Multimedia Tools and Applications (2022) 81:14403–14449
https://doi.org/10.1007/s11042-022-12043-7

Towards a fully homomorphic symmetric cipher scheme resistant to plain-text/cipher-text attacks

Khalil Hariss[1,2] · **Hassan Noura**[3]

Received: 8 January 2021 / Revised: 22 April 2021 / Accepted: 3 January 2022 /
Published online: 25 February 2022

Abstract

Users' privacy becomes nowadays an important need and a big challenge for a lot of enterprises and service providers especially after adopting the cloud migration strategy. Thus, Homomorphic Encryption (HE) came as a novel cryptographic approach that enables users' privacy at the cloud side by allowing computation over encrypted data. Existing HE schemes are based on either symmetric or asymmetric encryption algorithms. While asymmetric HE schemes provide the high and the required level of security, they suffer from high computational complexity and high storage overhead making a big majority of them not practical for real world applications. On the other hand, symmetric schemes assure the required efficiency, but they are vulnerable to attacks and especially the known plain-text/cipher-text attacks making their usage limited in practical implementation. The main objective of this paper is to design a new symmetric HE variant that provides the desired level of efficiency in implementation and the immunity against data breaches especially the known plain-text/cipher-text attacks. The proposed scheme, named Homomorphic Hybrid Symmetric Encryption Scheme (HHSES), which is based on combining the homomorphic behavior of two well-known symmetric encryption schemes that are the MORE (Matrix Operation for Randomization and Encryption) approach and the Domingo Ferrer (DF) scheme. The performance analysis of HHSES confirms its efficiency for real-world applications in comparison with a big variety of existing and well known symmetric and asymmetric schemes. A main drawback of HHSES is the cipher-text dimension expansion after the homomorphic multiplication since homomorphic operations are restricted to polynomial operations over the matrices. Therefore, to fix this issue, we propose a specific Key Switching (KS) technique after the homomorphic multiplication that reduces the cipher-texts' dimension without altering its homomorphic behavior and the primitive classified data. Security analysis of the new scheme also verifies its immunity against different types of attacks and especially the known plain-text/cipher-text attacks. Another important contribution in this work is the optimization of the HHSES encryption and decryption procedures by making them parallelized using the Chinese Remainder Theorem (CRT). The implementation results have shown that the proposed optimization technique reduces the execution time of the HHSE encryption and decryption algorithms with a ratio close to $\frac{1}{2}$. To the best of our

✉ Hassan Noura
hassan.noura@univ-fcomte.fr

Extended author information available on the last page of the article.

knowledge, HHSES is the first symmetric HE scheme immune against the known/chosen plain-text/cipher-text attacks.

Keywords Symmetric homomorphic cipher scheme · Known plain-text/cipher-text attack · Domingo ferrer · MORE approach · PORE approach · RSA ring · Cloud computing

1 Introduction

Cloud computing ensures different services through the Internet. These resources include tools and applications like data storage, servers, databases, networking, and software. This technology presents a big variety of benefits for users such as scalabilty, cost reduction, disaster recovery, virtualization, and opportunities for outsourcing of storage and computation. Despite all of the advantages listed previously, users' privacy in a cloud system is a major concern since in some critical cases, performing operations on encrypted data is mandatory. Users are then obliged to expose some of their secret parameters and decryption policies to the cloud. Thus, computing over the encrypted data becomes a need especially when users' life has become trapped within the cloud. While cryptography [15] and steganography [23–25] are two important techniques used for preserving secret data confidentiality and for communicating via secret messages, traditional algorithms related to these two techniques are still limited in a cloud computing scenario.

HE is a modern cryptographic research topic, different from traditional cryptographic algorithms, that allows non-trusted parties to compute over encrypted data. This new type of encryption is crucial in the modern world [37] especially with cloud-based applications. An attractive field of exploring and benefiting from HE advantages is bio-metric data analysis such as DNA analysis [17], Palmprint [14], fingerprint [18] and eye-print [33] authentication. An illustration of HE implementation in such applications is shown in Fig. 1. In the presented figure, data is encrypted and stored in the cloud using HE algorithms, and different parties can send encrypted requests to the cloud. The latter performs operations over encrypted data and ships back encrypted answers to the trusted parties for decryption. Adopting a secure and efficient HE algorithm as a practical solution for bio-metric data privacy and operations forms an important variant in comparison with some existing techniques such as PalmHash Code and PalmPhasor Code [22] and Dual-key-binding cancelable palmprint cryptosystem [21]. Especially that some of the latter solutions such as PalmHash Code suffer from vulnerability to statistical attacks [20].

HE was first known as Privacy Homomorphism (PH) and was introduced in 1978 by Rivest, Shamir and Adleman with the RSA scheme [30]. At that time, RSA allowed only multiplication over encrypted data. Nowadays, existing HE schemes allow different types of computations over encrypted data. While a wide range of the state of art (including symmetric and asymmetric homomorphic variants) is given in [1, 11], a brief description of some well known schemes is shown in Table 1.

The stepping stone in designing an asymmetric HE scheme was achieved by the IBM researcher Craig Gentry [8] in 2009. Gentry used ideal lattices to design the first Fully Homomorphic Encryption (FHE) scheme that allows non-bounded circuit evaluation. Afterwards, several encryption schemes based on asymmetric algorithms were proposed such as the DGHV (Dijk, Gentry, Halevi and Vaikuntanathan) scheme [38] and the BGV scheme [3]. DGHV is a FHE asymmetric scheme that works over the integers. DGHV supports a limited number of homomorphic operations due to the increase in noise after computing the

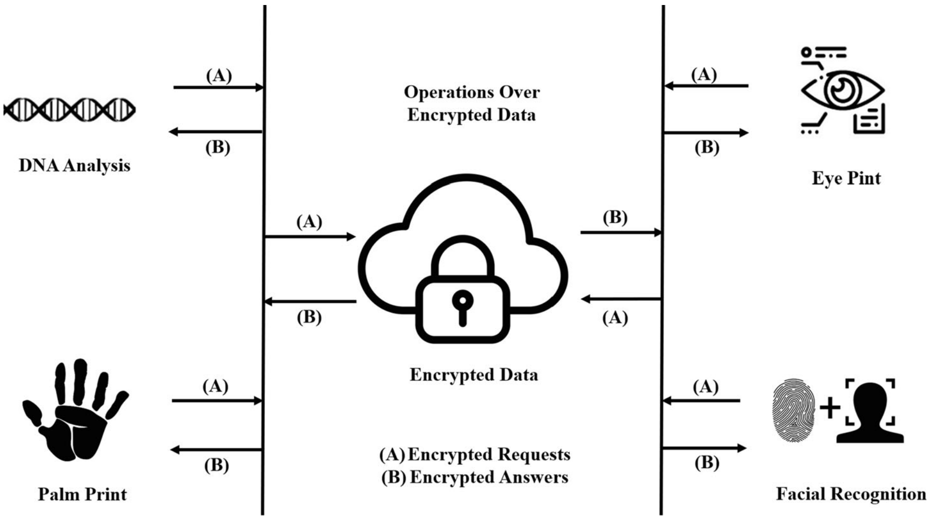


Fig. 1 Possible HE implementation in real world applications

cipher-text. Bootstrapping is a refresh mechanism introduced in the literature that decreases the noise level after each homomorphic operation while preserving the primitive plain-text. With bootstrapping, DGHV can evaluate circuits with non-bounded depth. BGV is another asymmetric FHE scheme that works over the lattices. BGV also suffers from high noise levels after performing homomorphic operations. Modulus Switching (MS) is a new technique introduced in the literature that extends the circuit evaluation depth by reducing the noise level after homomorphic operations and preserving the original plain-text. Despite of the high level of security presented by the DGHV and the BGV schemes, they suffer from high computation complexity and communication overhead. For example as given in [6], the size of the DGHV public key can attain 2.3 GB and with an optimized implementation on a high end workstation, key generation takes 2.2 hours, one bit encryption takes 3 minutes, and cipher-text refresh mechanism takes 30 minutes. Similarly, BGV is also based on complex lattice calculations.

In this paper, we will show that the proposed solution presents a new approach of symmetric HE algorithm that can require less performance overhead, and is more robustness compared to symmetric FHE schemes since it is based on different cryptographic concepts. Therefore, the proposed solution can be considered as a good candidate for limited real-time applications or limited devices. In general different asymmetric HE schemes provide a high level of security, meanwhile their performance is still far from being practical despite of all the accelerated implementations present in the literature such as [10, 39].

Other researchers took a different direction by investigating the design of symmetric encryption schemes using linear algebraic operations. In 2002, Joseph Domingo Ferrer published a symmetric polynomial based HE scheme known as Domingo Ferrer scheme (DF) [7]. In 2003, David Wagner analyzed the vulnerabilities of DF and drove a known plain-text/cipher-attack using polynomial resultant and Gaussian elimination [35]. Matrix Operation for Randomization and Encryption (MORE) is another symmetric HE scheme [19, 36] based on linear matrix operations. While the security of the MORE approach is based on the hardness of Rabin's crypto-system and the factorization of the product of two large primes, a known plain-text/cipher-text attack can be executed by

Table 1 Description of existing HE schemes

Scheme	Type	Structure	Homomorphic behavior	Security
RSA [30]	Asymmetric	Integers' Ring	Multiplicative	Hardness of Factoring Large Integers ($N = p \times q$)
Paillier [28]	Asymmetric	Integers' Ring	Additive	Hardness of Composite Residuosity (DCR) Problem
DGHV [38]	Asymmetric	Integers' Ring	Somewhat Homomorphic	Hardness of General Approximate Common Divisor (GACD)
BGV [3]	Asymmetric	Lattices' Ring	Somewhat Homomorphic	Hardness of Learning With Errors (LWE)
MORE [19, 36]	Symmetric	Matrix Based	Additive and Multiplicative	Known Plain-text/Cipher-text Attack $\rightarrow$ Matrix Structure
PORE [19]	Symmetric	Polynomial's Ring	Additive and Multiplicative	Known Plain-text/Cipher-text Attack $\rightarrow$ Linear Structure
DF [7]	Symmetric	Polynomial's Ring	Additive and Multiplicative	Known Plain-text/Cipher-text Attack $\rightarrow$ Polynomial Resultants
NOHE [13]	Symmetric	Bit's Ring	Additive and Multiplicative	Known Plain-text/Cipher-text Attack $\rightarrow$ No Avalanche Effect

calculating the eigen-vectors. The PORE (Polynomial Operation for Randomization and Encryption) approach [19] is also a symmetric HE scheme based on polynomial operation over the cipher-texts. The latter scheme is sensitive to known plain-text/cipher-text attack where secret key can be revealed while knowing only one couple of plain-text and its correspondent cipher-text. Not Operation for Homomorphic Encryption (NOHE) [13], is another symmetric HE scheme based on the homomorphic behavior of the logic NOT gate. As stated in [13], a known plain-text/cipher-text attack can be performed on the NOHE scheme due to the lack of the avalanche effect.

