




# An Efficient FHE Scheme to Secure Cloud Computing

Khalil Hariss<sup>1,2</sup><sup>a</sup>, Abed Ellatif Samhat<sup>1</sup><sup>b</sup> and Maroun Chamoun<sup>2</sup><sup>c</sup>

<sup>1</sup>Faculty of Engineering - CRSI, Lebanese University, Hadat, Beirut, Lebanon

<sup>2</sup>Faculté D'Ingénierie - CIMTI, Université Saint Joseph, Mar Roukoz, Beirut, Lebanon

**Keywords:** Cloud Computing, Cloud Privacy, Fully Homomorphic Encryption, Domingo Ferrer, Learning with Error, Key Switching, Circuit Evaluation, Storage Overhead, Known Plain-text Attack, Polynomial Resultant.

**Abstract:** In this paper, we consider the privacy issue in cloud systems by using Homomorphic Encryption (HE) to provide secure computing at the cloud side. We use Domingo Ferrer (DF) homomorphic scheme to accomplish this task. Before implementing DF in a cloud scenario, we resolve its main problems. The two concerned problems are sensitivity to known plain-text attack and cipher-text dimension growth after homomorphic multiplication causing high storage overhead and reducing the scheme efficiency. DF is first made secure for cloud systems by making the scheme much more resistant to the concerned attack due to the change of the encryption procedure. Second, DF is made efficient for cloud computing by introducing a new technique, called Key Switching (KS). This technique reduces the high overhead by decreasing the extended cipher-text dimension obtained after a homomorphic multiplication and preserving the homomorphic behavior. While users' privacy at the Cloud side is preserved with HE, KS technique relies on publishing a matrix  $M$ . Different secret keys are encrypted within  $M$  based on the hardness of Learning With Error (LWE). A deep crypt-analysis and implementations under Python using SageMath Library are done in order to validate the efficiency of our proposal.

## 1 INTRODUCTION

Cloud computing is formed of an embedded system of different configurable computers that provide services over the internet with a low cost of management. Cloud systems rely on sharing different resources to give better opportunities for outsourcing of storage and computation. Users' privacy at the Cloud side becomes critical when computation over the encrypted stored data is required. In this case, they are obliged to reveal some of their secret parameters in order to perform arithmetic operations over their sensitive data after decrypting it, then re-encrypting. An appropriate solution for this problem is Homomorphic Encryption (HE) since it allows non trusted parties to perform computations over encrypted data. This new type of encryption is suitable for Cloud computing (Chauhan et al., 2015) and used in several real world applications such as medical application (Kocabas and Soyata, 2014), etc. A cryptographic scheme having an encryption function  $Enc$  is said to be homomorphic

if the evaluation of a circuit  $C$  satisfies the relation  $Enc(f(X)) = f(Enc(X))$ , where  $f$  is the function that performs the circuit  $C$  and  $X = [x_1, x_2, \dots, x_l]$  is a tuple of  $l$  input plain-texts.  $C$  is an electrical circuit that can be expressed in Boolean form. The function  $f$  should therefore have a polynomial form containing addition and multiplication operations. As a conclusion, building a Fully Homomorphic Encryption (FHE) scheme is done by satisfying the two following properties:

### 1. Addition


$$Enc_K(x_1 + x_2) = Enc_K(x_1) + Enc_K(x_2) \quad (1)$$


### 2. Multiplication


$$Enc_K(x_1 \times x_2) = Enc_K(x_1) \times Enc_K(x_2) \quad (2)$$

where  $Enc_K$  is the encryption function under a secret key  $K$  and  $(x_1, x_2)$  are two plain-texts.

The notion of HE first appeared with the RSA scheme in (Rivest et al., 1978). Several researchers worked afterwards on the challenge of creating a HE scheme that provides simultaneously efficiency and robustness. Josph Domimgo Ferrer published in

<sup>a</sup>  <https://orcid.org/0000-0001-9631-8467>

<sup>b</sup>  <https://orcid.org/0000-0002-1137-621X>

<sup>c</sup>  <https://orcid.org/0000-0002-1106-7409>

(Domingo-Ferrer, 2002) a HE scheme based on polynomial calculations. The most valued work was introduced by the IBM researcher Craig Gentry in (Gentry, 2009) where he introduced the first theoretical FHE scheme that supports unbounded number of circuit depth and his work consisted on lattice based cryptography. Following Gentry works, several algorithms were born such as the DGHV scheme (van Dijk et al., 2010), (Hariss et al., 2017) that works over the integers. DGHV suffers from high computational complexity, 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 (Coron et al., 2011). Crypt-analysis has also shown that DGHV is sensitive to Greatest Common Divisor (GACD) attack (Chen and Nguyen, 2012). BGV (Brakerski et al., 2012), (Hariss et al., 2017) is another FHE scheme that works over lattices where its computational complexity is also given by large poly-logarithmic factors. A wide state of art regarding HE is given in (Martins et al., 2017), (Aguilar-Melchor et al., 2013).

