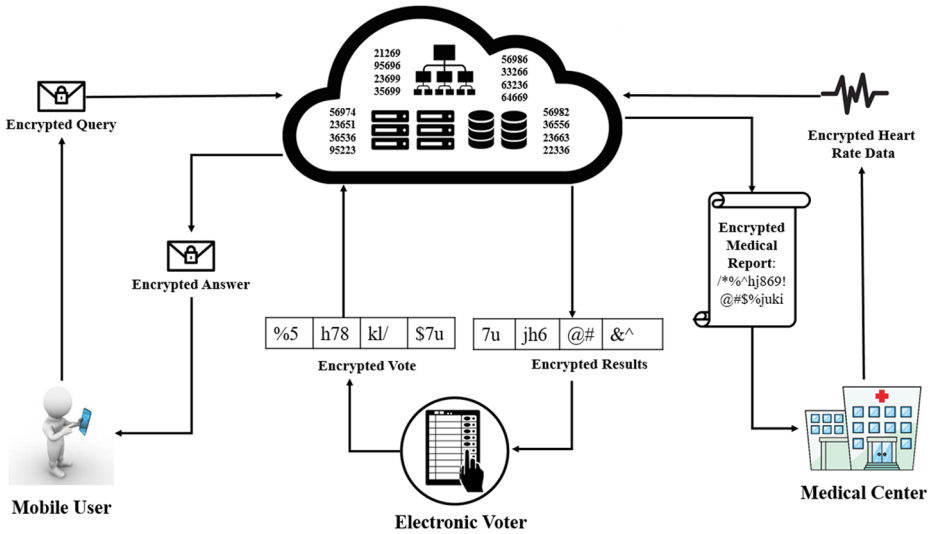


Fig. 1 HE in a cloud scenario

encrypted data is a need, traditional encryption techniques become insufficient. In this situation, users are obliged to reveal some of their secret parameters for non-trusted parties for performing computation over their sensitive data after decrypting it and then re-encrypting. The root for solving the concerned problem comes with Homomorphic Encryption (HE). HE is a new cryptographic research topic that allows third untrusted parties to process over encrypted data to achieve users privacy. It is required in several real modern applications especially with advanced secure cloud-based systems [23, 24]. An illustration for HE usage in modern applications (such as cloud querying [7, 17, 18], e-voting [2], medical data analysis [21, 36], IoT [19, 27, 32], etc) is given in Fig. 1. In the latter figure, cloud is performing operations over encrypted storage. The cloud is unaware of what is happening.

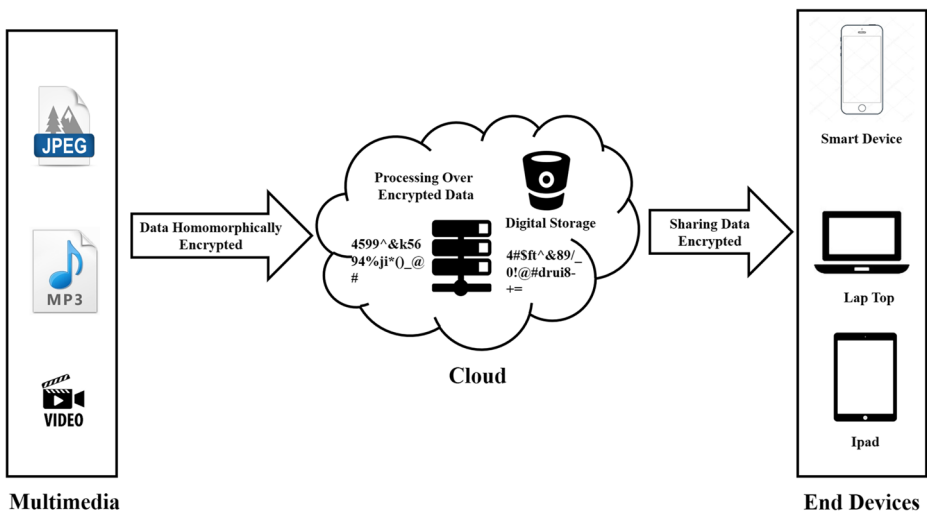


Fig. 2 HE in multimedia

Image processing in the cloud is a common application for HE in the digital world. Many images with information of high sensitivity and confidentiality are shared with the cloud to be processed and sent to different devices as discussed in [5, 38]. A cloud based implementation of HE with multimedia tools and devices is given in Fig. 2.

Figure 3 is given to clarify the concept of HE, where a simple circuit C is presented with 4 different plain-texts $x_i(1 \leq i \leq 4)$ with their respective cipher-texts $c_i(1 \leq i \leq 4)$ under an encryption scheme Enc with a secret key K . Circuit C is formed from a set of addition and multiplication gates since in real-world any electrical circuit C can be written as Boolean function formed of addition and multiplication operations. Building a HE scheme resides in finding an equivalence between computing over the cipher-texts and the plain-texts. In other words, an encryption scheme that satisfies the following relation should be built:

$$Cipher_{output} = Enc_K(Plain_{output}) \tag{1}$$

Having these two basic properties:

1. Addition

$$Enc_K(x) + Enc_K(y) \text{ mod } N = Enc_K(x + y, \text{ mod } M) \tag{2}$$

2. Multiplication

$$Enc_K(x) \times Enc_K(y) \text{ mod } N = Enc_K(x \times y, \text{ mod } M) \tag{3}$$

where $x, y \in \text{ring } Z_M$, Enc is the encryption function and K is a secret key.

Using these homomorphic properties, the required relation given in (1) can be demonstrated as follows:

$$\begin{aligned}
 Cipher_{output} &= ((c_{1+2}) \times (c_{3+4})) = (c_1 + c_2) \times (c_3 + c_4) = (Enc_K(x_1) + Enc_K(x_2)) \times \\
 &(Enc_K(x_3) + Enc_K(x_4)) = Enc_K(x_1 + x_2) \times Enc_K(x_3 + x_4) = Enc_K((x_1 + x_2) \times \\
 &(x_3 + x_4)) = Enc_K(x_{1+2} \times x_{3+4}) = Enc_K(Plain_{output}).
 \end{aligned}$$

As a conclusion, to build a Fully Homomorphic Encryption (FHE) scheme, the two basic properties given in both (2) and (3) should be satisfied. Several HE schemes have been investigated and presented in the literature [1, 6, 10, 13, 16, 29].

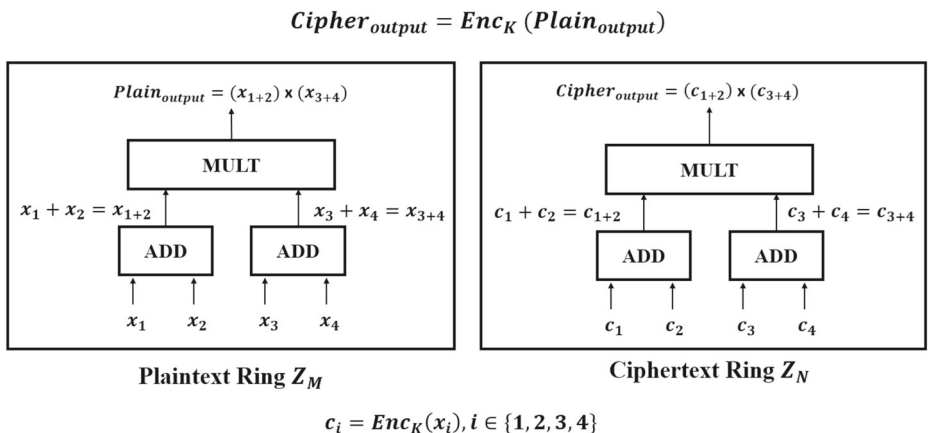


Fig. 3 Homomorphic computation

Existing HE schemes can be decomposed into two classes: asymmetric and symmetric schemes. Asymmetric schemes such as Gentry Cryptosystem [14, 15], DGHV [33] and BGV [3] and symmetric schemes such as Domingo Ferrer [11, 12], PORE (Polynomial Operation for Randomization and Encryption) [20, 30] and MORE (Matrix Operation for Randomization and Encryption) [37]. Asymmetric homomorphic algorithms suffer from high computational complexity, which prevents to consider them as a good solution. For example, DGHV public key size can attain 2,3 Gigabytes and with an optimized implementation on a high-end workstation, key generation takes 2,2 hours, encryption takes 3 minutes, and cipher-text refresh takes 30 minutes as given in [9]. Crypt-analysis has also shown that DGHV is sensitive to Greatest Common Divisor attack (GACD Attack) [8]. BGV also works over lattices where its computational complexity is also given by large poly-logarithmic factors. Symmetric ones suffer from a weak security level since they cannot resist to different types of attacks such as the chosen and known plain-text ones [11, 37].

Following the two classes of HE schemes, researchers are divided into two groups. The first group is trying to optimize the cipher implementation to reduce the computation resources in addition to communication and storage overhead of asymmetric schemes. While the second group is trying to enhance the security level to benefit from the efficiency and simplicity of symmetric ones.

In this paper, we follow the second group, and the main contribution is to present a new dynamic framework that can be implemented over homomorphic symmetric cipher schemes. The proposed solution introduces dynamic key-dependent cryptographic primitives that help the Domingo-Ferrer [11, 12] encryption scheme to reach a high level of security. In parallel, we are still benefiting from the simplicity. Also, we propose a compression technique to reduce the storage/communication overhead of the original Domingo-Ferrer cipher scheme. The main goal is to design an efficient fully homomorphic cipher scheme that can reach a good balance between security and performance level. Different security tests and crypt-analysis are done to evaluate the performance of the resulting algorithm that showed a high level of security and resistance against several types of attacks with acceptable execution time.