As far as we know, we have built the first symmetric encryption scheme that can resist several types of attacks including the known plain-text/cipher-text attacks. The proposed scheme is realized by mixing the homomorphic behavior of both, the MORE and the DF approaches. The resultant scheme is referred to as the Homomorphic Hybrid Symmetric Encryption Scheme (HHSES). Crypt-analysis has shown that HHSES is resistant to different types of attacks including the known plain-text/cipher-text attacks. Besides, high level of efficiency is validated by implementation.

The rest of this paper is organized as follows: Section 2 introduces the HE concept and homomorphic properties and presents the state of art and some of the very well-known HE schemes for both symmetric (MORE, PORE and DF) and asymmetric (BGV) approaches. Section 3 introduces the proposed scheme (HHSES) that combines the homomorphic behavior of MORE and DF approaches. Security analysis of the HHSES is given in Section 4, where different security tests validate the scheme's immunity against different types of attacks such as statistical attacks, and related key attacks. Security tests also verify the fulfilment of both uniformity and independence properties and the avalanche effect. Finally a theoretical crypt-analysis shows that the HHSES is immune against known plain-text/cipher-text attacks even with the lowest encryption parameters. Implementations under Python using SageMath library, and performance analysis of the new scheme are provided in Section 5 along with a comparison with symmetric (MORE, PORE and DF) and asymmetric (BGV) schemes. The correctness of the CRT optimization for both encryption and decryption procedures is validated by implementation in Section 6. Conclusion and future work are listed in Section 7.

2 State of art and existing schemes

In this section first HE concept is presented, then some of the well known symmetric (MORE, PORE and DF) and asymmetric (BGV) HE schemes are described.

2.1 Homomorphic Encryption Concept

An encryption scheme (β) having an encryption function (Enc) under a secret key (K) is said to be homomorphic, if for any circuit C that runs a function f, the evaluation procedure satisfies the following relation [9]:

$$Enc_K(f(X)) = f(Enc_K(X)) \tag{1}$$

Where $X = [x_1, x_2, \cdots, x_l]$ is a tuple of l input plain-texts.
C is an electrical circuit written as a Boolean function. Boolean functions have a polynomial form that consists of a set of addition and multiplication gates as presented in Fig. 2.

Thus, building a HE scheme that evaluates a set of cipher-texts over any given electrical circuit is accomplished by applying these two basic properties defined by:

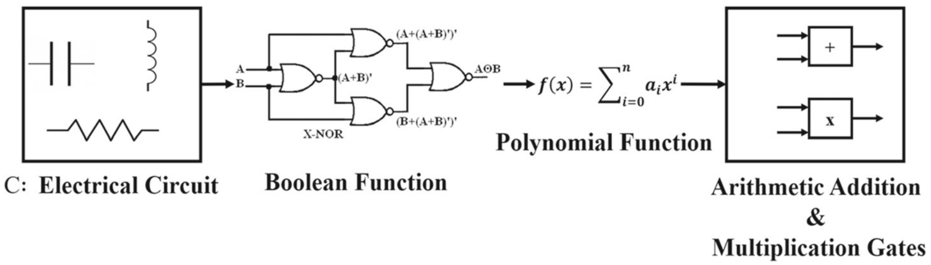


Fig. 2 Designing HE cipher scheme from electrical circuit to achieve homomorphic arithmetic operations: addition and multiplication

1. Addition:

$$Enc_K(x_1 + x_2 \quad mod(M)) = Enc_K(x_1) + Enc_K(x_2) \quad mod(N) \tag{2}$$

2. Multiplication:

$$Enc_K(x_1 \times x_2 \quad mod(M)) = Enc_K(x_1) \times Enc_K(x_2) \quad mod(N) \tag{3}$$

Where x_1 and x_2 are two inputs plain-texts, $\mathbb{Z}_{\mathbb{N}}$ represents the plain-texts ring and $\mathbb{Z}_{\mathbb{M}}$ represents the cipher-texts ring.

2.2 Symmetric HE schemes

In this part, different cryptographic functions of some well known symmetric HE schemes (MORE, PORE and DF) are explained in detailed. In addition, crypt-analysis and some problems related to their implementations are listed.

2.2.1 MORE approach

The MORE Approach is a symmetric encryption scheme that is based on matrix operations. It was published in 2012 by the authors of [36]. The security of the scheme resides in the hardness of factoring the product of large prime integers (i.e similar to RSA). A known plain-text/cipher-text attack is possible over the MORE due to its matrix structure and it is achieved by calculating the cipher matrix eigen-values. An explanation of this scheme is given below:

1. **Design Idea:** the basic idea starts with Rabin's encryption algorithm [5] introduced by the following equation:

$$E(x) = x^2(mod(N)) \tag{4}$$

   Where x is a plain-text and $N = p \times q$ (p and q are two large prime integers). The security of Rabin's crypto-system is based on the hardness of factoring large prime integers ($N = p \times q$). Rabin's crypto-system decryption procedure is then achieved using the Chinese Remainder Theorem (CRT) and the two prime factors of N (p and q).
2. **Basic Matrix Encryption:** given that Rabin's crypto-system allows only multiplication over encrypted data, a simple matrix based HE scheme is built as follows:

$$E(x, K) = K^{-1} \begin{bmatrix} x & 0 \\ 0 & r \end{bmatrix} K \quad mod(N) \tag{5}$$

where K is an invertible matrix in the ring $\mathbb{Z}_N$ and r is a random variable chosen also from the ring $\mathbb{Z}_N$.

3. **Homomorphic Properties:** given two plain-texts (x_1, x_2) and a secret (2×2) invertible matrix K from the ring $\mathbb{Z}_N$, homomorphic properties are verified as listed below:

   (a) Addition:
   $(E(x_1, K) + E(x_2, K)) \quad mod(N) = (K^{-1}\begin{bmatrix} x_1 & 0 \\ 0 & r_1 \end{bmatrix} K + K^{-1}\begin{bmatrix} x_2 & 0 \\ 0 & r_2 \end{bmatrix} K) \quad mod(N) =$
   $(K^{-1}\begin{bmatrix} x_1 + x_2 & 0 \\ 0 & r_1 + r_2 \end{bmatrix} K) \quad mod(N) = Enc(x_1 + x_2, K) \quad mod(N)$

   (b) Multiplication:
   $((E(x_1, K) \times E(x_2, K)) \quad mod(N) = (K^{-1}\begin{bmatrix} x_1 & 0 \\ 0 & r_1 \end{bmatrix} K \times K^{-1}\begin{bmatrix} x_2 & 0 \\ 0 & r_2 \end{bmatrix} K) \quad mod(N)=$
   $(K^{-1}\begin{bmatrix} x_1 \times x_2 & 0 \\ 0 & r_1 \times r_2 \end{bmatrix} K) \quad mod(N) = Enc(x_1 \times x_2, K) \quad mod(N)$
   Where r_1 and r_2 are two random variables chosen from the public ring $\mathbb{Z}_N$.

4. **Security of the Basic Matrix Scheme:**

   (a) Characteristic Equation: a possible attack on this scheme is achieved by calculating the eigen-values of the cipher matrix $E(x, k)$ given by (5); eigen-values represent the plain-texts. A practical solution for the attacker is deriving the characteristic equation given by:

   $$P(z) = det(z - E(x, K)) \quad mod(N) = z^2 - (x + r)z + xr = 0 \quad mod(N) \tag{6}$$

   Where z is the polynomial variable, x and r are the same cryptographic parameters given in (5). It is infeasible for the attacker to solve the characteristic equation given in (6) due to Rabin's crypto-system (i.e the hardness of revealing two large prime integers p and q $(N = p \times q)$).

   (b) Known Plain-text/Cipher-text Attack: knowing one couple of plain-text/cipher-text $(a, E(a, K))$, an attacker can drive a known plain-text/cipher-text attack over this basic matrix scheme. The related attack is achieved by calculating the eigen-vectors of the cipher matrix $E(a, k)$. Let $\vec{v}$ be an eigen-vector, $\vec{v}$ can be retrieved by applying the following equation:

   $$f(\vec{v}) = E(a, K) \times \vec{v} = a \times \vec{v} \tag{7}$$

Remark In the rest of this article, $diag(x_1, x_2, ...x_l)$ represents a diagonal square matrix of dimension $l \times l$ defined by:

$$diag(x_1, x_2, ...x_l) = \begin{bmatrix} x_1, & 0, & 0, & 0, & 0, & \cdots & 0 \\ 0, & x_2, & 0, & 0, & 0, & \cdots & 0 \\ 0, & 0, & x_3, & 0, & 0, & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0, & 0, & 0, & 0, & 0, & \cdots & x_l \end{bmatrix}$$

(c) **MORE Approach** [19, 36]**:** the formal MORE Approach is presented in this part. First, an invertible (4×4) matrix K is picked as a secret key of the encryption scheme. Starting from a plain-text $x \in \mathbb{Z}_N$, a diagonal matrix $diag(x, a, b, c)$ is constructed. Different parameters a, b and c listed in the previous diagonal matrix are the solutions to a set of linear congruences depending on the plain-text x and a random value $r \in \mathbb{Z}_N$ (i.e. a, b and c are calculated using the CRT). The corresponding cipher-text C is the similarity transformation of the matrix K by $diag(x, a, b, c) : C = K^{-1}\ diag(x, a, b, c)\ K$. Given $\{f_i\}_{i=1}^{m}$, where $f_i = p_i q_i$ such that p_i and q_i are two large prime integers and $N = \prod_{i=1}^{m} f_i$ such that $m = O(poly(\lambda))$, where λ is the security parameter.

The encryption algorithm is then represented by the following steps:

Remark Dimension is equal to 4 in the MORE Approach for simplicity, however, the scheme is applicable for any matrix dimension.