In this paper, we focus on Domingo Ferrer (DF) encryption scheme introduced in (Domingo-Ferrer, 2002) that suffers from two main vulnerabilities:

1. Sensitivity to known plain-text attack: the scheme is sensitive to this attack due to its algebraic structure as given in (Wagner, 2003).
2. Cipher Expansion: homomorphic multiplication is a polynomial multiplication that increases the cipher dimension exponentially after each operation. Thus causing a high storage overhead and reducing the scheme efficiency.

We propose to make DF suitable for Cloud systems first by enhancing its security level after modifying its encryption procedure to make it much more resistant against known plain-text attack. Second, we reduce the high overhead, by using a new technique called KS (Brakerski et al., 2012) based on publishing a new matrix  $M$ . Using this public matrix  $M$ , a new cipher  $c_{fresh}$  under a new secret key  $K_{fresh}$  can be obtained such that  $c_{fresh} = M \times c$  where  $dimension(c_{fresh}) < dimension(c)$  and  $Dec_{K_{fresh}}(c_{fresh}) = Dec_K(c)$  ( $c$  is a cipher with high storage overhead with respect to a secret key  $K$  with high storage overhead). Different secret parameters are encrypted within the public matrix  $M$  profiting from the hardness of Learning With Error (LWE) (Regev, 2009).

The rest of this paper is organized as follow: in section 2, the basic DF encryption scheme (Domingo-Ferrer, 2002) is explained, then we introduce the LWE problem given in (Regev, 2009). In section 3, the two main problems of DF are solved as given above.

Crypt-analysis is given in section 4 where leveraging the resistance of the scheme against known plain-text attack is validated, and the security level of KS is investigated. Implementations under Python using SAGEMath Library are given in section 5 in order to validate the performance of our proposal for Cloud systems. Finally conclusion and future works are listed in section 6.

## 2 PRELIMINARIES

### 2.1 DF Encryption Scheme

#### 2.1.1 Encryption Parameters

As stated in (Domingo-Ferrer, 2002), starting from a security parameter  $\lambda$ , different parameters are defined:

1. public modulus  $m \geq 10^{200}$  that have many small divisors.
2. public integer  $d > 2$  represents the cipher-texts dimension.
3. secret modulus  $m'$  should be a small divisor of the public modulus  $m$  ( $m = (m')^\lambda$ ).
4. secret key  $r \in Z_m$  invertible by the multiplicative law in the public ring  $Z_m$ .

#### 2.1.2 Encryption Procedure

Consider a plain-text  $a \in Z_{m'}$ , the encryption is given by two steps:

1. Decompose Function: The plain-text  $a$  is randomly decomposed into  $d$  elements  $(a_1, a_2, a_3, \dots, a_d) \in [Z_m]^d$  such that  $\sum_{i=1}^d a_i \pmod{m'} = a$ .
2. Encryption: An invertible secret key  $r$  is randomly picked from the public ring  $Z_m$ . The cipher-text  $\pi$  of the plain-text  $a$  is given by:  $\pi = [a_1 r, a_2 r^2, a_3 r^3, \dots, a_d r^d] \pmod{m}$ . Cipher-text can be written with a uni-variate polynomial form:  $\pi(t) = (a_1 r)t + (a_2 r^2)t^2 + (a_3 r^3)t^3 + \dots + (a_d r^d)t^d \pmod{m}$ .

#### 2.1.3 Decryption Procedure

Decryption at the classified level, is simply done by multiplying the  $i^{th}$  coordinate  $\pi(i)$  of  $\pi$  by the  $r^{-i} \pmod{m}$  to retrieve  $a_i \pmod{m}$  then we perform  $\sum_{i=1}^d a_i \pmod{m'}$  to retrieve  $a$ .

### 2.1.4 Homomorphic Properties

1. Addition: Using the uni-variate polynomial form  $\pi(t)$ , adding two cipher-texts in  $Z_m$  is simply adding two polynomials in  $Z_m[T]$ .
2. Multiplication: All terms are cross multiplied in  $Z_m$ , with  $d_1^{th}$  degree by  $d_2^{th}$  degree. The result is a polynomial of degree  $d = d_1 + d_2$ , finally terms of the same degree are added.