In the literature, it is mentioned that most cloud hosting systems rely on asymmetric encryption schemes due to their security level despite their computational complexity and storage overhead. The main purpose of this dynamic implementation is to enhance symmetric cipher schemes such as Domingo-Ferrer to guarantee for cloud systems simultaneously the high level of security and the low levels of computational complexity and storage overhead. This work opens the door for modern fully symmetric homomorphic ciphers that can reach a good balance between security and system performance by using the dynamic key-dependent cipher approach.

The rest is organized as follows, Section 2 explains the Domingo Ferrer HE algorithm. Section 3 introduces the proposed solution. Security analysis and performances of the resultant algorithm Enhanced Domingo Ferrer are given in Section 4 in addition to a comparison with Domingo Ferrer approach in terms of execution time. Conclusions and future works are presented in Section 5.

2 Domingo Ferrer approach

In [12], Joseph Domingo Ferrer described in 2002 a FHE scheme based on polynomial calculations, that supports both homomorphic properties: addition and multiplication.

2.1 Encryption parameters

The public parameters are (d, m) , where d is a positive integer ≥ 2 and m is a large integer. m should have many small divisors and at the same time there should be many integers less than m that can be inverted modulo m .

The secret key is $k = (r, m')$ where the secret parameter $r \in Z_m$ such that $r^{-1} \bmod(m)$ exists, $m' > 1$ is a divisor of m ($m = (m')^\lambda$, where λ is a security parameter). In this case the plain-texts are in the private ring $Z_{m'}$ and the cipher-texts are in the public ring $(Z_m)^d$.

2.2 Encryption process

Randomly split a plain-text $x \in Z_{m'}$ into d random secret elements: $x_1, x_2, x_3, \dots, x_d$, such $x = \sum_{j=1}^d x_j \bmod(m')$ and $x_j \in Z_{m'}$.

Compute $E_{(k,x)} = (x_1r, x_2r^2, \dots, x_dr^d) \bmod(m)$.

$E_{(k,x)}$ can be expressed in a polynomial form of variable t as follows:

$$E_{(k,x)}(t) = (x_1r)t + (x_2r^2)t^2 + (x_3r^3)t^3 + \dots + (x_dr^d)t^d \bmod(m).$$

2.3 Decryption process

Compute the product of the j^{th} coordinate by $r^{-j} \bmod(m)$ to retrieve $x_j \bmod(m)$ and compute $\sum_{j=1}^d x_j \bmod(m')$ to get x .

2.4 Homomorphic properties

Starting from two different plain-texts x_1 and $x_2 \in Z_{m'}$. c_1 and c_2 are respectively two cipher-texts based on the encryption process defined above (Section 2.2) and given by:

$$c_1 = [x_1^{(1)}r, x_1^{(2)}r^2, \dots, x_1^{(d)}r^d] \bmod(m)$$

$$c_2 = [x_2^{(1)}r, x_2^{(2)}r^2, \dots, x_2^{(d)}r^d] \bmod(m)$$

1. **Addition:** Using polynomial addition, the following can be written:

$$c_{add} = c_1 + c_2 = [(x_1^{(1)} + x_2^{(1)})r, (x_1^{(2)} + x_2^{(2)})r^2, \dots, (x_1^{(d)} + x_2^{(d)})r^d].$$

Using the secret key $[r^{-1}, r^{-2}, \dots, r^{-d}]$, $Dec_k(c_1 + c_2) = Dec_k(c_1) + Dec_k(c_2)$ is simply demonstrated. Thus, the scheme is additive homomorphic.

2. **Multiplication:** It works like in the case of polynomials: all terms are cross multiplied in Z_m , with a d_1^{th} degree by a d_2^{th} term yielding $(d_1 + d_2)^{th}$. Finally terms having the same degree are added up.

$$c_{mult} = c_1 \times c_2 = [(x_1^{(1)}x_2^{(1)})r^2, (x_1^{(1)}x_2^{(2)} + x_2^{(1)}x_1^{(1)})r^3, (x_1^{(1)}x_2^{(3)} + x_1^{(2)}x_2^{(2)} + x_1^{(3)}x_2^{(1)})r^4, \dots, x_1^{(d)}x_2^{(d)}r^{2d}].$$

Using the secret key $[r^{-2}, r^{-3}, r^{-4}, \dots, r^{-2d}]$, $Dec_k(c_1 \times c_2) = Dec_k(c_1) \times Dec_k(c_2)$ is validated and the scheme is homomorphic multiplicative.

encryption scheme (we recall that λ is the secret parameter, (m', r) is the secret key, d and m are the public parameters).

1. Recovering the secret parameter m' : From the M couples of (plain-text,cipher-text) given above, two couples $(a_i, C_i)/(a_{i+1}, C_{i+1})$ are randomly picked. Then the two following polynomials are built in the ring $Z_{m'}$:

$$f_i(t) = C_i^1 t + C_i^2 t^2 + C_i^3 t^3 + \dots + C_i^d t^d - a_i \pmod{m'}$$

$$f_{i+1}(t) = C_{i+1}^1 t + C_{i+1}^2 t^2 + C_{i+1}^3 t^3 + \dots + C_{i+1}^d t^d - a_{i+1} \pmod{m'}$$

It is sure that the two uni-variate polynomials f_i and f_{i+1} share the common secret parameter r^{-1} as a root. Calculating the Sylvester Matrix for these two polynomials, will give a multiple for the secret modulus m' (we recall that $f_i(r^{-1}) = f_{i+1}(r^{-1}) = 0 \pmod{m'}$ but m' until now is unknown).

The above procedure is repeated $2 \times poly(\lambda)$ times to form a set T that contains $2 \times poly(\lambda)$ multiples of m' .

The set $T = \{Syl_j(f_k, f_{k+1}), \text{ where } j \in \{1, 2, 3, \dots, 2 \times poly(\lambda)\}\}$.

Let $Y = [gcd(Syl_j, Syl_{j+1}), j \in \{1, 3, \dots, 2 \times poly(\lambda) - 1\}]$.

Depending on the well known fact given in [34], having two large integers a and b , $gcd(a, b) = 1$ is achieved with a probability close to $\frac{6}{\pi^2}$. Based on this fact, the probability to pick randomly $\{T_j, T_l\} \subset T$ having $gcd(T_j, T_l) = m'$ is close to $\frac{6}{\pi^2}$. Hence the probability to obtain at least one m' among the values of Y is close to $(1 - (1 - \frac{6}{\pi^2})^{poly(\lambda)})$. Finally m' is the most common vote among the $poly(\lambda)$ values of Y .

2. Recovering the equivalent secret parameter r^\diamond : After revealing the secret modulus m' , the equivalent secret key to be recovered is $r^\diamond = r \pmod{m'}$. To do this, from the M couples of plain-text/cipher-text, d couples are picked to form the set of polynomial equations given in (5).

$$\begin{aligned}
 &C_i^1 r^{-1} + C_i^2 r^{-2} + C_i^3 r^{-3} + \dots + C_i^d r^{-d} = a_i \pmod{m'} \\
 &C_{i+1}^1 r^{-1} + C_{i+1}^2 r^{-2} + C_{i+1}^3 r^{-3} + \dots + C_{i+1}^d r^{-d} = a_{i+1} \pmod{m'} \\
 &C_{i+2}^1 r^{-1} + C_{i+2}^2 r^{-2} + C_{i+2}^3 r^{-3} + \dots + C_{i+2}^d r^{-d} = a_{i+2} \pmod{m'} \\
 &\dots\dots\dots \\
 &C_{i+d-1}^1 r^{-1} + C_{i+d-1}^2 r^{-2} + C_{i+d-1}^3 r^{-3} + \dots + C_{i+d-1}^d r^{-d} = a_{i+d-1} \pmod{m'}
 \end{aligned}
 \tag{5}$$

Supposing that $z_h = r^{-h}$ for $1 \leq h \leq d$, the polynomial equations given in (5) are transformed into a linear system of d equations with the d unknowns (z_1, z_2, \dots, z_d) given by the following where $0 \leq j \leq d - 1$:

$$C_{i+j}^1 z_1 + C_{i+j}^2 z_2 + C_{i+j}^3 z_3 + \dots + C_{i+j}^d z_d = a_{i+j} \pmod{m'} \tag{6}$$

Finding the equivalent secret key resides in finding the value of $(z_1)^{-1} = r^\diamond$.

This is only possible when the relative matrix of the linear system given in (6) is invertible in the ring $Z_{m'}$.

Based on [4], given a prime number p , the probability $f(p)$ to find an invertible matrix in the ring Z_p is at least e^{-2h} where $h = \frac{1}{p-1}$.

The secret modulus m' can be decomposed into its prime factors as follow, $m' = p_1^{e_1} p_2^{e_2} p_3^{e_3} \dots p_k^{e_k}$.

In this case, the probability of finding an invertible matrix in $Z_{m'}$ is $f(m') = f(p_1) \times f(p_2) \dots \times f(p_k) \geq e^{-2h(m')}$ where $h(m') = \sum_{\{p_1, p_2, \dots, p_k\}} \frac{1}{p_i - 1}$.

After finding the invertible matrix in ring $Z_{m'}$, the linear system given in (6) can be resolved using Gaussian elimination with $O(d^3(\log(m'))^2)$, but only the value of z_1 is needed from the solution.

As a conclusion, Domingo Ferrer is secure against this attack only when M the number of couples (plain-text/cipher-text) is lower than the dimension d . In Section 4, it is shown that the proposed dynamic framework strengthens Domingo Ferrer scheme against this type of attacks even with the lowest possible value of the dimension d (i.e $d = 2$).