(d) **MORE Crypt-analysis:** as it was proven in [36], the MORE Approach is sensitive to chosen plain-text/cipher-text attack. The crypt-analysis of this scheme is given by the two following lemmas and theorem:

Algorithm 1 More encryption algorithm.

```
1: Step 1: Choose a random value $r \in \mathbb{Z}_N$.
2: Step 2:
3: for $i \leftarrow 1, m$ do
4:     Define a set of numbers $a_i$, $b_i$ and $c_i$
5:     For each $i$, exactly one of the $a_i$, $b_i$ and $c_i$ is equal to $x$. Let $a_i = x$ with probability $1 - \frac{1}{m+1}$, $b_i = x$ with probability $\frac{1}{2(m+1)}$ and $c_i = x$ with probability $\frac{1}{2(m+1)}$. Set the other two values equal to $r$. This way, for each $i$, one of the values $a$ , $b$, and $c$ is equal to $x$ and the other two are equal to $r$.
6: end for
7: Step 3: Using the CRT, let $a$, $b$ and $c$, be the solution of the set of simultaneous congruences $a = a_i\ mod f_i$, $b = b_i\ mod f_i$ and $c = c_i\ mod f_i$ for $1 \leq i \leq m$.
8: Step 4: Let $C = K^{-1}\ diag(x, a, b, c)\ K$.
9: Step 5: Given the cipher-text $C$, the decryption algorithm computes the plain-text $x = (K \times C \times K^{-1})$.
```

Lemma 1 *Let m' be the number of plain-text/cipher-text pairs the adversary has access to. If for some m', there exists an algorithm $A_d(C = E(x,\ k), \{x_l, C_l = E(x_l, K)\}_{l=1}^{m'})$ such that given m' chosen plain-text and cipher-text pairs $(x_l,\ C_l)$ and a cipher-text C, returns x with probability p, then there exists a PPT algorithm A_f using A_d as an oracle to factor f_i for some i with probability:*

$$p' = p(1 - \frac{1}{p_i})(1 - (1 - \frac{1}{m+1}(1 - \frac{1}{m+1})^{m'})^{m}) \tag{8}$$

Lemma 2 *Assuming that the probability to factor a λ bit integer in polynomial time is negligible, the encryption scheme is secure for $m' \leq m$.*

Theorem 1 *The bound m' of Lemma 1 can be weakened to $m' \leq m\, ln\ poly(\lambda)$, where $poly(\lambda)$ denotes some fixed polynomial in λ.*

Algorithm 2 PORE encryption algorithm.

```
1: Step 1: Pick a random variable $r \in \mathbb{Z}_{\mathbb{N}}$.
2: Step 2: Build a linear system of 2 equations and 2 unknowns $(c_1,\ c_2)$ as follows:
       $\begin{cases} v_1c_1 + c_2 = x \quad mod(N) \\ v_2c_1 + c_2 = r \quad mod(N) \end{cases}$
3: Step 3: solving the linear system given above: $c_1 = \frac{x-r}{v_1-v_2} \quad mod(N)$,
   $c_2 = \frac{rv_1 - xv_2}{v_1 - v_2} \quad mod(N)$.
```

2.2.2 PORE approach

The PORE Approach [19] stands for Polynomial Operation for Randomization and Encryption. This encryption scheme assures homomorphic operations over cipher-texts while using polynomial calculations. The PORE Approach is vulnerable to the known plain-text/cipher-text attacks due to its linear structure as will be discussed in the upcoming PORE crypt-analysis part. A detailed explanation of this scheme is given as follows:

1. Cryptographic Parameters:
   (a) Security Parameter λ: based on the chosen security level of the scheme.
   (b) Public Modulus N: given $\{f_i\}_{i=1}^{m}$, where $f_i = p_iq_i$ such that p_i and q_i are two large prime integers, the public modulus $N = \prod_{i=1}^{m} f_i$ such that $m = O(poly(\lambda))$.
   (c) Secret Key (v_1, v_2): 2 secret big integers chosen randomly from the public ring $\mathbb{Z}_N$.
   (d) Public Polynomial: a public polynomial $PP(v)$ of variable v is computed as follows: $PP(v) = (v - v_1)(v - v_2) = v^2 - (v_1 + v_2)v + v_1v_2 = v^2 + bv + c$, where $b = -(v_1 + v_2) \quad mod(N)$ and $c = v_1v_2 \quad mod(N)$. The two public parameters are shared with the third party to perform computation over encrypted data.

   Remark While knowing the two public parameters $b = -(v_1 + v_2) \quad mod(N)$ and $c = (v_1v_2) \quad mod(N)$ by the third non-trusted party, revealing secret values v_1, v_2 is done by finding the root of the public polynomial $PP(v) = v^2 + bv + c$ which is a hard problem based on the Rabin's cypto-system (i.e the hardness of factoring $N = \prod_{i=1}^{m} f_i$ where $f_i = p_iq_i$) as discussed in Section 2.2.1.

2. Encryption Procedure: starting from a plain-text $x \in \mathbb{Z}_{\mathbb{N}}$ and secret key $(v_1, v_2) \in \mathbb{Z}_N^2$, its corresponding cipher-text $C = (c_1, c_2) \quad mod(N)$ is calculated based on the following algorithm:
3. Decryption Procedure: having a cipher-text $C = (c_1, c_2)$ and the secret key $v = (v_1, v_2)$, the plain-text x is retrieved by applying the following equation:

$$x = c_1v_1 + c_2 \quad mod(N) \tag{9}$$

4. Homomorphic Properties: having two plain-texts x_1 and x_2 with their respective cipher-texts $C_1 = (c_1^1, c_2^1)$ and $C_2 = (c_1^2, c_2^2)$, homomorphic addition and multiplication over cipher-texts are achieved based on the following properties:
   (a) Homomorphic Addition: the resultant cipher-text after homomorphic multiplication is $C^{(1+2)}$ defined by $C^{(1+2)} = (c_1^{(1+2)}, c_2^{(1+2)}) = (c_1^1 + c_1^2, c_2^1 + c_2^2)$ because $x_1 + x_2 = c_1^1 v_1 + c_2^1 + c_1^2 v_1 + c_2^2 = (c_1^1 + c_1^2)v_1 + (c_2^1 + c_2^2)$.
   (b) Homomorphic Multiplication: the resultant cipher-text after homomorphic multiplication is defined by $C^{(1\times2)} = (c_1^{(1\times2)}, c_2^{(1\times2)}) = ((c_1^1+c_2^1)(c_1^2+c_2^2) - c_1^1c_1^2(1+b) - c_2^1c_2^2, \quad c_2^1c_2^2 - c_1^1c_2^1c)$, where b and c are the two public parameters given previously and shared with the third non-trusted party. Homomorphic multiplication is verified by applying the decryption equation given in (9) over $C^{(1\times2)}$:

$$\begin{aligned} c_1^{(1\times2)}v_1 + c_2^{(1\times2)} &= ((c_1^1+c_2^1)(c_1^2+c_2^2) - c_1^1c_1^2(1+b) - c_2^1c_2^2)v_1 + c_2^1c_2^2 - c_1^1c_2^1c \\ &= (c_1^1c_1^2)v_1^2 + (c_1^1c_2^2 + c_2^1c_1^2)v_1 + c_2^1 + c_2^2 \\ &= (c_1^1v_1 + c_2^1)(c_1^2v_1 + c_2^2) \\ &= (x_1 \times x_2) \quad mod(N) \end{aligned}$$

5. PORE Crypt-analysis: due to its linear structure (linear systems), the PORE Approach is vulnerable to known plain-text/cipher-text attack where revealing secret key (v_1, v_2) is possible while knowing only one couple of plain-text/cipher-text. Starting from one couple of known plain-text/cipher-text $(x, C = (c_1, c_2))$, the attacker can reveal the secret key while applying the following algoithm:

Algorithm 3 PORE known plain-text/cipher-text attack.

```
1: Step 1: Calculating $v_1$ → Solving: $c_2 + v_1c_1 = x \mod (N)$
2: Step 2: Calculating $v_2$ → Solving: $b = -(v_1 + v_2) \mod (N)$ and $c = (v_1v_2) \quad mod(N)$
```

2.2.3 Domingo rerrer (DF) scheme

DF scheme is a symmetric encryption scheme published in 2002 by Joseph Domingo Ferrer in [7]. The scheme is polynomial based, each cipher-text is seen as a uni-variate polynomial. Homomorphic properties are then defined as polynomial operations over the cipher-texts. DF scheme suffers form two main problems that are cipher-texts expansion after homomorphic multiplication and sensitivity to the known plain-text/cipher-text attack. An explanation of this scheme is given below:

1. Security Parameters: different security parameters of DF scheme [7] are given by the following:
   (a) λ: Security Parameter based on the chosen level of security.
   (b) m: Pubic Modulus $m > 10^{200}$ should have many small divisors.
   (c) d: Public Integer $d > 2$ represents the cipher-text dimension.
   (d) m': Secret Modulus, a small divisor of the public modulus such that $m = (m')^{\lambda}$.
   (e) r: Secret Key $r \in \mathbb{Z}_m$ should be invertible in the public ring $\mathbb{Z}_m$.
2. Encryption Procedure: Starting from a plain-text $a \in \mathbb{Z}_{m'}$, the encryption procedure is given by the two following steps:

(a) **Decompose Function**: the plain-text a is randomly decomposed into d elements $(a^{(1)}, a^{(2)}, a^{(3)}, \cdots, a^{(d)}) \in [\mathbb{Z}_m]^d$ such that $\sum_{i=1}^{d} a^{(i)} \quad mod(m') = a^{(1)}+a^{(2)}+\cdots+a^{(d)} \quad mod(m') = a$

(b) **Encryption Procedure**: an invertible secret key r is randomly picked from the public ring $\mathbb{Z}_m$, the cipher-text π of the plain-text a is then given by:
$\pi = [a^{(1)}r, a^{(2)}r^2, a^{(3)}r^3, \cdots, a^{(d)}r^d]$.
Cipher-text can be represented as a uni-variate polynomial $\pi(t)$ of variable t:
$\pi(t) = (a^{(1)}r)t + (a^{(2)}r^2)t^2 + (a^{(3)}r^3)t^3 + \cdots + (a^{(d)}r^d)t^d \quad (mod(m))$.