Starting from  $2^l$  cipher-texts of dimension  $d$  and after evaluating the multiplicative circuit of depth  $l$  given in Fig. 1. The cipher-text dimension becomes  $2^l(d - 1) + 1$ . Thus dimension grows exponentially with homomorphic multiplication. This problem will be solved in section 3 after introducing the KS technique. The security of KS is based on the hardness of *LWE* presented below.

## 2.2 Learning With Error (LWE)

### 2.2.1 LWE Problem

*LWE* is a machine learning problem introduced by Oded Regev in (Regev, 2009). In *LWE*, we need to learn the secret values of a vector  $s$  from a sample of the form  $(a_i, \langle s, a_i \rangle + e_i)$ , where  $a_i$  are uniformly chosen from  $Z_q^{(1,n)}$ ,  $e_i$  is a noise distribution over  $Z_q$ ,  $n$  is the dimension and  $q$  is the modulus.

In *LWE*, we define the Search Problem by:

Find  $s \in Z_q^{(1,n)}$  given  $p$  noisy random inner products:

$$\begin{aligned}
 a_1 &\leftarrow Z_q^{(1,n)} & b_1 &= \langle s, a_1 \rangle + e_1 \\
 a_2 &\leftarrow Z_q^{(1,n)} & b_2 &= \langle s, a_2 \rangle + e_2 \\
 \dots & \dots & \dots & \dots \\
 \dots & \dots & \dots & \dots \\
 \dots & \dots & \dots & \dots \\
 a_p &\leftarrow Z_q^{(1,n)} & b_p &= \langle s, a_p \rangle + e_p
 \end{aligned} \tag{3}$$

Where  $e_i \leftarrow \chi = \text{Gaussian Distribution over } Z_q$ .

### 2.2.2 LWE Matrix Form

Suppose that we have the matrix  $A =$

$$[a_1, a_2, a_3, \dots, a_p] = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{p1} \\ a_{12} & a_{22} & \dots & a_{p2} \\ \dots & \dots & \dots & \dots \\ a_{1n} & a_{2n} & \dots & a_{pn} \end{bmatrix} \text{ where}$$

$$a_i = [a_{i1}, a_{i2}, \dots, a_{in}]_{1 \leq i \leq p} \in Z_q^{(n,1)}.$$

$$\text{Let } s^t = [s_1, s_2, \dots, s_n] \in Z_q^{(1,n)}, \quad e^t = [e_1, e_2, \dots, e_p] \in Z_q^{(1,p)}.$$

The *LWE* problem given in Eq. 3 can be written based on the following matrix form:

$$b^t = s^t A + e^t \in Z_q^{(1,p)} \tag{4}$$

Eq. 4 can be seen as a lattice equation, where  $s^t A$  forms the lattice points. As a lattice based cryptography,  $e^t$  can be considered as a plain-text encrypted under a secret key  $s^t$  to obtain its corresponding cipher-text  $b^t$ . Decryption is a Closest Vector Problem (*CVP*) that consists of finding the closest lattice point to  $b^t$  which can simply be resolved only when having the secret key  $s^t$ . Starting from a security parameter  $\lambda$ , the security of *LWE* resides in taking the dimension  $n \sim \text{poly}(\lambda)$  and the modulus  $q \sim \text{poly}(n)$  and using  $\text{poly}(n)$  equations as given in (Brakerski et al., 2012), (Regev, 2009).

## 3 ENHANCING DF ENCRYPTION SCHEME

In this section, the two main problems of the DF scheme (know plain-text attack and cipher-text expansion) are investigated. First the security level of the basic scheme given in (Domingo-Ferrer, 2002) is enhanced by changing the encryption procedure, then a new technique called KS is introduced. KS reduces the storage overhead of the cipher-text after each homomorphic multiplication without altering the homomorphic behavior of the scheme or changing the primitive plain-text.

### 3.1 Enhancing The Security Level

#### 3.1.1 New Encryption Procedure

Starting from a plain-text  $a \in Z_m'$ , the first step is kept the same by randomly decomposing the plain-text  $a$  into  $d$  elements in  $Z_m$  such that  $\sum_{i=1}^d a_i \pmod{m'} = a$ .

The second step is modified by using a vector  $s$  of  $d$  secret invertible keys as follow  $s = [r_1, r_2, \dots, r_d]$  instead of using  $d$  powers of one invertible secret key  $r$  (i.e.  $[r, r^2, r^3, \dots, r^d]$ ). The cipher-text  $\pi$  will be  $\pi = [a_1 r_1, a_2 r_2, a_3 r_3, \dots, a_d r_d] \pmod{m}$ .