3 Proposed solution

Symmetric approaches are considered in this work. Several enhancements are proposed to overcome the drawbacks of the existing ones. The proposed enhancements are mainly implemented over Domingo Ferrer approach while preserving its homomorphic properties.

3.1 Enhanced Domingo Ferrer implementation

The proposed solution employs a dynamic structure in addition to the use of dynamic diffusion primitives. Figure 4 shows different steps of the proposed Enhanced Domingo Ferrer algorithm that will be detailed below.

3.1.1 Dynamic key generation

The two end hosts should agree on two secret parameters: a secret key and an initial vector (IV). In the proposed solution, the secret key is mixed with IV and the output is hashed

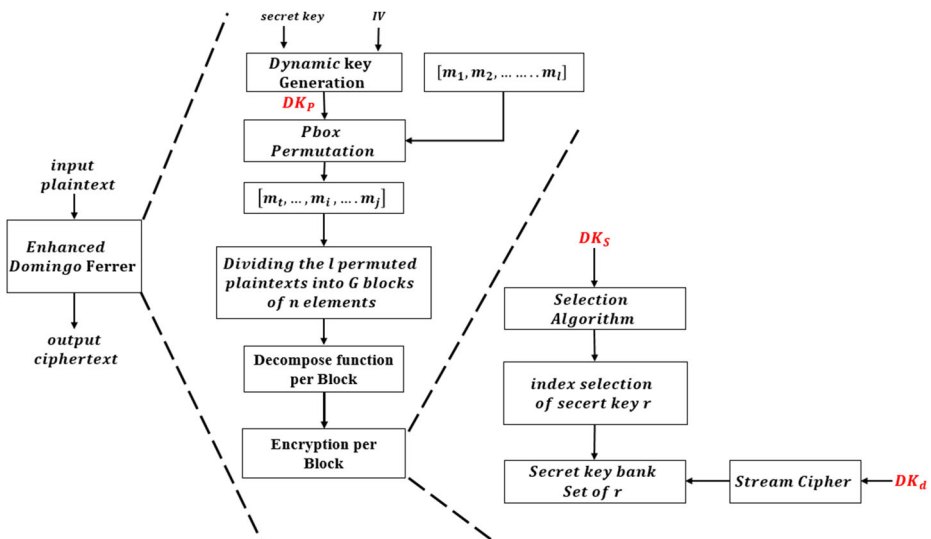


Fig. 4 Enhanced Domingo flow chart

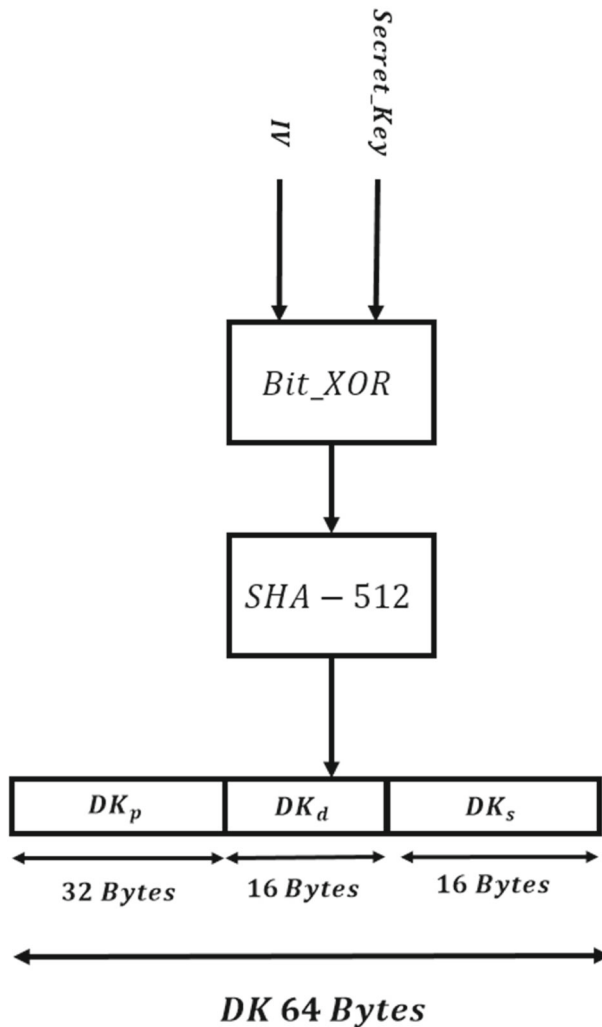


Fig. 5 Dynamic key generation

by using the secure hash function *SHA - 512* to produce a dynamic key (*DK*) that has 512 bits (64 bytes) length. This dynamic key *DK* is divided into three different dynamic sub-keys $DK = \{DK_p || DK_s || DK_d\}$, where each one of them is used to form a required cryptographic primitive. More details are presented in the following:

1. Permutation sub-key DK_p : It consists of the most significant 32 bytes of *DK* and it is used to construct a dynamic permutation table. The latter is used to permute the bytes of plain-texts. Let us indicate that the permutation encryption process preserves the homomorphic properties.
2. Diffusion sub-key DK_d : It consists of the second least significant 16 bytes of *DK*. DK_d is employed to produce a bank of invertible secret keys (r).

3. Selection sub-key DK_s : It consists of the first least significant 16 bytes of DK . DK_s is used to produce a selection (based on the produced permutation table) table. This table is used to select dynamically one of the produced secret key from the secret key bank for each input block.

The dynamic keys generation DK , and its corresponding sub-keys are (DK_p, DK_s, DK_d) are illustrated in Fig. 5.

The size of the secret key should be at least 128 bits, which means that its corresponding key-space is 2^{128} . Besides, the size of the dynamic key is 512 bits, and consequently, its corresponding key-space 2^{512} . This means that the proposed fully homomorphic cipher scheme can resist the brute-force attack.

3.1.2 Permutation box

Using DK_p , a permutation box is generated and applied over the input plain-text. In the proposed Enhanced Domingo Ferrer implementation, the creation of a permutation box is done similar to [26]. The key dependent permutation technique is employed because it preserves the homomorphic properties [26, 39].

The homomorphic behavior of a permutation box is given by:

Suppose that we have a permutation box 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 $\pi(Y) = [y_{p_i}]_{1 \leq i \leq N}$.

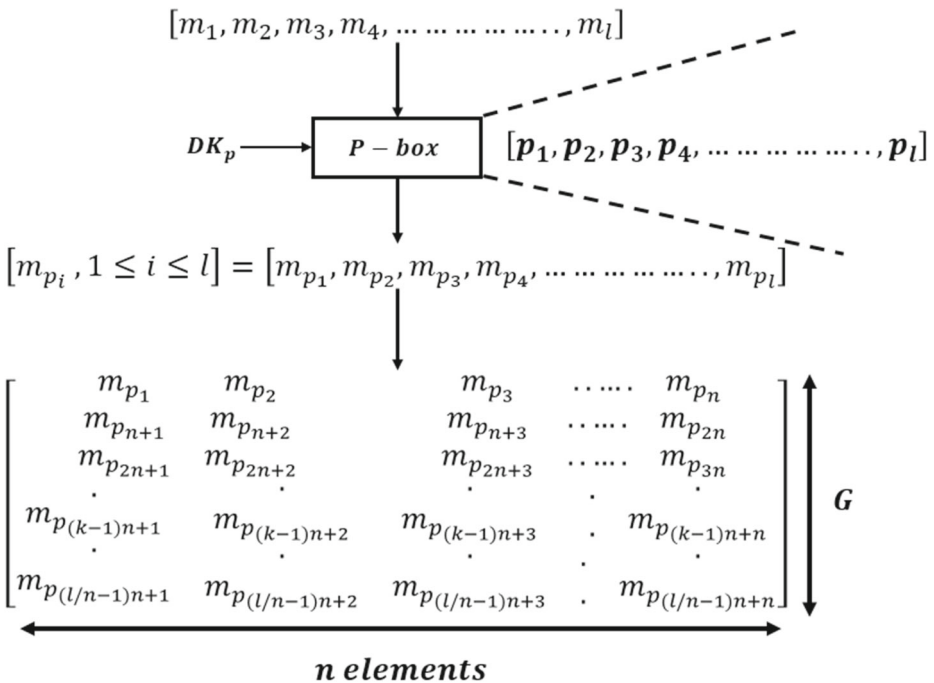


Fig. 6 Block decomposition

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 $\pi(X \odot Y) = \pi(X) \odot \pi(Y)$, the homomorphic behavior of π is deduced.

3.1.3 Dynamic block encryption

As shown in Fig. 4, the permuted plain-texts are decomposed into a G blocks, where $G = \lceil \frac{L}{n} \rceil$, and n is the block size. A detailed explanation of this procedure is given in Fig. 6.

Each block of dimension n is encrypted with Domingo Ferrer approach using an encryption secret key r chosen dynamically from a secret key bank based on a dynamic selection algorithm after the *DecomposeFunction*.

As given in Fig. 7, the dynamic block encryption of the k^{th} block represented by $[m_{p(k-1)n+1}, m_{p(k-1)n+2}, m_{p(k-1)n+3}, \dots, m_{p(k-1)n+n}]$ where $1 \leq k \leq G$, is given by the following steps:

- Step 1 (Index Selection):** Based on the index k , δ_k is chosen from a permutation box $\Delta = [\delta_1, \delta_2, \dots, \delta_k, \dots, \delta_G]$ built based on the selection secret key DK_s .
- Step 2 (Secret Key Selection):** Based on δ_k , secret key r_{δ_k} is chosen from the secret key bank.
- Step 3 (Dynamic Encryption):** Each element $m_{p(k-1)+j}$ from the k^{th} block, where $1 \leq j \leq n$ is decomposed by the *DecomposeFunction* then encrypted as follow $[(m_{p(k-1)n+j})^{(1)}r_{\delta_k}, (m_{p(k-1)n+j})^{(2)}r_{\delta_k}^2]$.