3. **Decryption Procedure:** the decryption is simply done by multiplying the i^{th} coordinate $\pi(i)$ of the cipher-text π by the $r^{-i} \quad mod(m)$ to retrieve $a^{(i)} \quad mod(m)$, then performing $\sum_{i=1}^{d} a^{(i)} \quad mod(m')$ to retrieve a.
4. **Homomorphic Properties:** the two homomorphic properties of DF (addition and multiplication) are investigated in this part. Let (C_1, C_2) respectively two cipher-texts generated from two different plain-texts (a_1, a_2) taken from the private ring $\mathbb{Z}_{m'}$ while using the security parameters and the DF encryption procedure given previously.

$$C_1 = [a_1^{(1)}r, a_1^{(2)}r^2, a_1^{(3)}r^3, \cdots, a_1^{(d)}r^d]$$

$$C_2 = [a_2^{(1)}r, a_2^{(2)}r^2, a_2^{(3)}r^3, \cdots, a_2^{(d)}r^d]$$

C_1 and C_2 are then represented in their uni-variate polynomial form with variable t as follows:

$$C_1 \rightarrow \pi_1(t) = (a_1^{(1)}r)t + (a_1^{(2)}r^2)t^2 + (a_1^{(3)}r^3)t^3 + \cdots + (a_1^{(d)}r^d)t^d mod(m).$$

$$C_2 \rightarrow \pi_2(t) = (a_2^{(1)}r)t + (a_2^{(2)}r^2)t^2 + (a_2^{(3)}r^3)t^3 + \cdots + (a_2^{(d)}r^d)t^d mod(m)$$

Given that, homomorphic operations of DF encryption scheme are restricted to polynomial calculations over encrypted data, homomorphic addition and multiplication are achieved as given below:

(a) Addition: the two polynomials $\pi_1(t)$ and $\pi_2(t)$ are added in the public ring $\mathbb{Z}_m[T]$ as follows:

$$\pi_1(t) + \pi_2(t) = ((a_1^{(1)} + a_2^{(1)})r)t + ((a_1^{(2)} + a_2^{(2)})r^2)t^2 + \\ ((a_1^{(3)} + a_2^{(3)})r^3)t^3 + ...((a_1^{(d)} + a_2^{(d)})r^d)t^d mod(m) \tag{10}$$

The resultant cipher-text after homomorphic addition is:

$$C_{1+2} = [(a_1^{(1)}+a_2^{(1)})r, (a_1^{(2)}+a_2^{(2)})r^2, (a_1^{(3)}+a_2^{(3)})r^3, \ldots, (a_1^{(d)}+a_2^{(d)})r^d]. \tag{11}$$

C_{1+2} is decrypted using the vector:

$$[r^1, r^{-2}, r^{-3}, \cdots, r^{-d}] \in [\mathbb{Z}_m]^d \tag{12}$$

Thus, $Dec_{(r,m')}(C_{1+2}) = (a_1 + a_2)$ is validated. Thus, the scheme is additive homomorphic.

(b) Multiplication: the two polynomials $\pi_1(t)$ and $\pi_2(t)$ are multiplied in the public ring $\mathbb{Z}_m[T]$ as follows:

$$\pi_1(t) \times \pi_2(t) = ((a_1^{(1)}a_2^{(1)})r^2)t^2 + ((a_1^{(1)}a_2^{(2)} + a_2^{(1)}a_1^{(2)})r^3)t^3 + ((a_1^{(1)}a_2^{(3)} \\ +a_1^{(3)}a_2^{(1)} + a_1^{(2)}a_2^{(2)})r^4)t^4 + \ldots + ((a_1^{(d)}a_2^{(d)})r^{2d})t^{2d}$$

The resultant cipher-text after homomorphic multiplication is:

$$C_{1\times 2} = [(a_1^{(1)}a_2^{(1)})r^2, (a_1^{(1)}a_2^{(2)} + a_2^{(1)}a_1^{(2)})r^3, (a_1^{(1)}a_2^{(3)} + a_1^{(3)}a_2^{(1)} + a_1^{(2)}a_2^{(2)})r^4, ..., (a_1^{(d)}a_2^{(d)})r^{2d}] \tag{13}$$

The decryption of $C_{1\times 2}$ is done using the vector $[r^2, r^3, r^4, \cdots, r^{2d}] \in \mathbb{Z}^{2d-1}$ and $Dec_{(r,m')} = (a_1 \times a_2)$ is validated. Thus, the scheme is multiplicative homomorphic.

5. DF cipher-text expansion: as mentioned previously, homomorphic multiplication in DF scheme is a polynomial multiplication over the cipher-texts. The cipher-text dimension will grow exponentially with homomorphic multiplication. One disadvantage of the DF scheme is that after evaluating a multiplicative circuit of high depth, the scheme loses its efficiency due to the resultant communication overhead (i.e. after performing k multiplication operations over 2^k cipher-texts of dimension d, the cipher-text dimension becomes $2^k(d-1)+1$).
6. DF Crypt-analysis: DF crypt-analysis has shown that the scheme is sensitive to known plain-text attack due to its algebraic structure. Using uni-variate polynomial resultant and Gaussian elimination, David Wagner has shown in [35] that starting with $4 \times poly(\lambda)$ (λ is the security parameter) couples of plain-text/cipher-text ($a_i/C_i = [a_i^{(1)}r, a_i^{(2)}r^2, \cdots, a_i^{(d)}r^d]$), revealing secret parameters (r, m') is possible with a probability close to $(1-(1-\frac{6}{\pi^2})^{poly(\lambda)})$.

2.3 BGV asymmetric scheme

BGV is a lattice based asymmetric encryption scheme published by the authors of [3]. The scheme suffers from two main problems that are cipher-text expansion after homomorphic multiplication and noise increase after circuit evaluation. The first problem is resolved using the Key Switching (KS) technique and the second one is resolved using the Modulus Switching (MS) technique. The security of the scheme is based on the hardness of Learning With Errors (LWE) and, as far as we know, no attack is introduced againt the concerned scheme. A detailed explanation of the BGV scheme is given below:

1. Basic Scheme: the basic BGV [3, 4] scheme can be modeled as a symmetric encryption scheme, that operates over the bit level. Secret key and cipher-text are given respectively by $s \in \mathbb{Z}_q^{[m,1]}$ and $c \in \mathbb{Z}_q^{[m,1]}$, where m is the lattice dimension.
   Decryption is given by the two following vector and matrix forms, where decryption works as long as the noise $((s.c)mod(q) << \alpha q$:

$$\begin{matrix} Vector \quad Form : ((\langle s, c\rangle mod(q))mod(2) \\ Matrix \quad Form : ((s.c)mod(q))mod(2) \end{matrix} \tag{14}$$

2. Building the Homomorphic Scheme
   (a) Addition: the scheme is an instance of Error Correcting Code (ERC), addition is valid as long as the noise is small enough.
   (b) Multiplication: building homomorphic multiplication is done using tensor product as given in the following equation:

$$M = u \otimes v = \{M_{ij} = (u_i v_j)\} \tag{15}$$

It is easy to demonstrate that:

$$s(u \otimes v)s^t = \langle s, u\rangle\langle s, v\rangle = (s.u)(s.v) \tag{16}$$

If the noise level is low enough after multiplicative operation, decryption can be written as:

$$(s(u \otimes v)s^t mod(q))mod(2) = ((\langle s, u\rangle\langle s, v\rangle)mod(q))mod(2) \tag{17}$$

It is more preferable to deal with vectors rather than matrices and the following linearization is proposed:

Given that:

$$c = u \otimes v = \begin{bmatrix} u_1v_1 & u_1v_2 & \cdots & u_1v_m \\ u_2v_1 & u_2v_2 & \cdots & u_2v_m \\ \vdots & \vdots & \ddots & \vdots \\ u_mv_1 & u_mv_2 & \cdots & u_mv_m \end{bmatrix}$$

An extended linear version of c is represented by $c^* = vect(u \otimes v) = [u_1v_1, u_1v_2, ..., u_mv_m]$

$$s \otimes s = \begin{bmatrix} s_1s_1 & s_1s_2 & \cdots & .s_1s_m \\ s_2s_1 & s_2s_2 & ...... & s_2s_m \\ \vdots & \vdots & \ddots & \vdots \\ s_ms_1 & s_ms_2 & \cdots & s_ms_m \end{bmatrix}$$

and

$$s^* = vect(s \otimes s) = [s_1s_1, s_1s_2, \cdots, s_ms_m]$$

It is simple to demonstrate that $\langle c^*, s^*\rangle = \langle s, u\rangle\langle s, v\rangle = sMs^t$ and the decryption equation can then be written by the following:

$$Dec(c^*) = (\langle c^*, s^*\rangle mod(q))mod(2) \tag{18}$$

(18) works as long as the noise $(\langle c^*, s^*\rangle mod(q))$ is quite small.

3. Dimension Growth: due to the tensor product used with homomorphic multiplication, the cipher-text dimension will grow exponentially. Starting from 2^k cipher-texts of dimension m each and after doing k multiplication operations, the resultant cipher-text dimension becomes m^{2^k}.

   To resolve the cipher-text dimension expansion present in BGV scheme, Key Switching (KS) is introduced. The main concept of KS is that after having an extended cipher-text c^* (of dimension $n > m$) with respect to an extended secret key $s^* = vect(s \otimes s)$ (of dimension $n > m$), a new lower dimension cipher-text c' with respect to a lower dimension secret key s' is built such that:

$$Dec_{s'}(c') = Dec_{s^*}(c^*) \tag{19}$$

   To achieve KS, an encryption matrix $M \in \mathbb{Z}_q^{[m,n]}$ is published and defined by:

$$M(s^* \rightarrow s), \quad c' = Mc^* \quad ([m.1] = [m, n][n, 1]) \quad m < n \tag{20}$$

Remark The detailed explanation, implementation and performance analysis regarding KS are given in [3, 11].