$\pi$  can be written in a multi-variate polynomial form:  $\pi(t_1, t_2, t_3, \dots, t_d) = (a_1 r_1) t_1 + (a_2 r_2) t_2 + (a_3 r_3) t_3 + \dots + (a_d r_d) t_d \pmod{m}$ .

We refer to Basic DF, the primitive scheme explained in (Domingo-Ferrer, 2002), while Modified DF refers to the new encryption procedure.

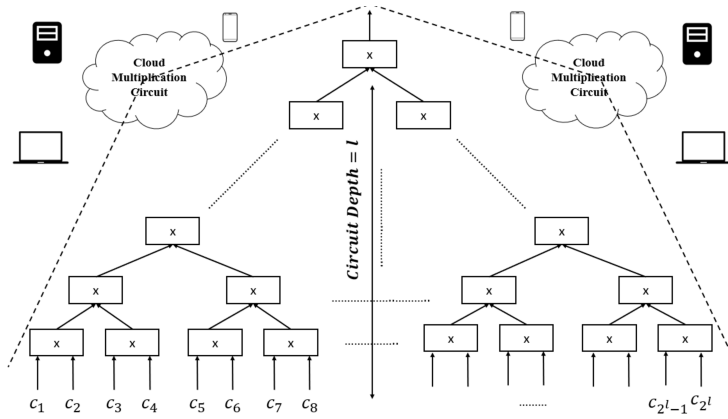


Figure 1: Cloud Multiplication Circuit.

**3.1.2 Decryption Procedure**

Having the secret parameters  $(m', s = [r_1, r_2, \dots, r_d])$ , decryption at the classified level can be done by multiplying  $\pi(i)$  by  $r_i^{-1}$  to obtain  $a_i \pmod{m}$ . Then we perform  $\sum_{i=1}^d a_i \pmod{m'}$ .

**3.1.3 Homomorphic Properties**

Starting from two plain-texts  $a_1, a_2$ , the two respective cipher-texts using Modified DF for  $a_1$  and  $a_2$  are:

$$\pi_1 = [a_1^{(1)} r_1, a_1^{(2)} r_2, \dots, a_1^{(d-1)} r_{d-1}, a_1^{(d)} r_d] \pmod{m}$$

$$\pi_2 = [a_2^{(1)} r_1, a_2^{(2)} r_2, \dots, a_2^{(d-1)} r_{d-1}, a_2^{(d)} r_d] \pmod{m}$$

1. Addition: Using multi-variate polynomial addition,  $\pi_{1+2} = \pi_1 + \pi_2$  can be written as:

$$[(a_1^{(1)} + a_2^{(1)})r_1, (a_1^{(2)} + a_2^{(2)})r_2, \dots, (a_1^{(d-1)} + a_2^{(d-1)})r_{d-1}, (a_1^{(d)} + a_2^{(d)})r_d] \pmod{m}$$

Using  $s^{-1} = [r_1^{-1}, r_2^{-1}, \dots, r_d^{-1}]$ , we can demonstrate that  $Dec_s(\pi_1 + \pi_2) = Dec_s(\pi_1) + Dec_s(\pi_2)$  and the modified scheme is additive homomorphic.

2. Multiplication: Using multi-variate polynomial multiplication,  $\pi_{1 \times 2} = \pi_1 \times \pi_2$  can be written as:

$$[a_1^{(1)} a_2^{(1)} r_1^2, a_1^{(2)} a_2^{(2)} r_2^2, \dots, a_1^{(d-1)} a_2^{(d-1)} r_{d-1}^2, a_1^{(d)} a_2^{(d)} r_d^2, \\ (a_1^{(1)} a_2^{(2)} + a_2^{(1)} a_1^{(2)})r_1 r_2, (a_1^{(1)} a_2^{(3)} + a_2^{(1)} a_1^{(3)})r_1 r_3, \dots, (a_1^{(1)} a_2^{(d)} + a_2^{(1)} a_1^{(d)})r_1 r_d, \\ (a_1^{(2)} a_2^{(3)} + a_2^{(2)} a_1^{(3)})r_2 r_3, \dots, (a_1^{(2)} a_2^{(d)} + a_2^{(2)} a_1^{(d)})r_2 r_d, \dots, (a_1^{(d-1)} a_2^{(d)} + a_2^{(d-1)} a_1^{(d)})r_{d-1} r_d] \pmod{m}$$

We can demonstrate that the cipher-text dimension after homomorphic multiplication with Modified DF will be  $N = C_d^2 + d = \frac{d!}{(d-2)!2!} + d$ .