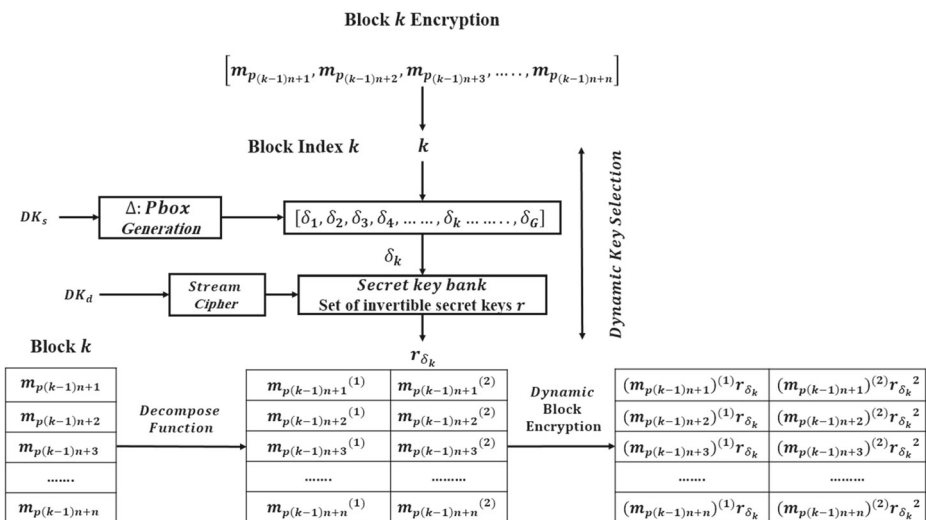


Fig. 7 Dynamic block encryption

3.1.4 Proposed decompose function

The main job of the *DecomposeFunction* is to decompose any plain-text x in the private ring $Z_{m'}$ into d random secret elements $\{x_1, x_2, x_3, \dots, x_d\}$ in the public ring $Z_{\theta m'}$, such that $x = \sum_{i=1}^d x_i \pmod{m'}$.

Figure 8, presents the first step the *Decomposefunction* that divides a plain-text from $Z_{m'}$ into two integers (d is taken 2 for simplicity) in $Z_m = Z_{\theta m'}$. $[x_1, x_2, \dots, x_l]$ is a vector of l plain-texts in the private ring $Z_{m'}$, where each private plain-text x_k ($k \in \{1, 2, 3, \dots, l\}$) is decomposed into two secret integers x^1_k, x^2_k in the public ring $Z_{\theta m'}$ such that $x^1_k + x^2_k = X \in Z_{\theta m'}$ and $X \pmod{m'} = x_k$ (X is called the extended plain-text). The *DecomposeFunction* is based on the following two steps:

- First Step:** the first step is given in Fig. 8, where the *DecomposeFunction* takes two input parameters: the input plain-texts vector $([x_1, x_2, \dots, x_l])$, and an integer sequence s in the ring $Z_{\theta m'}$ (the sequence s is generated based on DK_d and a cipher algorithm like RC4). The sequence s is transformed into a real sequence $h \in [0, 1]$, and for each plain-text x_k , a set $\{X_i\}$ of extended integers is generated in the ring $Z_{\theta m'}$ such that $\frac{\theta m'}{2} < X_i < \theta m', X_i \pmod{m'} = x_k$ and $length\{X_i\} = W$ (Fig. 8).
- Second Step:** X is the extended integer chosen from the set $\{X_i\}$ based on a dynamic selection algorithm constructed from the sequence h and W the length of $\{X_i\}$ ($X = X_{ceiling(h(k) \times W)}$). Once X is chosen, the first decomposition parameter x^1_k is a random integer between 1 and $\frac{\theta m'}{2}$, and the second one is $x^2_k = X - x^1_k$.

3.1.5 Enhanced Domingo Ferrer block encryption

As explained previously in Figs. 4, 6 and 7, Enhanced Domingo Ferrer encryption is done at the block level. From each block a plain-text is taken and decomposed using the *DecomposeFunction* listed above and secret parameters for each block are selected from a secret key bank based on a dynamic selection algorithm. A detailed explanation of the Enhanced Domingo Ferrer block encryption is given in Section 3.1.3 (Dynamic Block Encryption).

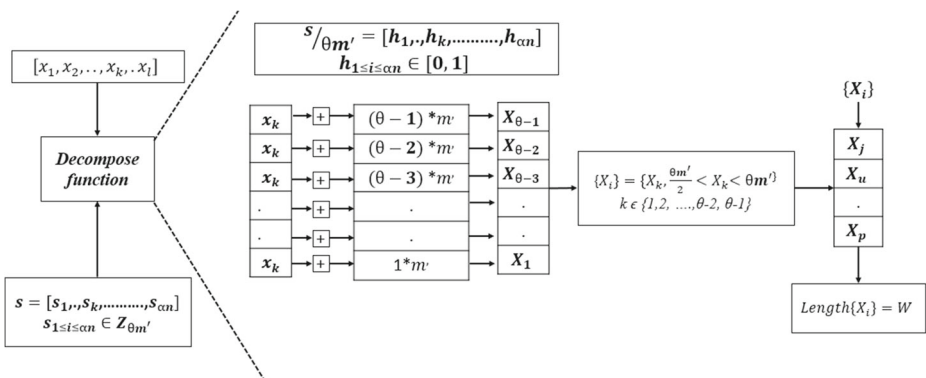


Fig. 8 First step of decompose function

3.1.6 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 the Domingo Ferrer Decryption process, the secret key r should be invertible by the multiplicative law in the public ring Z_m . We propose a generation algorithm to generate a shared secret key bank between the two end hosts. The shared secret key bank should have the following form $[k, r_k, r_k^{-1}]$, where $k \in \{1, 2, \dots, H\}$ and H is the secret bank length ($H < G$).

Algorithm 1 (r_k, r_k^{-1}) generation.

```

1: procedure ( $r_k, r_k^{-1}$ )= $r_k$ -GENERATION( $DK_d, k, \alpha, H$ )
2:    $s \leftarrow RC4(DK_d, \alpha, H)$ 
3:    $(r_k)^{-1} \leftarrow 0$ 
4:    $j \leftarrow 0$ 
5:   while  $(r_k)^{-1} = 0$  do
6:      $u(k, j) \leftarrow s_{k+j}$ 
7:      $product \leftarrow 1 + \theta \times m' \times u(k, j)$ 
8:      $i \leftarrow 2$ 
9:     while  $i \leq N - 1$  do
10:      if  $mod(product, i) = 0$  then
11:         $(r_k)^{-1} \leftarrow i$  break
12:      else
13:         $i = i + 1$ 
14:      end if
15:    end while
16:     $j = j + 1$ 
17:  end while
18:   $(r_k) \leftarrow mod(\frac{product}{(r_k)^{-1}}, \theta \times m')$ 
19: return ( $r_k, r_k^{-1}$ )
20: end procedure

```

In the Pseudo-Code listed in Algorithm 1, the two end hosts generate the k^{th} secret key (r_k). The generation of H keys requires the repetition of this pseudo code H iterations. The two end hosts know the secret key DK_d , based on it, they can generate a secret sequence s of length αH where α is a secret integer that they agree on. They should build from the secret sequence s a parameter $u(k, j)$ to pick an invertible multiplicative element r_k in the ring Z_m .

In general, the number of invertible multiplicative elements in any ring Z_m is limited that leads to a limited number of encryption keys r_k . The proposed dynamic key dependent approach reaches a high level of security since in each session the H secret keys are generated dynamically in different order based on DK . In addition, during the encryption process a key selection algorithm based on a dynamic approach as given in Fig. 7.

3.1.7 Dynamic key selection algorithm

The Dynamic Key selection is given based on the following steps:

1. **P-box Δ creation:** Using the DK_s and any stream cipher algorithm, another permutation box $\Delta = [\delta_i]_{1 \leq i \leq G}$ that has the length of the number of blocks G is created and $\delta_i \in \{1, 2, 3, \dots, H\}$.
2. **index δ_k selection:** For the k^{th} block, the index δ_k is chosen from the permutation box Δ as follow: $k \rightarrow [\delta_1, \delta_2, \delta_3, \dots, \delta_k, \dots, \delta_G] \rightarrow \delta_k$.
3. **Secret Key Selection:** Based on index δ_k , the secret key r_{δ_k} is picked from a secret key bank, for the k^{th} block encryption as follow:
 $\delta_k \rightarrow [Secret\ Key\ Bank\ of\ H\ keys] \rightarrow r_{\delta_k}$

An illustration of dynamic key selection is given in Fig. 9 (also mentioned in Fig. 7).

3.1.8 Stream cipher usage

In the different steps of this dynamic framework, RC4 is adopted as a stream cipher for generating different secret sequences and permutation boxes. Several publications in the literature like [22, 25] talked about breaking this stream cipher. But RC4 in this dynamic implementation is not mandatory, any more robust stream cipher can be used such as AES-CTR for example. Also RC4 is not used for ciphering plain-texts. Indeed RC4 is only used for generating random sequences.

3.1.9 Decryption process

The Decryption process is simply the inverse of the encryption process. Having DK and IV , all the 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 $r_{\delta_k}^{-1}$, such that $1 \leq k \leq G$.
2. **Second Step** After retrieving the decryption key for each block, the receiving end host apply the Domingo Ferrer decryption process.