4. Level Somewhat Homomorphic Scheme

   (a) Basic BGV SHE Scheme: Level SHE scheme symbolizes a level by level path. For a circuit of depth d, d random secret keys are generated as $s_i = [1|t_i] \in \mathbb{Z}_q^m$ where $0 \leq i \leq d-1$. For each level i of the circuit, a public matrix $M_i \in \mathbb{Z}_q^{[m,n]}$ is published. For level 0, $M_0 = M(0 \to s_0)$ is generated and for any level i $M_i = M(s_i^{**} \to s_i)$ is generated.

      i Parameters Generation: starting from security parameter λ, the lattice dimension is given by $m \approx poly(\lambda)$ and the public ring dimension is given by $q \approx poly(m)$.

      ii Public Matrices Generation:

$$\begin{aligned} s_0 M_0 &= 2e_0 \\ s_i M_i &= 2e_i + s_{i-1}^{**} \end{aligned} \tag{21}$$

      iii Encryption Procedure: the encryption mechanism in this case is done based on the following equation:

$$Enc(b) = \quad M_0 r + \begin{bmatrix} b \\ 0 \\ \cdots \\ 0 \end{bmatrix} \tag{22}$$

      where r is a random vector $\in \{0, 1\}^n$.

      iv Decryption Procedure: the decryption mechanism at any level i is performed using the secret key s_i and applying the following equation:

$$Dec(c, i) = (\langle s_i, c_i \rangle mod(q)) mod(2) \tag{23}$$

      v Homomorphic Properties: starting from two cipher-texts (c_1, i) and (c_2, i) at a level i, homomorphic properties can be done by the following:

         A. Addition: addition is simply performed by $(c_{add}, i) = (c_1 + c_2, i)$.

         B. Multiplication: multiplication is performed by applying the tensor product over the two cipher-texts (c_1, i) and (c_2, i) at level i. The resultant cipher-text is then linearized by $c^* = vect(c_1,\ c_2)$. Finally, the fresh cipher-text is calculated by $c' = Mc^*$based on the KS technique and the public matrix M.

   (b) Making the Scheme Fully Homomorphic: the noise at level i is given by $(sc_i mod(q))$. Noise level will be doubled after addition and squared after multiplication. In this way, a circuit of limited depth can be evaluated as long as the noise is lower than the modulus q. To evaluate deeper circuit, the Modulus Switching (MS) technique is introduced in the literature. MS is based on switching into another modulus (different than q) but the decryption is always possible. The key challenge with MS is that while having a cipher-text c with respect to a secret key s, a new cipher-text c' for some modulus $f < q$ should be built such that:

$$((\langle c', s\rangle mod(f))mod(2) = ((\langle c, s\rangle mod(q))mod(2) \tag{24}$$

Remark The detailed explanation, implementation and performance analysis of MS are given in [3, 11].

5. BGV Crypt-analysis: The hardness of the BGV scheme is based on the hardness of Learning With Error (LWE) introduced by Oded Regev in [26, 29]. As far as we known, no attack is introduced in the literature against the BGV scheme.

3 Homomorphic hybrid symmetric encryption scheme (HHSES)

HHSES is a new symmetric encryption scheme obtained by mixing the homomorphic behavior of two well known symmetric variants: the MORE approach and the DF scheme. Following the scheme's topology, homomorphic properties are based on polynomial operations over matrices. One main disadvantage of the new scheme is exponential cipher-text expansion due to polynomial multiplication over matrices. The scheme will suffer from high storage overhead and low performance especially when dealing with circuits of high depth. To resolve the latter problem, KS technique [3, 11] is applied to reduce the cipher-text dimension and hence improve its efficiency. One main characteristic of HHSES is its resistance against different types of attacks including the known plain-text/cipher-text attacks. A detailed explanation of designing the new scheme is presented in the next.

3.1 Building the scheme

In this part, we will describe the mathematical concept of the proposed HE cipher scheme including the security parameters in addition to encryption and decryption procedures.

3.1.1 HHSES parameters

HHSES parameters are a mix of both MORE and DF encryption parameters and are given as follows:

1. λ: security parameter, based on the required security level.
2. ψ: secret modulus, that represents the dimension of the private ring $\mathbb{Z}_{\psi}$, where different plain-texts are chosen.
3. N, RSA ring modulus: is given as $N = \prod_{i=1}^{m} f_i$, $\{f_i\}_{i=1}^{m}$ and $f_i = p_i q_i$ such that p_i and q_i are two large prime integers, where $m = O(poly(\lambda))$.
4. $\Psi = \psi \times N$: is the public modulus that represents the dimension of the public ring $\mathbb{Z}_{\Psi}$.
5. d: cipher-text dimension, each plain-text is decomposed into d elements using the random decompose function of DF given in Section (3.1.2).
6. r: invertible secret key, an invertible secret key chosen from the public ring $\mathbb{Z}_{\Psi}$.
7. K: invertible secret matrix given by $K = (k_{i,j})$, where $k_{i,j} \in \mathbb{Z}_{\Psi}$.
   In our previous article [12], we proposed a lightweight and practical technique to generate an invertible random square matrix K as explained in Appendix A. The authors of [34] detected a vulnerability in this model if the dynamic key approach is not used, which is in contrast to the presented assumption and the main idea of this work. The

strength of the enhanced MORE approach in resisting known plain-text/cipher-text attacks is weak as explained in details in Appendix B if the employed cryptographic primitives are static. Here, a new countermeasure is proposed against this vulnerability in the case of static cryptographic primitives are used. The proposed solution applies a new random model in generating the invertible secret matrix key K, which is based on the following steps.

(a) Step 1: lower bound triangular matrix A, a random invertible lower bound matrix A of dimension $(n \times n)$ is generated based on the following form:

$$A = \begin{bmatrix} a_{1,1} & 0 & 0 & 0 & \cdots & 0 \\ a_{1,2} & a_{2,2} & 0 & 0 & \cdots & 0 \\ a_{3,1} & a_{3,2} & a_{3,3} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ a_{n,1} & a_{n,2} & a_{n,3} & 0 & \cdots . & a_{n,n} \end{bmatrix} \tag{25}$$

The matrix A is created above such that $gcd(a_{i,i}, \Psi) = 1$ for $i \in \{1, 2, 3, ..., n\}$, thus $(a_{i,i})^{-1} \in Z_{\Psi}$. The lower bound matrix A is invertible since $Det(A) = \prod_{i=1}^{n} a_{i,i}$ and $(Det(A))^{-1} = \prod_{i=1}^{n} (a_{i,i})^{-1}$.

(b) Step 2: higher bound triangular matrix B, a random invertible matrix B of dimension $(n \times n)$ is generated based on the following form:

$$B = \begin{bmatrix} b_{1,1} & b_{1,2} & b_{1,3} & \cdots & b_{1,n} \\ 0 & b_{2,2} & b_{2,3} & \cdots & b_{2,n} \\ 0 & 0 & b_{3,3} & \cdots & b_{3,n} \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & \cdots & b_{n,n} \end{bmatrix} \tag{26}$$

B is built such that $gcd(b_{i,i}, \Psi) = 1$. Higher bound matrix B is invertible since $(Det(B))^{-1}$ exists in the ring Z_{Ψ}.

(c) Step 3: random secret invertible matrix K Generation, the invertible matrix K is then given by $A \times B$. $(K)^{-1}$ exits and $(K)^{-1} = (A \times B)^{-1} = (B)^{-1} \times (A)^{-1}$.

Due to the new secret matrix K generation, the attack given in [34] is no more possible. It is mentioned previously that the latter attack profits from the linear relations existent within the matrix K given in (38). The new matrix generation presented is based on creating the invertible secret matrix K from random values where no clear patterns or linear relations exist among different elements of the matrix K. This new way of building K can generate odd and even matrices, while with the previous one ((38)) only matrices with even dimensions were possible.

3.1.2 Encryption procedure

Starting from a vector $X = [x_1, x_2, ..., x_k, ..., x_l]$ of l plain-texts such that $x_i \in \mathbb{Z}_{\psi}$, the encryption procedure is given by the following steps:

1. DF Encryption Procedure: each element $x_k \in X$ is decomposed and encrypted using DF cryptographic parameters (d and r) as follows:

$$\begin{cases} X_1 = [x_1^{(1)}r, x_1^{(2)}r^2, ..., x_1^{(d)}r^d] \quad mod(\Psi) \\ X_2 = [x_2^{(1)}r, x_2^{(2)}r^2, ..., x_2^{(d)}r^d] \quad mod(\Psi) \\ \vdots \\ X_k = [x_k^{(1)}r, x_k^{(2)}r^2, ..., x_k^{(d)}r^d] \quad mod(\Psi) \\ \vdots \\ X_l = [x_l^{(1)}r, x_l^{(2)}r^2, ..., x_l^{(d)}r^d] \quad mod(\Psi) \end{cases} \tag{27}$$

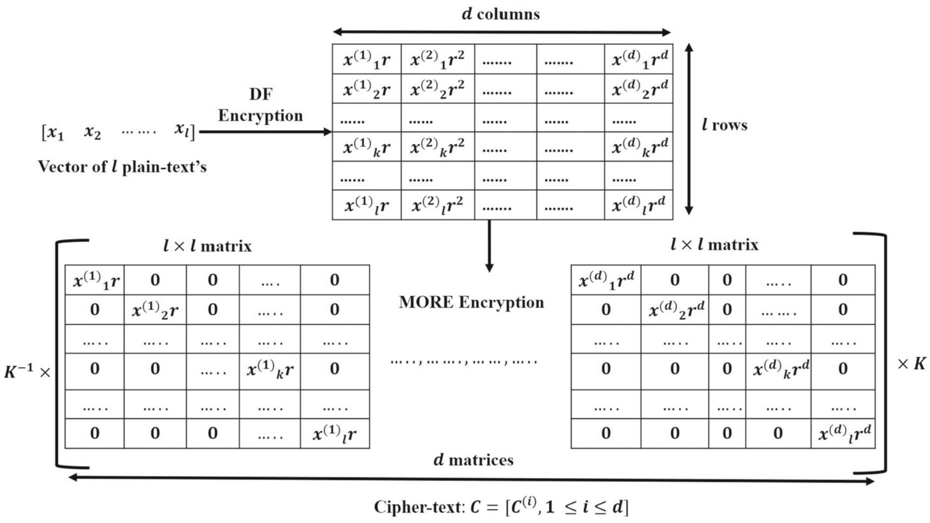
2. Matrix Polynomial Form: different vectors $X_1, X_2, .., X_k, .., X_l$ given above are grouped by the following polynomial matrix form:

$$\begin{aligned} P(t) &= \sum_{i=1}^{i=d} diag(x_1^{(i)}r^i, x_2^{(i)}r^i, x_3^{(i)}r^i, \ldots, x_l^{(i)}r^i)t^i \quad mod(\Psi) \\ &= diag(x_1^{(1)}r, x_2^{(1)}r, x_3^{(1)}r, \ldots, x_l^{(1)}r)t + diag(x_1^{(2)}r^2, x_2^{(2)}r^2, x_3^{(2)}r^2, \ldots, x_l^{(2)}r^2)t^2 \\ &\quad + \ldots + diag(x_1^{(d)}r^d, x_2^{(d)}r^d, x_3^{(d)}r^d, \ldots, x_l^{(d)}r^d)t^d \quad mod(\Psi) \end{aligned}$$