N will be reduced to  $d$  after introducing KS technique. To decrypt  $\pi_{1 \times 2}$ , an extended version of the secret key

$s^* = [r_1^2, r_2^2, \dots, r_{d-1}^2, r_d^2, r_1 r_2, r_1 r_3, \dots, r_1 r_d, r_2 r_3, \dots, r_2 r_d, \dots, r_{d-1} r_d]$  of dimension  $N$  is used. The decryption of  $\pi_{1 \times 2}$  is done applying the following inner product:

$$\langle s_{inverse}^*, \pi_{1 \times 2} \rangle \pmod{m'} \quad (5)$$

where  $s_{inverse}^* = [r^{-1} \pmod{m}, r \in s^*]$ .

Based on Eq.5, we can validate that  $Dec_s(\pi_1 \times \pi_2) = Dec_s(\pi_1) \times Dec_s(\pi_2)$  and the new encryption procedure is multiplicative homomorphic.

**Remark 1.** Eq.5 can be written in a matrix form: plain-text =  $s_{inverse}^* \pi_{1,2} \pmod{m'}$ , where  $s_{inverse}^* \in Z_m^{(1,N)}$  and  $\pi_{1,2} \in Z_m^{(N,1)}$ .

**3.2 KS Technique**

As it is mentioned previously, with homomorphic multiplication the cipher-text dimension in both cases (basic or modified) will expand into a bigger value than the initial cipher-text dimension  $d$ . The main concept of KS is that having an extended cipher-text  $c^*$  with dimension  $N$ , with respect to an extended secret key  $s^*$  also with dimension  $N$ , a new cipher-text  $c'$  of dimension  $d$  with respect to a new secret key  $s'$  with dimension  $d$  should be calculated verifying this equation:

$$Dec_{s^*}(c^*) = Dec_{s'}(c') \quad (6)$$

The main idea is to publish an encryption matrix  $M \in Z_m^{(d,N)}$  defined by:

$$M(s^* \rightarrow s') \text{ such that } c' = M c^* \quad (7)$$

(Dimension :  $(d, 1) = (d, N) \times (N, 1)$ )

The new secret key is  $s' = [r'_1, r'_2, \dots, r'_{d-1}, r'_d]$ , given that  $s_{inverse}' = [r^{-1} \pmod{m}, r \in s']$  and  $t' =$



### 4.1.3 Oracle Model

Starting from  $M = 2 \times poly(\lambda) \times (d + 1)$  couples of plain-text  $a_i$  and its corresponding cipher-text  $C_i$ .

1. Building the Matrix  $W$ :

From the  $M$  couples of plain-text/cipher-text, the system of Eq.10 is built by picking  $(d + 1)$  couples.

$$C_{i+k}^1 r_1^{-1} + C_{i+k}^2 r_2^{-1} + \dots + C_{i+k}^d r_d^{-1} = a_{i+k} \pmod{m'} \quad (10)$$

where  $0 \leq k \leq d$ .

Adding to the system of Eq. 10 an extra parameter  $r_{d+1} = 1$ , gives birth to a new system formed of  $(d + 1)$  equations with  $(d + 1)$  unknowns given in Eq.11.

$$C_{i+k}^1 r_1^{-1} + C_{i+k}^2 r_2^{-1} + \dots + C_{i+k}^d r_d^{-1} - a_{i+k} r_{d+1}^{-1} = 0 \pmod{m'} \quad (11)$$

where  $0 \leq k \leq d$ .

It is very sure that the  $(d + 1)$  multi-variate homogeneous polynomials of Eq.11 share a common root  $(r_1^{-1}, r_2^{-1}, r_3^{-1}, \dots, r_d^{-1}, r_{d+1}^{-1} = 1)$ . The resultant of these  $(d + 1)$  polynomials should be a multiple of  $m'$  since all numbers are in the ring  $Z_{m'}$ . The resultant matrix  $W_i$  is built as listed in section 4.1.1 for multi-variate case. The  $k^{th}$  row of matrix  $W_i$  is given by  $[C_{i+k}^1, C_{i+k}^2, C_{i+k}^3, \dots, C_{i+k}^d, -a_{i+k}]$  for  $0 \leq k \leq d$ . Its determinant should be a multiple of the secret modulus  $m'$ .