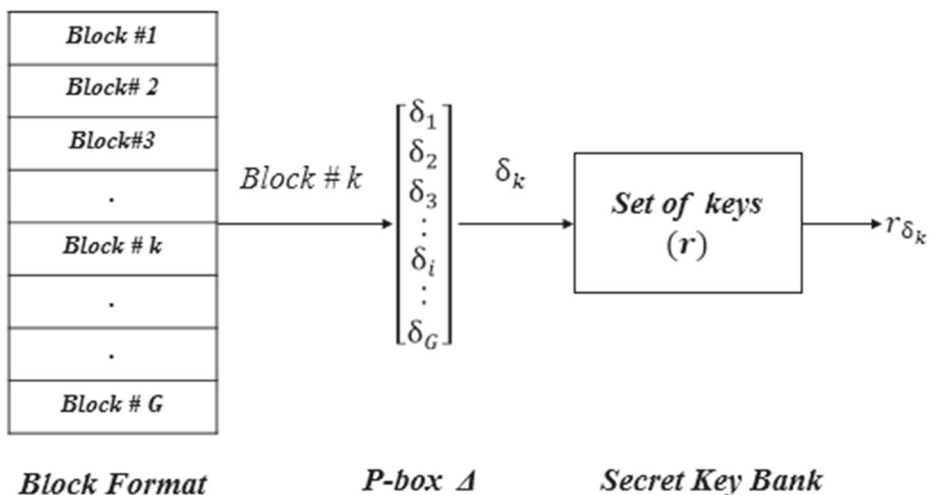


Fig. 9 Dynamic key selection

- Third Step** The receiving end host generates the inverse secret permutation vector π^{-1} by using DK_p and the following transformation:

$$\pi^{-1}[\pi[i]] = i, \text{ where } i \in \{1, 2, 3, \dots, l\} \tag{7}$$

4 Security analysis and performances

To study the performance of the resultant algorithm, several tests are done forming the security analysis. A set of plain-texts in the ring Z_{256} are taken to run different simulations under MATLAB for Enhanced Domingo Ferrer. We also compare the execution time of Enhanced Domingo Ferrer with Domingo Ferrer. As for the storage overhead, we note that the proposed algorithm will not provide additional overhead.

4.1 Proposed compression algorithm

In the proposed Enhanced Domingo Ferrer scheme, the security performances of the resulting algorithm is studied with public parameter $d = 2$. This provides the lowest storage overhead. The private ring is taken Z_{256} while the public ring is Z_{1792} ($1792 = 256 * 7, m' = 256, m = 1792$ and $\theta = 7$). The encryption of a plain-text $x \in Z_{256}$ is given by the couple $c = (c_1, c_2)$ such that $c_1, c_2 \in Z_{1792}$. One drawback of Domingo Ferrer scheme is that the cipher-text ring is larger than the plain-text ring. The original elements of plain-text are in the private ring Z_{256} while the encrypted elements are in the public ring Z_{1792} . Therefore, the size of the resulting cipher-text may be reduced by compressing it, since data compression can improve the efficiency of this scheme regarding storage or communication overhead.

To achieve compression, the following procedure is introduced: let $C = (C_1, C_2)$ denotes a cipher-text and the vectors $T_i = \text{mod}(C_i, 256)$ and $Q_i = \text{floor}(C_i/256), i \in \{1, 2\}$. Figure 10 shows The distributions of Q_1 and Q_2 has shown a lot of redundancy and can be compressed in both cases. The well-known *LZW* compression algorithm is used to compress Q . Simulation under MATLAB has shown that the compression ratio for C_1 is equal to 0.7261 while for C_2 is equal to 0.7285.

4.2 Statistical analysis

Statistical Analysis is a set of tests used to evaluate the proposed algorithm against statistical attacks. Results interpretation is done similar to [28]. With Enhanced Domingo Ferrer the security tests are done over the set of vectors $T_i (i \in \{1, 2\})$ obtained after the compression process.

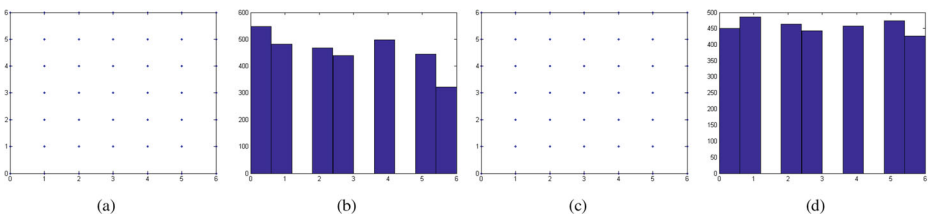


Fig. 10 a,b Q1 Distribution. c,d Q2 Distribution

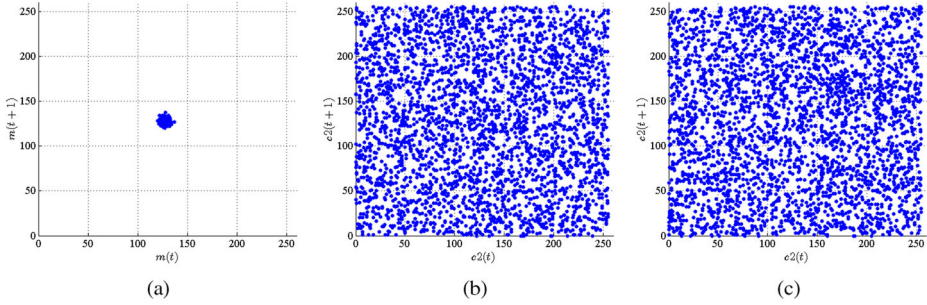


Fig. 11 Recurrence Test: **a-** Original message, **b-**Enh Domingo Ferrer T1 **c-**Enh Domingo Ferrer T2

4.2.1 Recurrence test

A good crypto-system should give a high level of randomness among the cipher-text space. Recurrence Test is used to measure the evolution of randomness and to estimate the correlations among the data by considering the variation between a sequence of data $x_i = x(i,1), x(i,2), x(i,3), \dots x(i,m)$, a vector with delay $t \geq 1$ given by $x_i(t) = x(i,t), x(i,2t), x(i,3t), \dots x(i,mt)$. Figure 11 shows the correlation between $x_i(t)$ and $x_i(t + 1)$ for the original and the encrypted data respectively. Figure 11a represents the correlation among a set of plain-texts with mean value equal to 128 and a low standard deviation equal to 16. Figure 11b and c shows the variation between $x_i(t)$ and $x_i(t + 1)$ for Enhanced Domingo Ferrer. The cipher-text space presents a high level of randomness and no clear pattern is shown after the encryption process.

4.2.2 Distribution test

To resist against statistical attacks, a strong cipher algorithm should not reveal any information about its cipher distribution. Thus, encrypted data distribution should be close to uniform. A Gaussian plain-text distribution with a mean value equal to 128 and standard deviation equal to 16 is taken in Fig. 12a, and the distribution of the obtained set of cipher-texts is illustrated in Fig. 12b, c for the Enhanced Domingo Ferrer. Comparing the different results of Fig. 12, the cipher-text distribution after applying the encryption process is close

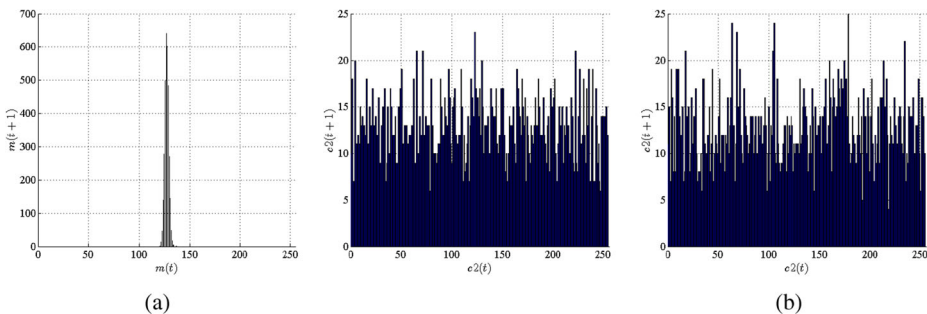


Fig. 12 Distribution Test: **a-** Original message, **b-** Enh Domingo Ferrer T1 **c-** Enh Domingo Ferrer T2

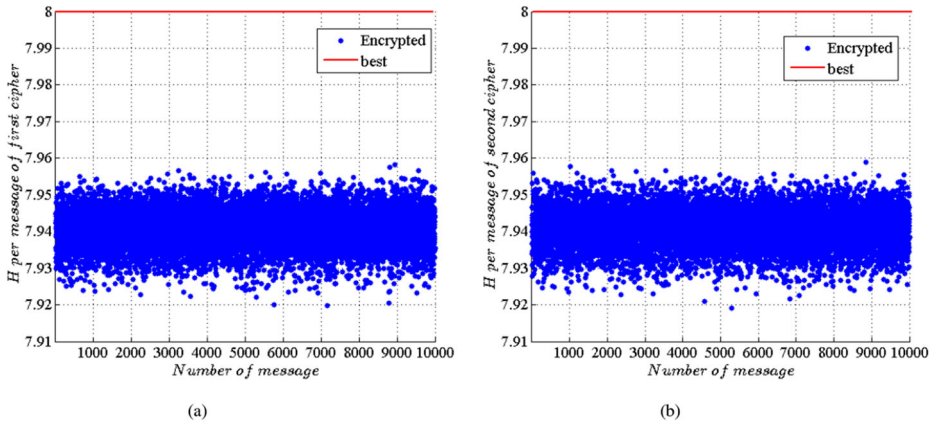


Fig. 13 Entropy Test: (a) Enh Domigo Ferrer T1 (b) Enh Domigo Ferrer T2

to a uniform distribution. As a conclusion, Enhanced Domingo Ferrer can strongly resist against any statistical attack.