3. MORE Encryption Procedure: finally an invertible secret matrix K is built using the matrix generation engine explained in Section 3.1.1. Thus, the encryption of the vector X is given by the following equation:

$$Enc(X) = K^{-1} \cdot P(t) \cdot K \quad mod(\Psi) \tag{28}$$

An illustration of the encryption procedure is given in Fig. 3.



d columns
DF Encryption
$[x_1 \ x_2 \ .... \ x_l]$
Vector of l plain-text's
l rows
$x^{(1)}{}_1r$ $x^{(2)}{}_1r^2$ .... $x^{(d)}{}_1r^d$
$x^{(1)}{}_2r$ $x^{(2)}{}_2r^2$ .... $x^{(d)}{}_2r^d$
$x^{(1)}{}_kr$ $x^{(2)}{}_kr^2$ .... $x^{(d)}{}_kr^d$
$x^{(1)}{}_lr$ $x^{(2)}{}_lr^2$ .... $x^{(d)}{}_lr^d$
$l \times l$ matrix
$x^{(1)}{}_1r$ 0 0 0
0 $x^{(1)}{}_2r$ 0 0
0 0 ..... $x^{(1)}{}_kr$ 0
0 0 0 ..... $x^{(1)}{}_lr$
MORE Encryption
$l \times l$ matrix
$x^{(d)}{}_1r^d$ 0 0 0
0 $x^{(d)}{}_2r^d$ 0 0
0 0 0 $x^{(d)}{}_kr^d$ 0
0 0 0 0 $x^{(d)}{}_lr^d$
$K^{-1} \times$
$\times K$
d matrices
Cipher-text: $C = [C^{(i)}, 1 \leq i \leq d]$

Fig. 3 The proposed HHSES encryption algorithm

3.1.3 Decryption procedure:

HHSES is a symmetric encryption scheme, the decryption procedure is then the inverse of the encryption process. Hence, decryption is based on the following steps:

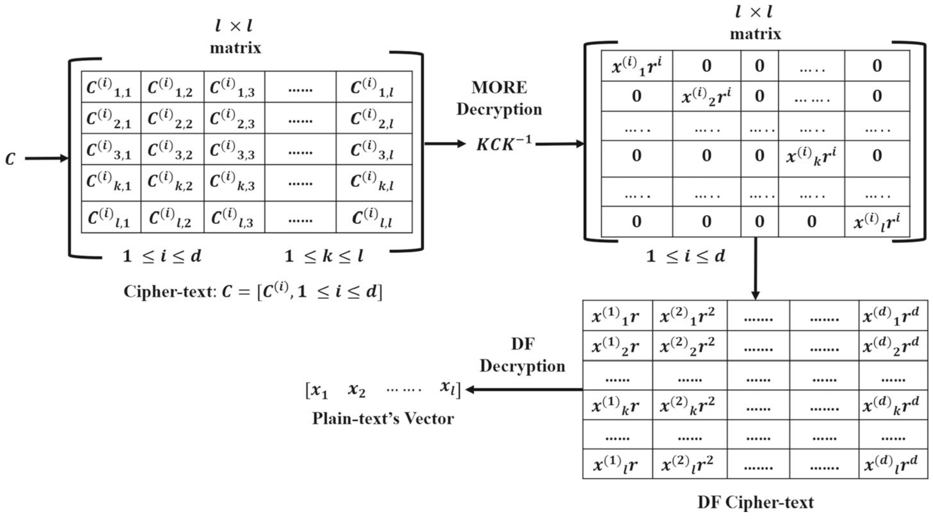
1. Retrieving the Polynomial Matrix $P(t)$: this is done by applying the following equation:
$$P(t) = K \cdot Enc(X) \cdot K^{-1} \quad mod(\Psi) \tag{29}$$
2. Retrieving the X_i Vectors: once the polynomial matrix $P(t)$ is retrieved, the following vectors are built
$X_i = [x_i^{(1)}r, x_i^{(2)}r^2, x_i^{(3)}r^3, ..., x_i^{(d)}r^d]$ for $1 \leq i \leq l$.
3. Retrieving the plain-texts vector X: after building the X_i vectors, the DF decryption procedure is applied as given in Section 2.2.3 to calculate the primitive plain-text vector X.

An illustration of the decryption procedure is given in Fig. 4.

3.2 Homomorphic properties

Starting from two different plain-text vectors X and Y of dimension l such that: $X = [x_1, x_2, x_3, ...x_l]$ and $Y = [y_1, y_2, y_3, ...y_l]$ where $x_i, y_i \in \mathbb{Z}_\psi$. The encryption of X and Y using HHSES is given by:
$Enc(X) = K^{-1} \cdot P_x(t) \cdot K$ such that $P_x(t) = \sum_{i=1}^{i=d} diag(x_1^{(i)}r^i, x_2^{(i)}r^i, x_3^{(i)}r^i, ..., x_l^{(i)}r^i)t^i \quad mod(\Psi)$.
$Enc(Y) = K^{-1} \cdot P_y(t) \cdot K$ such that $P_y(t) = \sum_{i=1}^{i=d} diag(y_1^{(i)}r^i, y_2^{(i)}r^i, y_3^{(i)}r^i, ..., y_l^{(i)}r^i)t^i \quad mod(\Psi)$.
Homomorphic properties are verified as follows:

1. Addition:
$Enc(X+Y) = Enc(X) + Enc(Y) = K^{-1} \cdot P_x(t) \cdot K + K^{-1} \cdot P_y(t) \cdot K = K^{-1} \cdot$



$l \times l$ matrix
C
$C^{(i)}_{1,1}$ $C^{(i)}_{1,2}$ $C^{(i)}_{1,3}$ ...... $C^{(i)}_{1,l}$
$C^{(i)}_{2,1}$ $C^{(i)}_{2,2}$ $C^{(i)}_{2,3}$ ...... $C^{(i)}_{2,l}$
$C^{(i)}_{3,1}$ $C^{(i)}_{3,2}$ $C^{(i)}_{3,3}$ ...... $C^{(i)}_{3,l}$
$C^{(i)}_{k,1}$ $C^{(i)}_{k,2}$ $C^{(i)}_{k,3}$ ...... $C^{(i)}_{k,l}$
$C^{(i)}_{l,1}$ $C^{(i)}_{l,2}$ $C^{(i)}_{l,3}$ ...... $C^{(i)}_{l,l}$
$1 \leq i \leq d$
$1 \leq k \leq l$
Cipher-text: $C = [C^{(i)}, 1 \leq i \leq d]$
MORE Decryption
KCK^{-1}
$l \times l$ matrix
$x^{(i)}{}_1r^i$ 0 0 0
0 $x^{(i)}{}_2r^i$ 0 0
0 0 0 $x^{(i)}{}_kr^i$ 0
0 0 0 0 $x^{(i)}{}_lr^i$
$1 \leq i \leq d$
$x^{(1)}{}_1r$ $x^{(2)}{}_1r^2$ .... $x^{(d)}{}_1r^d$
$x^{(1)}{}_2r$ $x^{(2)}{}_2r^2$ .... $x^{(d)}{}_2r^d$
$x^{(1)}{}_kr$ $x^{(2)}{}_kr^2$ .... $x^{(d)}{}_kr^d$
$x^{(1)}{}_lr$ $x^{(2)}{}_lr^2$ .... $x^{(d)}{}_lr^d$
DF Cipher-text
DF Decryption
$[x_1 \quad x_2 \quad .... \quad x_l]$
Plain-text's Vector

Fig. 4 The corresponding decryption scheme of HHSES

$(P_x(t) + P_y(t)) \cdot K = K^{-1} \cdot (P_{x+y}(t)) \cdot K$
Decryption is done by applying these steps:

(a) Step1: $K(Enc(X+Y)K^{-1}$ is calculated to retrieve $P_{x+y}(t)$.
(b) Step2: $P_{x+y}(t) = \sum_{i=1}^{i=d} diag((x_1^{(i)} + y_1^{(i)})r^i, (x_2^{(i)} + y_2^{(i)})r^i, (x_3^{(i)} + y_3^{(i)})r^i, ..., (x_l^{(i)} + y_l^{(i)})r^i)t^i \quad mod(\Psi)$.
(c) Step3: based on the polynomial $P_{x+y}(t)$, the following vectors are retrieved as follows:

$$\begin{cases} X_1 + Y_1 = [(x_1^{(1)} + y_1^{(1)})r, (x_1^{(2)} + y_1^{(2)})r^2, ..., (x_1^{(d)} + y_1^{(d)})r^d] \\ X_2 + Y_2 = [(x_2^{(1)} + y_2^{(1)})r, (x_2^{(2)} + y_2^{(2)})r^2, ..., (x_2^{(d)} + y_2^{(d)})r^d] \\ \vdots \\ X_l + Y_l = [(x_l^{(1)} + y_l^{(1)})r, (x_l^{(2)} + y_l^{(2)})r^2, ..., (x_l^{(d)} + y_l^{(d)})r^d] \end{cases}$$

By applying the DF decryption procedure given in Section 2.2.3, it is simple to calculate $X + Y = [x_1 + y_1, x_2 + y_2, ..., x_l + y_l]$ and the scheme is additive homomorphic.