2. Recovering the Secret Modulus  $m'$ :

The procedure listed in the previous section is repeated  $2 \times poly(\lambda)$  times to build  $2 \times poly(\lambda)$  matrices  $W_i$ .

Let  $T = [Det(W_i), i \in \{1, 2, 3, \dots, 2 \times poly(\lambda)\}]$  formed of  $2 \times poly(\lambda)$  multiples of  $m'$ .

Let  $Y = [gcd(T_i, T_{i+1}), i \in \{1, 3, 5, \dots, 2 \times poly(\lambda) - 1\}]$ .

Depending on the well known fact given in (Vogel, 2010) that 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  $\{T_j, T_k\} \subset T$  having  $gcd(T_j, T_k) = m'$  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  $Y$ .

3. Recovering the Equivalent Secret Key:

After revealing the secret modulus  $m'$ , the equivalent secret key to be recovered is  $[r_1^\diamond, r_2^\diamond, r_3^\diamond, \dots, r_d^\diamond]$  where  $r_i^\diamond = r_i \pmod{m'}$ . To do this, from the  $M$  couples of plain-text/cipher-text,  $d$  couples are picked to form the linear system of Eq.12.

$$C_{i+k}^1 r_1^{-1} + C_{i+k}^2 r_2^{-1} + \dots + C_{i+k}^d r_d^{-1} = a_{i+k} \pmod{m'} \quad (12)$$

where  $0 \leq k \leq d - 1$ .

Finding the equivalent secret key can be achieved by solving this linear system. This is only possible when its relative matrix is invertible in the ring  $Z_{m'}$ . Based on (Brent and McKay, 1987), 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 system of Eq. 12 can be resolved using Gaussian elimination with  $O(d^3(\log(m'))^2)$ .

As a conclusion, Modified DF is secure against this attack only when  $M$ , the number of couples plain-text/cipher-text, is lower than  $d$ .

## 4.2 Comparison between Basic and Modified DF in Terms of Security Level

In this section, we will show that the Modified DF is more robust than Basic DF.

1. Recovering  $m'$ :

Recovering  $m'$  for Basic DF is based on univariate polynomials. Sylvester matrix given in section 4.1.1 needs 2 polynomials instead of  $(d + 1)$ . To recover the secret key  $m'$  with a probability close to  $(1 - (1 - \frac{6}{\pi^2})^{poly(\lambda)})$ , we need to start the attack with  $2 \times poly(\lambda) \times 2$  couples of plain-text/cipher-text in the basic version instead of  $2 \times poly(\lambda) \times (d + 1)$  in the modified one. Hence recovering  $m'$  is made  $O(d)$  much more harder than the basic one.

2. Recovering the Equivalent Secret Key:

In the case of Basic DF, the cipher-text of a

plain-text  $a$  with a secret key  $r$  is given by  $C = [a_1r, a_2r^2, a_3r^3, \dots, a_dr^d]$ .

Supposing that  $z_h = r^h$  for  $1 \leq h \leq d$ , a linear system similar to the one given in Eq.12 is built, but only one unknown  $z_1 = r$  should be calculated instead of  $d$  unknowns  $[r_1, r_2, r_3, \dots, r_d]$  in Modified DF.

### 4.3 KS Security

In this section the hardness of *LWE* is investigated because KS security depends on it.

Different secret keys  $s^*$ ,  $s' = [r'_1 | (t')^{-1}]$  and  $m'$  are encrypted within the public matrix  $M = \begin{bmatrix} b \\ A \end{bmatrix}$ , where  $b =$

$(-t'A + m'e + s_{inverse}^*)r'_1 \in Z_m^{(1,N)}$  and  $A = [A_{i,j}, 1 \leq i \leq d-1, 1 \leq j \leq N]$ .

We recall that  $Z_m$  is the public ring and  $N = C_d^2 + d$ .

To reveal secret and random values in the public matrix  $M$ , an attacker can build from the matrix  $b$  a linear system of  $N$  equations having  $(2d + N + 1)$  unknowns. This system have infinite solutions which makes recovering the secret keys infeasible especially when taking high values and respecting the security conditions of *LWE* given in section 2.

In addition within the public matrix  $b$ ,  $m'e + s_{inverse}^*$  is considered as a plain-text encrypted under the secret key  $s'$ . Decryption is a hard lattice problem (*CVP*) where resolving it can be done only when knowing the secret  $s'$  (Brakerski et al., 2012), (Regev, 2009).

## 5 EXPERIMENTATIONS

In this section, our proposal is validated by doing different implementations of DF.