4.3 Security analysis

Security Analysis tests are used to evaluate the resistance against attacks such as related key attack, chosen cipher-text/plain-text attack, side channel attack, etc. Different security tests are done for 10000 iterations.

4.3.1 Entropy test

A practical way to measure the level of uncertainty in a random variable is to calculate its entropy value. The entropy of a source message m is defined by the following equation:

$$H(m) = \sum_{i=0}^{2^M-1} p(m_i) \log_2 \frac{1}{p(m_i)} \tag{8}$$

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 the proposed implementation, the cipher values are in the ring Z_{256} , the ideal value of the entropy should be equal to 8 ($2^8 = 256$). Figure 13 presents the results of the Enhanced Domingo Ferrer. One can see that mean values are close to 8 with a low standard deviation ($std1 = 0.005252$, $std2 = 0.005298$). The resultant cipher-texts are considered as a truly random source.

4.3.2 Difference test

A good cipher algorithm should at least provide 50% difference between the cipher-texts and the plain-texts at the bit level. To evaluate this value, the difference percentage at the bit level between 10000 cipher-texts and plain-texts is calculated. Figure 14 gives the obtained results after simulation. The proposed algorithm satisfies the difference property because the mean values for both ciphers are 50 with low standard deviations ($std1 = 0.3418$, $std2 = 0.3142$).

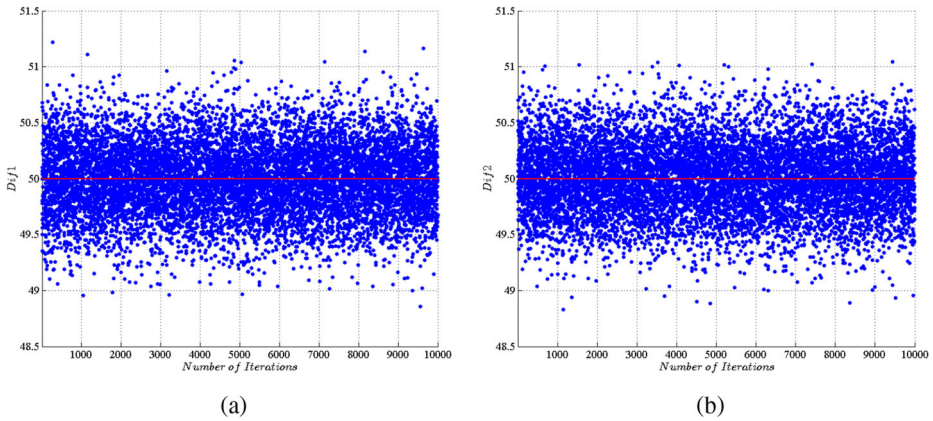


Fig. 14 Difference Test: (a) Enh Domingo Ferrer T1 (b) Enh Domingo Ferrer T2

4.3.3 Correlation test

In any crypto-system, to ensure that the cipher-text does not reveal any useful information about the plaint-texts distribution. It should give a low correlation between the distribution of cipher-texts and that of plain-texts. The correlation coefficient between an original message x and the encrypted message y is computed as follows:

$$\rho_{x,y} = \frac{cov(x, y)}{\sqrt{D(x) \times D(y)}} \tag{9}$$

where $cov(x, y) = E[\{x - E(x)\}\{y - E(y)\}]$;

$$E(x) = \frac{1}{n} \times \sum_{k=1}^n x_k$$

and $D(x) = \frac{1}{n} \times \sum_{k=1}^n \{x_k - E[x]\}^2$

The results are shown in Fig. 15. The correlation mean values are close to zero with low standard deviations ($std1 = 0.0001654$, $std2 = -6.6436 \times 10^{-6}$). This means, that the cipher-texts of the proposed algorithm does not reveal any information about the plain-texts.

4.3.4 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 are equal to those of K_w , except a random Least Significant Bit (LSB) of a random byte, and T_b is the length of the original and cipher packets (in bits). The KS is calculated as follows:

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

$w = 1, 2, \dots, 1000$.

A good crypto-system should give a KS close to 50. In Fig. 16, the KS test is done for 10000 iterations; the mean values are also close to 50 with a low standard deviations($std1 =$

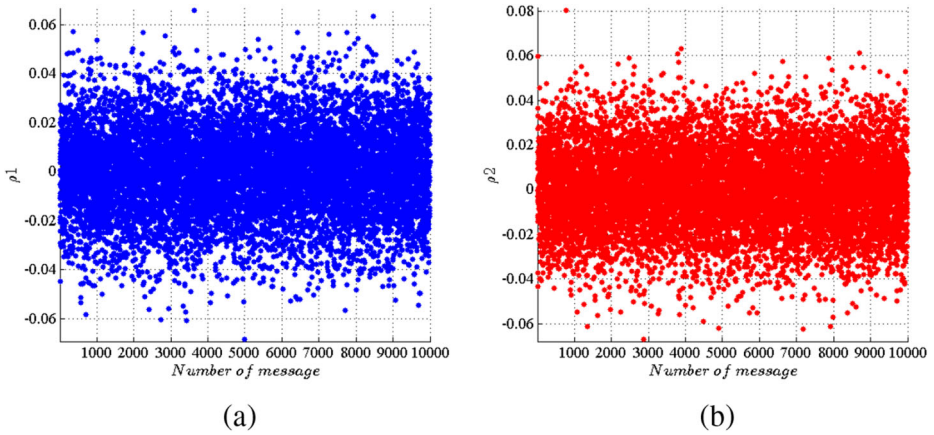


Fig. 15 Correlation Test: a-Enh Domingo Ferrer T1 b- Enh Domingo Ferrer T2

0.3146, $std2 = 03122$). As a conclusion the proposed algorithm provide high resistance against related key attacks.

4.3.5 Plaintext sensitivity

The Plain-text Sensitivity (PS) test is related to the avalanche effect. In other words, it gives the difference at the bit level between the resultant cipher-texts of two plain-texts that differ only in 1 bit. A good crypto-system should at least give 50% difference. Figure 17 shows the results of PS test for 10000 iterations. The Enhanced Domingo Ferrer provides this effect since the mean values are close to 50 with low standard deviations ($std1 = 0.3138$, $std2 = 03122$).

The robustness of the Enhanced Domingo Ferrer in the PS test resides in the randomness of the *DecompseFunction*, because encrypting the same plain-text twice will give two different cipher-texts (i.e each time the output of the *DecomposeFunction* will be different).

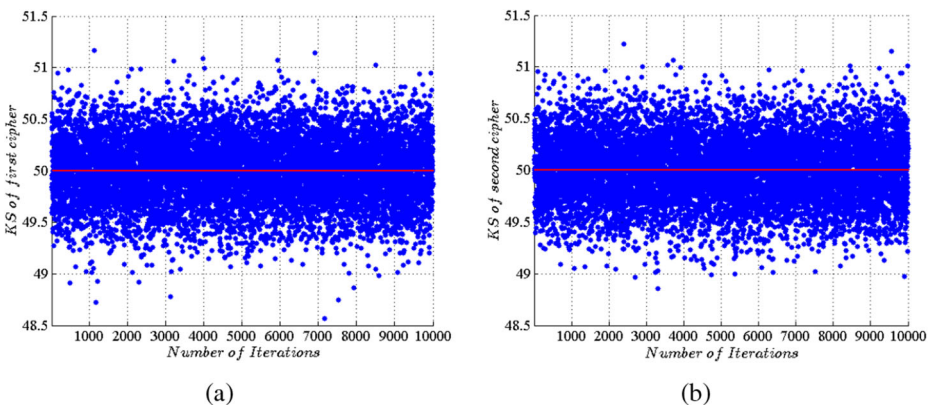


Fig. 16 KS Test: (a) Enh Domigo Ferre2 T1 (b) Enh Domigo Ferrer T2

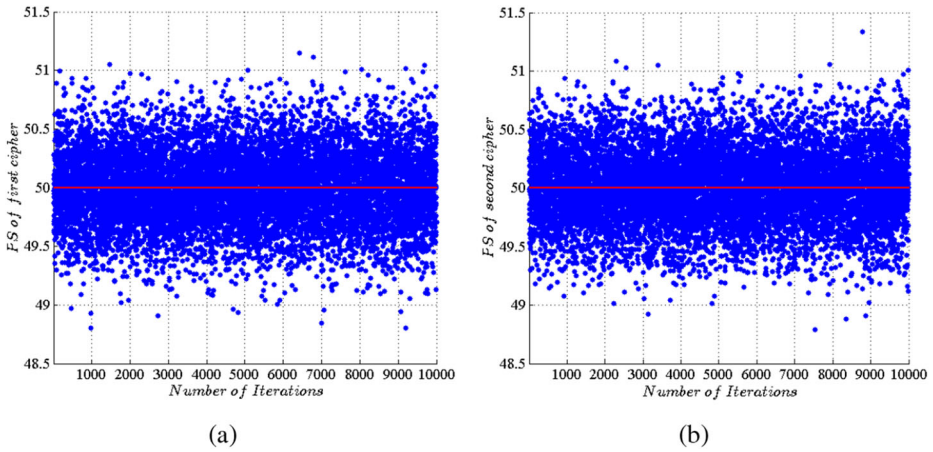


Fig. 17 PS Test: (a)-Enh Domingo Ferrer T1 (b)-Enh Domingo Ferrer T2

4.4 Enhanced Domingo Ferrer and equivalence key attack

In addition to the experimental tests listed previously, the algebraic structure of the modified Domingo Ferrer scheme is studied. We recall as explained and illustrated in Figs. 4, 6 and 7 that the proposed dynamic framework consists on dividing any message of l plain-texts into G blocks. $G = \lceil \frac{l}{n} \rceil$ where n is the number of elements per block. The dynamic implementation resides in encrypting each block of n elements with different secret parameter r chosen randomly and dynamically from a secret key bank in the ring Z_m . On the other hand, an equivalence key attack on Domingo Ferrer scheme was given in [35] and explained in Section 2.5 of this paper. To evaluate the security performance of the new encryption scheme (Enhanced Domingo Ferrer), three different cases should be taken into consideration:

1. First Case: if $n \geq d$: (d is the cipher-text dimension)

In this case, the block dimension n is higher or equal than the dimension d of the cipher-text. The cipher of a k^{th} block is $[m_{p(k-1)n+1}, m_{p(k-1)n+2}, m_{p(k-1)n+3}, \dots, m_{p(k-1)n+n}]$ is given by the following matrix:

$$\begin{bmatrix} C_{11}, C_{12}, \dots, C_{1d} \\ C_{21}, C_{22}, \dots, C_{2d} \\ \dots \\ C_{d1}, C_{d2}, \dots, C_{dd} \\ \dots \\ C_{n1}, C_{n2}, \dots, C_{nd} \end{bmatrix} = \begin{bmatrix} (m_{p(k-1)n+1})^{(1)}r_{\delta_k}, (m_{p(k-1)n+1})^{(2)}r_{\delta_k}^2, \dots, (m_{p(k-1)n+1})^{(d)}r_{\delta_k}^d \\ (m_{p(k-1)n+2})^{(1)}r_{\delta_k}, (m_{p(k-1)n+2})^{(2)}r_{\delta_k}^2, \dots, (m_{p(k-1)n+2})^{(d)}r_{\delta_k}^d \\ \dots \\ (m_{p(k-1)n+d})^{(1)}r_{\delta_k}, (m_{p(k-1)n+d})^{(2)}r_{\delta_k}^2, \dots, (m_{p(k-1)n+d})^{(d)}r_{\delta_k}^d \\ \dots \\ (m_{p(k-1)n+n})^{(1)}r_{\delta_k}, (m_{p(k-1)n+n})^{(2)}r_{\delta_k}^2, \dots, (m_{p(k-1)n+n})^{(d)}r_{\delta_k}^d \end{bmatrix}$$

The attacker can drive an equivalence key attack, while having the M couples of plain-texts and cipher-texts. Recovering the secret modulus m' will be possible. An enough number of couples (plain-text, cipher-text) encrypted with the same secret parameter r is available to build the vector $Y = [gcd(Syl_j, Syl_{j+1}), \dots, j \in \{1, 3, \dots, 2 \times poly(\lambda) - 1\}]$.

Also an attacker can build the linear system formed of d unknowns and d equations given in (6) for recovering the equivalent secret key r^\diamond in the ring $Z_{m'}$.

2. Second Case: if $2 \leq n < d$:

In this case, the block dimension n is higher or equal than 2 but lower than the dimension d of the cipher-text. The cipher of a k^{th} block is $[m_{p(k-1)n+1}, m_{p(k-1)n+2}, m_{p(k-1)n+3}, \dots, m_{p(k-1)n+n}]$ is given by the following matrix where n is lower than d :

$$\begin{bmatrix} C_{11}, C_{12}, \dots, C_{1d} \\ C_{21}, C_{22}, \dots, C_{2d} \\ \dots \\ C_{n1}, C_{n2}, \dots, C_{nd} \end{bmatrix} = \begin{bmatrix} (m_{p(k-1)n+1})^{(1)}r_{\delta_k}, (m_{p(k-1)n+1})^{(2)}r_{\delta_k}^2, \dots, (m_{p(k-1)n+1})^{(d)}r_{\delta_k}^d \\ (m_{p(k-1)n+2})^{(1)}r_{\delta_k}, (m_{p(k-1)n+2})^{(2)}r_{\delta_k}^2, \dots, (m_{p(k-1)n+2})^{(d)}r_{\delta_k}^d \\ \dots \\ (m_{p(k-1)n+n})^{(1)}r_{\delta_k}, (m_{p(k-1)n+n})^{(2)}r_{\delta_k}^2, \dots, (m_{p(k-1)n+n})^{(d)}r_{\delta_k}^d \end{bmatrix}$$

An attacker can recover the secret modulus m' by calculating the different resultants of n polynomials sharing the same root r . But building a linear system formed of d unknowns and d equations similar to the one given in (6) is impossible. The number of (plain-text, cipher-text) couples encrypted with the same secret key r is $n < d$. Thus, recovering the equivalent secret key r^\diamond will be impossible.

3. Third Case: $n = 1$:

In this case, each block is formed of one plain-text. Thus recovering both the secret modulus m' and the equivalent secret key r^\diamond is impossible. It is very hard to find two polynomials that share the same secret r as a root to find a multiple of the secret modulus m' . In the same time, if d different couples of plain-text and cipher-text are taken, the linear system given in (6) will have the following form:

$$C_{i+j}^1 z_{1j} + C_{i+j}^2 z_{2j} + C_{i+j}^3 z_{3j} + \dots + C_{i+j}^d z_{dj} = a_{i+j} \pmod{m'}, \quad (11)$$

where $1 \leq j \leq d$.

Each j^{th} linear equation given in (11) is encrypted with different secret elements as follow: $[z_{1j} = r_j^{-1}, z_{2j} = r_j^{-2}, z_{3j} = r_j^{-3}, \dots, z_{dj} = r_j^{-d}]$. Thus resolving this linear system is impossible.

4.5 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, the Enhanced Domingo Ferrer encryption scheme is implemented at the block level (i.e for each plain-text block different encryption key is chosen during the encryption from the secret key bank). The same implementation can be done at the byte level (i.e for each byte or plain-text in the ring Z_{256} a different secret key is chosen from the secret key bank: number of plain-texts per block (n) is equal to 1) which improves the security performances as explained in Section 4.4. Table 1 shows different executions time for different implementations by varying the plain-text size. Simple Domingo Ferrer is taking the lowest execution while the Enhanced Domingo Ferrer at the Byte level is taking the highest with better security performances and higher resistance against attacks.

Table 1 Execution time in seconds

| Plaintext size in bytes | 800 | 3200 | 5600 | 8000 |
|-------------------------|----------|----------|----------|----------|
| Simple Domingo | 0.0060 s | 0.0134 s | 0.0192 s | 0.0334 s |
| Enhanced Domingo block | 0.0071 s | 0.0195 s | 0.0376 s | 0.0464 s |
| Enhanced Domingo byte | 0.1743 s | 0.6802 s | 1.2415 s | 1.8283 s |

5 Conclusion

In this paper, an efficient solution is presented to introduce a high level of security for the symmetric HE algorithms. Indeed, this solution is introduced into the Domingo Ferrer to build from it a new enhanced version, that we called the Enhanced Domingo Ferrer. The resultant algorithm has shown a high level of security and a high level of randomness. The proposed Enhanced Domingo Ferrer is a very strong algorithm and can resist to several kinds of attacks after analyzing the different security test results. In the same time, it is providing an efficient execution time.

The most secure implementation comes at the byte level, where each different plain-text is encrypted with different security parameters. The byte implementation can avoid efficiently the equivalence key attack as discussed and interpreted in Section 4.4. The byte implementation is even secure with the lowest possible storage overhead ($d = 2$). As a conclusion, users of Enhanced Domingo Ferrer are free to choose the number of elements per block n in comparison with d based on their environment and security requirements.

Future work will consider implementing the enhanced version of Domingo Ferrer in a multimedia real-world application.

Acknowledgements This paper was partially supported by funds from the Maroun Semaan Faculty of Engineering and Architecture at the American University of Beirut.

Appendix: Domingo Ferrer example

- Suppose that $m' = 256$ and $m = 256 \times 7 = 1792$. (m should always be a multiple of m' , and let $d = 4$).
- Two plain-texts $x_1 = 157$ and $x_2 = 220$ are picked from the private ring Z_{256} .
- x_1, x_2 are randomly divided into 4 integers respectively x_1^j, x_2^j such that $x_1^j, x_2^j \in Z_{1792}$ and $j \in \{1, 2, 3, 4\}$:

$$x_1 = 157 = (570 + 230 + 420 + 473) \bmod(256).$$

$$x_2 = 220 = (700 + 300 + 256 + 241) \bmod(256).$$

Let $r = 717$ invertible in Z_{1792} and $r^{-1} = 5$.

- The encryption of x_1 and x_2 is given by the following:

$$E(x_1) = E(157) = (570 \times 717 \bmod(1792), 230 \times 717^2 \bmod(1792), 420 \times 717^3 \bmod(1792), 473 \times 717^4 \bmod(1792)) = (114, 726, 1652, 233).$$

$$E(x_2) = E(220) = (700 \times 717 \bmod(1792), 300 \times 717^2 \bmod(1792), 259 \times 717^3 \bmod(1792), 241 \times 717^4 \bmod(1792)) = (140, 12, 1407, 1153).$$

- Homomorphic properties:

The decryption of $E(x_1) + E(x_2)$ is done by multiplying each j^{th} position by r^{-j} .

$$r^{-1} = 5, r^{-2} = 25, r^{-3} = 125, r^{-4} = 625$$

The decryption of $E(x_1) + E(x_2)$ is given by $(1270 + 530 + 679 + 714) \bmod(256) = 121$.