2. Multiplication:
$Enc(X \times Y) = Enc(X) \times Enc(Y) = (K^{-1} \cdot P_x(t) \cdot K) \times (K^{-1} \cdot P_y(t) \cdot K) = K^{-1} \cdot (P_x(t) \times P_y(t)) \cdot K = K^{-1} \cdot (P_{x \times y}(t)) \cdot K$. Decryption is done by applying the following steps:

(a) Step 1: $K(Enc(X \times Y))K^{-1}$ is calculated to retrieve $P_{x \times y}(t)$.
(b) Step 2:

$$\begin{aligned} P_{x\times y}(t) = {} & diag((x_1^1 y_1^1)r^2, (x_2^1 y_2^1)r^2, ..., (x_l^1 y_l^1)r^2)t^2 \\ & + diag((x_1^{(1)} y_1^{(2)} + y_1^{(1)} x_1^{(2)})r^3, (x_2^{(1)} y_2^{(2)} + y_2^{(1)} x_2^{(2)})r^3, \ldots, (x_l^{(1)} y_l^{(2)} \\ & + y_l^{(1)} x_l^{(2)})r^3)t^3 + diag((x_1^{(1)} y_1^{(3)} + x_1^{(3)} y_1^{(1)} + x_1^{(2)} y_1^{(2)})r^4, (x_2^{(1)} y_2^{(3)} \\ & + x_1^{(3)} y_2^{(1)} + x_2^{(2)} y_2^{(2)})r^4, \ldots, (x_l^{(1)} y_l^{(3)} + x_1^{(3)} y_l^{(1)} + x_1^{(2)} y_l^{(2)})r^4)t^4 \\ & + \ldots + diag((x_1^{(d)} y_1^{(d)})r^{2d}, (x_2^{(d)} y_2^{(d)})r^{2d}, \ldots, (x_l^{(d)} y_l^{(d)})r^{2d})t^{2d} \end{aligned}$$

(a) Step 3: based on the polynomial $P_{x \times y}$, the following vectors are retrieved:

$$\begin{cases} X_1 \times Y_1 = [(x_1^1 y_1^1)r^2, (x_1^{(1)} y_1^{(2)} + y_1^{(1)} x_1^{(2)})r^3, (x_1^{(1)} y_1^{(3)} + x_1^{(3)} y_1^{(1)} + x_1^{(2)} y_1^{(2)})r^4, ..., (x_1^{(d)} y_1^{(d)})r^{2d}] \\ X_2 \times Y_2 = [(x_2^1 y_2^1)r^2, (x_2^{(1)} y_2^{(2)} + y_2^{(1)} x_2^{(2)})r^3, (x_2^{(1)} y_2^{(3)} + x_2^{(3)} y_2^{(1)} + x_2^{(2)} y_2^{(2)})r^4, ..., (x_2^{(d)} y_2^{(d)})r^{2d}] \\ \vdots \\ X_l \times Y_l = [(x_l^1 y_l^1)r^2, (x_l^{(1)} y_l^{(2)} + y_l^{(1)} x_l^{(2)})r^3, (x_l^{(1)} y_l^{(3)} + x_l^{(3)} y_l^{(1)} + x_l^{(2)} y_l^{(2)})r^4, ..., (x_l^{(d)} y_l^{(d)})r^{2d}] \end{cases}$$

after applying the DF decryption procedure given in Section 2.2.3, it is simple calculate $X \times Y = [x_1 \times y_1, x_2 \times y_2, ..., x_l \times y_l]$ and the scheme is multiplicative homomorphic.

3.3 Dimension growth and KS technique

In this part, KS technique is introduced to resolve HHSES dimension growth after homomorphic multiplication while preserving the homomorphic behavior of the scheme and the original plain-text. More details are presented in the next.

3.3.1 Cipher-text expansion

A main problem of the HHSES is the cipher-text dimension expansion after homomorphic multiplication. For example, as it is given in Section 3.2, the multiplication of two cipher-texts formed of d matrices will give a resultant cipher-text vector formed of $2d-1$ matrices. Following the same analogy of DF polynomial calculation and starting from 2^k cipher-texts each one formed of d matrices, the resultant cipher-text is formed of $(2^k(d-1)+1)$ matrices. Thus, evaluating a multiplicative circuit with high depth becomes inefficient due to the high storage overhead. A practical solution to enhance the HHSES efficiency and to reduce the cipher-text dimension after each homomorphic multiplication is applying Key Switching (KS) technique.

3.3.2 KS technique for DF

As it is given previously in (19) and (20), the main idea of KS is to publish a matrix $M \in \mathbb{Z}_{\Psi}^{[H,d]}$ $(H = 2d-1)$ that verifies the following:
$M(s^* \to s)$ such that $c' = Mc^*$, $(Dimension : (d, 1) = (d, H) \times (H, 1))$.
KS was applied over DF encryption scheme as it is listed in [16]. The new secret key is given by $s' = [r', r'^2, r'^3, ..., r'^d]$ and $s'_{inverse} = [r^{-1} \quad mod(m), r \in s']$ and $t' = [s'(i)_{inverse}, 2 \leq i \leq d]$. The matrix M is of dimension (d, H) and is formed of two sub-matrices b and A having the following form $\begin{bmatrix} b \\ A \end{bmatrix}$.
Matrix b forms the first row of the public matrix M and it is given by $b = (-t'A + \psi e + s^*_{inverse})r^* \in \mathbb{Z}_{\Psi}^{[1,H]}$. The matrix A is of dimension $(d-1, H)$ and is generated based on a uniform distribution over $\mathbb{Z}_{\Psi}$. Different secret parameters (s^*, s', ψ) are encrypted within the matrix M based on the hardness of LWE [26, 29] and $e = [e_1, e_2, e_3, ..., e_H]$ is a random Gaussian noise matrix.
Given that $c' = Mc^*$, the following relations are demonstrated:

$$\begin{aligned} s'_{inverse}c' \quad mod(\psi) &= s'_{inverse}(Mc^*) \quad mod(\psi) = (s'_{inverse}M)c^* - k\psi \\ &= (\psi e + s^*_{inverse})c^* - k\psi \\ &= s^*_{inverse}c^* + \psi(ec^* - k) \\ &= s^*_{inverse}c^* \quad mod(\psi) \end{aligned}$$

where $k \in \mathbb{Z}$ and ((20), (19)) are both satisfied.

3.3.3 Applying KS technique for HHSES

Given a public matrix M generated based on Section 3.3.2.

$$M = \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} & \cdots & M_{1,H} \\ M_{2,1} & M_{2,2} & M_{2,3} & \cdots & M_{2,H} \\ M_{3,1} & M_{3,2} & M_{3,3} & \cdots & M_{3,H} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ M_{d,1} & M_{d,2} & M_{d,3} & \cdots & M_{d,H} \end{bmatrix}$$

where $M \in \mathbb{Z}_{\Psi}^{[d,H]}$ and a DF cipher-text C_{mult} obtained after multiplication that has the following form:

$$C_{mult} = \begin{bmatrix} (x_1^1 y_1^1) r^2 \\ (x_1^1 y_1^2 + y_1^1 x_1^2) r^3 \\ (x_1^1 y_1^3 + x_1^3 y_1^1 + x_1^2 y_1^2) r^4 \\ \vdots \\ (x_1^d y_1^d) r^{2d} \end{bmatrix} \in \mathbb{Z}_{\Psi}^{[H=2d-1,1]}$$

The new fresh cipher-text C_{Fresh} of dimension $[d, 1]$ is given by $C_{Fresh} = M \times C_{mult}$, where

$$C_{Fresh} = \begin{bmatrix} M_{1,1} \times (x_1^1 y_1^1) r^2 + \cdots + M_{1,H} \times (x_1^d y_1^d) r^{2d} \\ M_{2,1} \times (x_1^1 y_1^1) r^2 + \cdots + M_{2,H} \times (x_1^d y_1^d) r^{2d} \\ \vdots \\ M_{d,1} \times (x_1^1 y_1^1) r^2 + \cdots + M_{d,H} \times (x_1^d y_1^d) r^{2d} \end{bmatrix} \in \mathbb{Z}_{\Psi}^{[d,1]}$$

Given a cipher-text C_{mult}^{HHSES} obtained after homomorphic multiplication using HHSES, the latter is a polynomial matrix that has the following form: $K^{-1} \cdot (P_{x \times y}(t)) \cdot K$, where K is the invertible secret matrix of HHSES and

$$\begin{aligned} P_{x\times y}(t) = {} & diag((x_1^1 y_1^1) r^2, (x_2^1 y_2^1) r^2, \cdots, (x_l^1 y_l^1) r^2) t^2 \\ & + diag((x_1^{(1)} y_1^{(2)} + y_1^{(1)} x_1^{(2)}) r^3, (x_2^{(1)} y_2^{(2)} + y_2^{(1)} x_2^{(2)}) r^3, \cdots, (x_l^{(1)} y_l^{(2)} \\ & + y_l^{(1)} x_l^{(2)}) r^3) t^3 + diag((x_1^{(1)} y_1^{(3)} + x_1^{(3)} y_1^{(1)} + x_1^{(2)} y_1^{(2)}) r^4, (x_2^{(1)} y_2^{(3)} + x_1^{(3)} y_2^{(1)} \\ & + x_2^{(2)} y_2^{(2)}) r^4, ..., (x_l^{(1)} y_l^{(3)} + x_1^{(3)} y_l^{(1)} + x_1^{(2)} y_l^{(2)}) r^4) t^4 \\ & + \ldots + diag((x_1^{(d)} y_1^{(d)}) r^{2d}, (x_2^{(d)} y_2^{(d)}) r^{2d}, ..., (x_l^{(d)} y_l^{(d)}) r^{2d}) t^{2d} \end{aligned}$$

C_{mult}^{HHSES} is a cipher-text formed of $(2 \times d - 1)$ matrices. The main concept of KS is to generate a new cipher-text C_{Fresh}^{HHSES} formed of d matrices without modifying the original plain-text. This can be achieved by building a global public matrix M_{Global} formed of $d \times H$ sub-matrices of dimension $(l \times l)$ each having the following form:
$\left[K^{-1} diag(M_{i,j}, M_{i,j}, ...M_{i,j}) K\right]$ where $1 \leq i \leq d$ and $1 \leq j \leq H = 2d - 1$.
C_{mult}^{HHSES} is presented by the following form:

$$\begin{bmatrix} K^{-1} \times diag((x_1^1 y_1^1) r^2, \cdots, (x_l^1 y_l^1) r^2) \times K \\ K^{-1} \times diag((x_1^{(1)} y_1^{(2)} + y_1^{(1)} x_1^{(2)}) r^3, \cdots, (x_l^{(1)} y_l^{(2)} + y_l^{(1)} x_l^{(2)}) r^3) \times K \\ \vdots \\ K^{-1} diag((x_1^{(d)} y_1^{(d)}) r^{2d}, \cdots, (x_l^{(d)} y_l^{(d)}) r^{2d}) \times K \end{bmatrix}$$

The following equation can than be demonstrated:

$$C_{Fresh}^{HHSES} = M_{Global} \times C_{mult}^{HHSES} \tag{30}$$

4 HHSES security analysis

In this section, a deep security analysis is done for the HHSES to validate its high immunity against several types of attacks. Different security tests are implemented as given in [27] to show its immunity against statistical attacks and related key attacks, and to prove that the scheme provides the avalanche effect, the uniformity and the independence properties. Finally a theoretical crypt-analysis shows that the new scheme is resistant against known plain-text/cipher-text attacks. Different security tests are implemented under Python using Sagemath library with a personal laptop having the following specifications: OS Ubuntu 14.04, RAM 3.9 GB, Processor Intel Core $i7 - 8550$U CPU @ 1.8 GHZ, 64 bit, Disk 24.1 GB. For the HHSES implementation, the security parameter λ is taken 20, DF dimension $d = 2$, secret modulus $\psi = 256$. Different plain-texts are sampled as Bytes ($\mathbb{Z}_{\psi=256}$) from a Gaussian distribution having a mean value $\mu = 128$ and standard deviation $\sigma = 16$.