All implementations are done under Python with SAGEMath Library using a machine having the following specification: CPU-Intel Xeon, E5-2630, 2.40 GHZ, 8 CORES, 128 GB RAM.

Starting from a security parameter  $\lambda = 90$ , different parameters are taken as follows: dimension  $d$  is varied from 25 to 100 with step equal to 25, secret modulus  $m' = 336$ , public modulus  $m = (m')^\lambda = (336)^{90} > 10^{200}$ .

### 5.1 First Experimentation

In this experimentation, two different plain-texts  $a_1, a_2$  are taken from the plain-texts secret ring  $Z_{336}$ , with two respective cipher-texts  $C_1, C_2$  obtained by applying Modified DF. Dimension  $d$  is varied from 25 to 100 as given above. By this implementation, the mean

encryption time for 100 plain-texts, the mean decryption time for 100 cipher-texts, generation time of one public matrix  $M$  are calculated. Finally, we calculate  $C_{mult} = C_1 \times C_2$  and using KS technique a new fresh cipher-text  $C' = M \times C_{mult}$  is obtained, where  $Dimension(C') < Dimension(C_{mult})$  provided by the mean execution time of KS for 100 iterations. In Tab.1, we show dimension variation for different matrices in function of the public parameter  $d$ .

Table 1: Matrix and Cipher-texts Dimensions.

$d$	Dimension	$M$	$C_{mult}$	$C'$
25		25x325	325	25
50		50x1275	1275	50
75		50x2850	2850	75
100		50x5050	5050	100

Different execution times in function of the public parameter  $d$  are given in Tab. 2. Different results showed acceptable execution times for encryption, decryption and KS technique. Execution time increase with dimension  $d$  growth is already expected while providing a higher level of security. It is true that for different values of  $d$  in Tab. 2, matrix  $M$  generation is taking the highest value, but it is generated only once at the user side. User will execute encryption/decryption respectively for each couple of plain-text/cipher-text and KS is performed by the Cloud after each homomorphic multiplication.

### 5.2 Second Experimentation

In the second experimentation, the evaluation procedure of the Cloud circuit given in Fig. 1 is done using three different implementations:

1. Implementation 1: Basic DF Without KS.
2. Implementation 2: Basic DF With KS.
3. Implementation 3: Modified DF With KS.

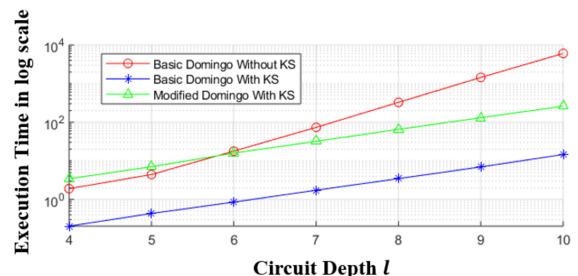


Figure 2: Execution Time In Function of Cloud Circuit Depth  $l$ .

Table 2: Execution times in function of  $d$ .

$d$ \ Procedures	Mean Enc. Time	Mean Dec. Time	Matrix $M$ Gen. Time	Mean KS time
25	0.00046307 s	0.00018298 s	0.136013 s	0.01836656 s
50	0.0009448 s	0.00037313 s	0.740623 s	0.15121927 s
75	0.00128236 s	0.000484769 s	2.084263 s	0.56401619 s
100	0.00175928 s	0.000723319 s	5.392393 s	1.27906641 s

Choosing  $d = 50$ , a comparison between different implementations is done in terms of execution time and storage overhead in function of circuit depth  $l$ . In Fig.2, different execution times for the three implementations are drawn in function of circuit depth  $l$  and given in log scale just for making results much clearer. For circuit depth  $l$  higher than 6, which is the complex circuits case faced in real world applications, Basic DF Without KS is taking the highest execution time, afterwards comes Modified DF With KS, and the lower execution time is for Basic DF With KS (for example in linear scale, with  $l = 10$  and  $l = 7$  respectively, Basic DF Without KS took 6046.26 seconds respectively 73.619958 seconds, while Modified DF With KS took 261.21 seconds respectively 31.993578 seconds and Basic DF With KS took 14.57 seconds respectively 1.733248 seconds). This result is interpreted by the fact that for a circuit with high depth, Basic DF increases the storage overhead of ciphertexts due to homomorphic multiplications.

Starting from  $d = 50$ , cipher dimension grows exponentially in function of circuit depth  $l$  in the case of Basic DF Without KS (for example for  $l = 4$  Cipher Dimension becomes 785 and for  $l = 10$  Cipher Dimension becomes 50177), while it remains 50 for the two others cases while using KS.