Given that $(x_1 + x_2) \bmod(256) = 121$ and the proposed algorithm is additive homomorphic.

$$E(x_1) \times E(x_2) = ((114, 726, 1652, 233) \times (140, 12, 1407, 1153)) \bmod(1792)$$

The multiplication is done modulo 1792 based on a polynomial calculation, thus:

$$(114r + 726r^2 + 1652r^3 + 233r^4) \times (140r + 12r^2 + 1407r^3 + 1153r^4) = 1624r^2 + 864r^3 + 774r^4 + 1144r^5 + 1358r^6 + 1547r^7 + 1641r^8.$$

$E(x_1) \times E(x_2)$ can be expressed as (0, 1624, 864, 774, 1144, 1358, 1547, 1641).

$$\begin{aligned} r^{-1} \bmod(1792) &= 5, & r^{-2} \bmod(1792) &= 25, & r^{-3} \bmod(1792) &= 125, \\ r^{-4} \bmod(1792) &= 625, & r^{-5} \bmod(1792) &= 1333, & r^{-6} \bmod(1792) &= 1289, \\ r^{-7} \bmod(1792) &= 1069, & r^{-8} \bmod(1792) &= 1761 \end{aligned}$$

The decryption is defined by the following:

1. $a_1 = 0$
2. $a_2 = 1624 \times 25 \bmod(1792) = 1176$.
3. $a_3 = 864 \times 125 \bmod(1792) = 480$.
4. $a_4 = 774 \times 625 \bmod(1792) = 1702$.
5. $a_5 = 1144 \times 1333 \bmod(1792) = 1752$.
6. $a_6 = 1358 \times 1289 \bmod(1792) = 1470$.
7. $a_7 = 1547 \times 1069 \bmod(1792) = 1519$.
8. $a_8 = 1641 \times 1761 \bmod(1792) = 1097$.

$(1176 + 480 + 1702 + 1752 + 1470 + 1519 + 1097) \bmod(256) = 236$ given that $(x_1 \times x_2) \bmod(256) = 157 \times 220 \bmod(256) = 236$, the proposed algorithm is multiplicative homomorphic.

References

1. Aguilar-Melchor C, Fau S, Fontaine C, Gogniat G, Sirdey R (2013) Recent advances in homomorphic encryption: a possible future for signal processing in the encrypted domain. *IEEE Signal Process Mag* 30(2):108–117
2. Anggriane SM, Nasution SM, Azmi F (2016) Advaned e-voting system using paillier homomorphic encryption algorithm. In: *International conference on informatics and computing*, pp 338–342
3. Brakerski Z, Gentry C, Vaikuntanathan (2012) (leveled) fully homomorphic encryption without bootstrapping. In: *Proceedings of the 3rd innovations in theoretical computer science conference, ITCS '12*. ACM, New York, pp 309–325
4. Brent RP (1987) Determinants and ranks of random matrices over zm . *Discret Math* 66(1):35–49
5. Challa R, VijayaKumari G, Sunny B (2015) Secure image processing using LWE based homomorphic encryption. In: *IEEE International conference on electrical, computer and communication Technologies (ICECCT)*. Coimbatore, pp 1–6
6. Chan AC-F (2009) Symmetric-key homomorphic encryption for encrypted data processing. In: *2009 IEEE International conference on communications*, pp 1–5
7. Chauhan KK, Sanger AKS, Verma A (2015) Homomorphic encryption for data security in cloud computing. In: *2015 International conference on information technology (ICIT)*, pp 206–209

8. Chen Y, Nguyen PQ (2012) Faster algorithms for approximate common divisors: breaking fully-homomorphic-encryption challenges over the integers. In: Pointcheval D, Johansson T (eds) EURO-CRYPT 2012, volume 7237 of lecture notes in computer science. IACR, Springer, Cambridge, pp 502–519
9. Coron J-S, Mandal A, Naccache D, Tibouchi M (2011) Fully homomorphic encryption over the integers with shorter public keys. In: Rogaway P (ed) Advances in cryptology – CRYPTO 2011. Springer, Berlin, pp 487–504
10. Fau S, Sirdey R, Fontaine C, Aguilar-Melchor C, Gogniat G (2013) Towards practical program execution over fully homomorphic encryption schemes. In: 2013 IEEE Eighth international conference on P2P, parallel, grid, cloud and internet computing (3PGCIC), pp 284–290
11. Ferrer JD (1996) A new privacy homomorphism and applications. *Inform Process Lett* 60(5):277–282
12. Ferrer JD (2002) A provably secure additive and multiplicative privacy homomorphism. Universitat Rovira i Virgili, Dept. of Computer Engineering and Maths. In: ISC '02 Proceedings of the 5th international conference on information security. Springer, London, pp 471–483
13. Fontaine C, Galand F (2007) A survey of homomorphic encryption for nonspecialists. *Springer EURASIP J Inf Secur* 2007(1):1–10
14. Gentry C (2009) A fully homomorphic encryption scheme. PhD thesis. Stanford University
15. Gentry C (2009) Fully homomorphic encryption using ideal lattices. In: STOC '09 Proceedings of the forty-first annual ACM symposium on theory of computing. ACM, New York, pp 169–178
16. Haridas D, Venkataraman S, Varadan G (2012) Strengthened iterated Hill cipher for encrypted processing. In: 2012 2nd IEEE International conference on parallel distributed and grid computing (PDGC), pp 491–496
17. Hariss K, Noura H, Samhat AE, Chamoun M (2018) Design and realization of a fully homomorphic encryption algorithm for cloud applications. In: Cuppens N, Cuppens F, Lanet JL, Legay A, Garcia-Alfaro J (eds) Risks and security of internet and systems. Springer International Publishing, Cham, pp 127–139
18. Jin B, Jiang D, Xiong J, Chen L, Li Q (2018) D2D data privacy protection mechanism based on reliability and homomorphic encryption. *IEEE Access* 6:51140–51150
19. Kapusta K, Memmi G, Noura H (2019) Additively homomorphic encryption and fragmentation scheme for data aggregation inside unattended wireless sensor networks. *Ann Telecommun* 74(3-4):157–165
20. Kipnis A, Hibshoosh E (2012) Efficient methods for practical fully homomorphic symmetric-key encryption. Randomization and Verification IACR Cryptology ePrint Archive 2012:637
21. Kocabas O, Soyata T (2014) Medical data analytics in the cloud using homomorphic encryption, pp 471–488
22. Kwok SHM, Lam EY (2008) Effective uses of FPGAs for brute-force attack on RC4 ciphers. *EEE Trans Very Large Scale Integr Syst* 16:8
23. Li J, Li YK, Chen X, Lee PPC, Lou W (2015) A hybrid cloud approach for secure authorized deduplication. *IEEE Trans Parallel Distrib Syst* 26(5):1206–1216
24. Li P, Li J, Huang Z, Li T, Gao C-Z, Yiu S-M, Chen K (2017) Multi-key privacy-preserving deep learning in cloud computing. *Futur Gener Comput Syst* 74:76–85
25. Mister S, Tavares SE (1998) Cryptanalysis of RC4-like Ciphers. *Selected Areas in Cryptography*
26. Noura H, Courrousé D (2015) Hldca-wsn:homomorphic lightweight data confidentiality for wireless sensor network. *Int Assoc Cryptogr Res IACR* 2015:928
27. Noura H, Salman O, Chehab A, Couturier R (2019) Preserving data security in distributed fog computing. *Ad Hoc Netw*, p 101937
28. Noura H, Samhat AE, Harkous Y, Yahya TA (2015) Design and realization of a neural block cipher. In: 2015 International conference on applied research in computer science and engineering (IACR). Beirut, pp 1–6. <https://doi.org/10.1109/ARCSE.2015.7338131>
29. Rivest R, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. *Commun ACM* 21(2):120–126
30. Sharma I (2013) Fully homomorphic encryption scheme with symmetric keys. Rajasthan Technical University, Kota. University College of Engineering, Department of Computer Science and Engineering
31. Sylvester J (1851) On a remarkable discovery in the theory of canonical forms and of hyperdeterminants
32. Tong L, Wenbin C, Yi T, Hongyang Y (2018) A homomorphic network coding signature scheme for multiple sources and its application in IoT. *Secur Commun Netw*, 1–6. <https://doi.org/10.1155/2018/9641273>
33. van Dijk M, Gentry C, Halevi S, Vaikuntanathan V (2010) Fully homomorphic encryption over the integers. *EUROCRYPT* 2010 (LNCS) 6110:24–43
34. Vogel M (2010) An introduction to the theory of numbers, 6th edition by g.h. Hardy and e.m. Wright. *Contemp Phys* 51:283–283
35. Wagner D (2003) Cryptanalysis of an algebraic privacy homomorphism. *Inform Secur* 2851:234–239

36. Wang L, Li L, Li J, Li J, Gupta BB, Liu X (2019) Sensing of medical images with confidentially homomorphic aggregations. *IEEE Internet Things J* 6(2):1402–1409. <https://doi.org/10.1109/JIOT.2018.2844727>
37. Xiao L, Bastani O, Yen I-L (2012) An efficient homomorphic encryption protocol for Multi-user systems Citeseer. *IACR Cryptology ePrint Archive*, vol 2012, pp 193
38. Yang P, Gui X, An J, Tian F (2017) An efficient secret key homomorphic encryption used. *Image Process Serv Secur Commun Netw* 2017(Article ID 7695751):11
39. Zhang P, Jiang Y, Lin C, Fan Y, Shen X (2010) P-coding: secure network coding against eavesdropping attacks. *INFOCOM, 2010 Proceedings IEEE*, pp 1–9

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Khalil Hariss is a PhD student in computer science in Lebanese University, Beirut, Lebanon.



Hassan Noura is a research associate in the faculty of electrical and computer engineering of the American University of Beirut, Lebanon.



Abed Ellatif Samhat is a professor in the faculty of electrical and computer engineering of the Lebanese University, Beirut, Lebanon.