The new KS technique increases the efficiency of the scheme by reducing the storage overhead of ciphertexts. The difference in execution time between Basic DF with KS and Modified DF With KS is expected since the encryption is made more complicated to leverage the security level.

## 6 CONCLUSION AND FUTURE WORK

In this paper, we modified DF in order to be suitable for Cloud applications by resolving its two main problems: sensitivity to known plain-text attack and cipher expansion after homomorphic multiplication. The first problem was resolved by changing the encryption procedure. Crypt-analysis has shown that the concerned attack becomes harder with the modified version.

The second problem was treated by building a new technique called KS that reduces the cipher-text dimension after each homomorphic multiplication. KS is based on publishing a matrix  $M$  where different secret parameters are encrypted within it based on the hardness of *LWE*. Crypt-analysis has shown no secret parameters leakage at the Cloud side.

Different implementations have validated our work since KS improved the behavior of the scheme in terms of execution time and storage overhead. Crypt-analysis of DF has shown that the scheme (Modified or Basic version) is secure against known plain-text attack with high values of dimension  $d$  and the public modulus  $m$ . Cloud systems can support these security requirements since they are formed of big data centers having intensive computation power and storing capabilities.

Future work will consider applying the Modified DF under *LWE* in a secure Cloud computing scenario for a medical health care and a university applications.

## ACKNOWLEDGEMENTS

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

## REFERENCES

Aguilar-Melchor, C., Fau, S., Fontaine, C., Gogniat, G., and Sirdey, R. (2013). Recent advances in homomorphic encryption: A possible future for signal processing in the encrypted domain. *IEEE Signal Processing Magazine*, 30(2):108–117.

Allgower, E. L., Georg, K., and Miranda, R. (1992). The method of resultants for computing real solutions of polynomial systems. *SIAM J. Numer. Anal.*, 29(3):831–844.

Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (2012). (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ITCS '12*, pages 309–325, New York, NY, USA. ACM.

Brent, R. P. and McKay, B. D. (1987). Determinants and ranks of random matrices over  $\mathbb{Z}_m$ . *Discrete Mathematics*, 66(1):35 – 49.



- Chauhan, K. K., Sanger, A. K. S., and Verma, A. (2015). Homomorphic encryption for data security in cloud computing. In *2015 International Conference on Information Technology (ICIT)*, pages 206–209.
- Chen, Y. and Nguyen, P. Q. (2012). Faster Algorithms for Approximate Common Divisors: Breaking Fully-Homomorphic-Encryption Challenges over the Integers. In Pointcheval, D. and Johansson, T., editors, *EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 502–519, Cambridge, United Kingdom. IACR, Springer.
- Coron, J.-S., Mandal, A., Naccache, D., and Tibouchi, M. (2011). Fully homomorphic encryption over the integers with shorter public keys. In Rogaway, P., editor, *Advances in Cryptology – CRYPTO 2011*, pages 487–504, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Domingo-Ferrer, J. (2002). A provably secure additive and multiplicative privacy homomorphism. In *Proceedings of the 5th International Conference on Information Security, ISC '02*, pages 471–483, London, UK, UK. Springer-Verlag.
- Gentry, C. (2009). *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford, CA, USA. AAI3382729.
- Hariss, K., Chamoun, M., and Samhat, A. E. (2017). On dghv and bgv fully homomorphic encryption schemes. In *2017 1st Cyber Security in Networking Conference (CSNet)*, pages 1–9.
- Kocabas, O. and Soyata, T. (2014). *Medical Data Analytics in the Cloud Using Homomorphic Encryption*, pages 471–488.
- Martins, P., Sousa, L., and Mariano, A. (2017). A survey on fully homomorphic encryption: An engineering perspective. *ACM Computing Surveys*, 50:1–33.
- Regev, O. (2009). On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40.
- Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126.
- Sylvester, J. (1851). *On a Remarkable Discovery in the Theory of Canonical Forms and of Hyperdeterminants*.
- van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V. (2010). Fully homomorphic encryption over the integers. In Gilbert, H., editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 24–43, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Vogel, M. (2010). An introduction to the theory of numbers, 6th edition, by g.h. hardy and e.m. wright. *Contemporary Physics - CONTEMP PHYS*, 51:283–283.
- Wagner, D. (2003). Cryptanalysis of an algebraic privacy homomorphism. In Boyd, C. and Mao, W., editors, *Information Security*, pages 234–239, Berlin, Heidelberg. Springer Berlin Heidelberg.