Remark During the encryption procedure, DF dimension is taken $d = 2$ (in Section 5.4, a theoretical crypt-analysis shows that the scheme is resistant to known plain-text/cipher-text attacks even with the lowest DF dimension, $d = 2$). The resultant cipher-text $C = [C_1, C_2]$ is then formed of 2 matrices.

Remark Different elements of the cipher-text $C = [C_1, C_2]$ are in the ring $\mathbb{Z}_\Psi$ (Ψ is the public modulus). To accomplish distribution, recurrence, correlation and entropy tests cipher-texts are converted to the Bytes level as illustrated in Fig. 5. In the latter figure, each element $(c \in C_i)_{(1 \leq i \leq 2)}$ of the cipher-text C is decomposed into bits $[b^i_1, b^i_2, ..., b^i_{\lceil log_2 \rceil (\Psi)}]_{(1 \leq i \leq 2)}$, then the $(\lceil log_2 \rceil (\Psi))$ bits for each cipher-text are grouped by sets of 8 bits and converted into integers to form the new cipher-text at the byte level $[B^i_1, B^i_2, ..., B^i_{\frac{\lceil log_2 \rceil (\Psi)}{8}}]_{(1 \leq i \leq 2))}$, where $B^i_j \in \mathbb{Z}_{256}$.

4.1 Resistance against statistical attacks

Resistance against statistical attacks is assured by the proposed scheme while providing a high level of randomness. Thus, the resultant cipher-text should present the independence and uniformity criteria. Uniformity can be proved visually by plotting the histogram of the cipher-texts and applying the entropy test. On the other hand, the independence property is validated visually by plotting the recurrence of the encrypted data and calculating the percentage of difference in bit level between the original and the encrypted data (difference

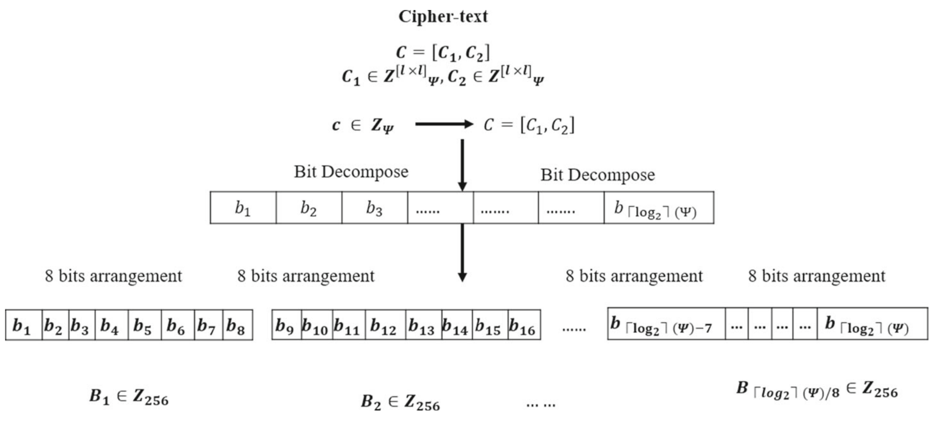

Cipher-text
$C = [C_1, C_2]$
$C_1 \in Z^{[l \times l]}_\Psi, C_2 \in Z^{[l \times l]}_\Psi$
$c \in Z_\Psi \longrightarrow C = [C_1, C_2]$
Bit Decompose
Bit Decompose
b_1 b_2 b_3 .... $b_{\lceil \log_2 \rceil (\Psi)}$
8 bits arrangement
8 bits arrangement
8 bits arrangement
8 bits arrangement
b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8
b_9 b_{10} b_{11} b_{12} b_{13} b_{14} b_{15} b_{16}
......
$b_{\lceil \log_2 \rceil (\Psi) - 7}$ ... ... $b_{\lceil \log_2 \rceil (\Psi)}$
$B_1 \in Z_{256}$
$B_2 \in Z_{256}$
... ...
$B_{\lceil log_2 \rceil (\Psi)/8} \in Z_{256}$

Fig. 5 Cipher-text byte conversion procedure

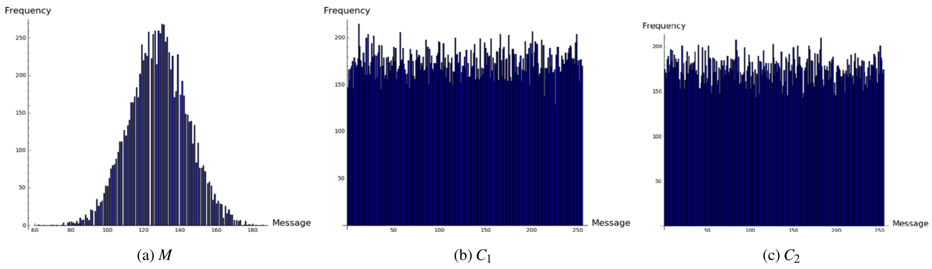


Fig. 6 Distribution test for a random original message M (a) and for two cipher messages C_1 (b) and C_2 (c) obtained by using the proposed HHSES scheme with a random secret key

test) and the correlation between the original and the encrypted data. All these tests are applied in the following sections to prove the resistance of the scheme against the statistical attacks.

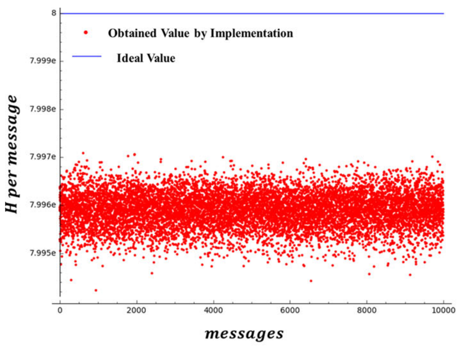
4.1.1 Uniformity property

1. **Distribution Test:**
   To guard against statistical attacks, a good crypto-system should give a distribution close to uniform among the cipher-texts. The distributions of the plain-texts and their correspondent cipher-texts are given in Fig. 6. The plain-texts' distribution is Gaussian where different data messages are close to each other (low standard deviation as given previously). After applying the HHSES scheme over a plain-texts message M of size 16 Bytes, one can visually detect that the two cipher-texts' distributions are close to uniform and no clear pattern is shown. Thus, an attacker is unable to reveal the plain-texts' distribution after drawing the two cipher-texts' distributions. Hence, the scheme is resistant against statistical attacks.
2. **Entropy Test:**
   The entropy is used to measure the level of uncertainty in a random variable. The entropy of a message m is calculated by the following equation:

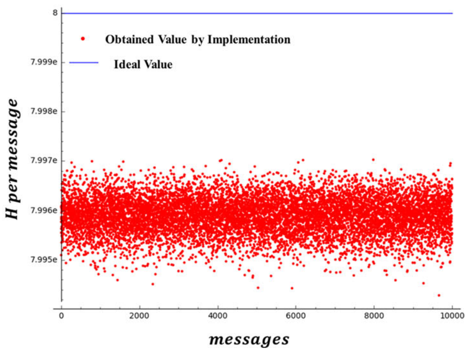
$$\sum_{i=0}^{2^{\Theta}-1} p(m_i) log_2 \frac{1}{p(m_i)} \tag{31}$$

$p(m_i)$ represents the probability of occurrence of symbol m_i and 2^{Θ} is the total states of information source. A truly random entropy is equal to Θ. As given in Fig. 5, the cipher-texts are transformed into the Byte level ($\mathbb{Z}_{256}$, $2^8 = 256$), thus the resultant cipher-texts of the HHSES are considered as a truly random source if the entropy values are close to 8. In Fig. 7, the entropy values are calculated for both cipher-texts C_1 and C_2 for 10000 iterations and for a plain-texts message M of 16 Bytes.

In the presented figure, the red dots present the different entropy values obtained by implementing the entropy tests over the cipher-texts and the blue line represents the ideal value (equal to 8 in our case). The obtained results gave for cipher-text C_1 a mean value equal to 7.996 with a low standard deviation equal to 0.000361 and for cipher-text C_2 a mean value equal to 7.9959 with a low standard deviation equal to 0.0003666 as given in Table 2. After analyzing the obtained results, one can deduce that the different



(a) C_1



(b) C_2

Fig. 7 Variation of the entropy test for the obtained cipher-text $C1$ (a) and C_2 (b) for 10000 random secret keys

obtained entropy values are close to the ideal value (8) for both cipher-texts. Thus, the HHSES scheme presents the required uniformity.

4.1.2 Independence property

As mentioned previously, several tests can be done to prove the independence criterion which are the recurrence, correlation and difference tests.

1. **Recurrence Test:**
   Starting from a data sequence $x_i = x_{(i,1)}, x_{(i,2)}, x_{(i,3)}, ..., x_{(i,m)}$ and vector with delay $t \geq 1$ constructed by $x_i(t) = x_{(i,t)}, x_{(i,2t)}, x_{(i,3t)}, ..., x_{(i,mt)}$, the recurrence test is achieved by calculating the correlation among these two sequences to measure the evolution of randomness. Figure 8a represents the correlation among $x_i(t)$ and $x_i(t+1)$ for the plain-text while Fig. 8b and c are for both cipher-texts C_1 and C_2. For the plain-texts, we used a Gaussian distribution with a mean value equal to 128 with a low standard deviation as given in Fig. 8a. After applying the HHSES over a plain-texts'

Table 2 Statistical analysis result for the different security tests

Security test	Cipher-text	Min	Max	Mean	Std
Entropy Test	C_1	7.9942	7.997	7.996	0.000361
Entropy Test	C_2	7.9943	7.997	7.9959	0.0003666
Difference Test	C_1	49.683	50.313	49.998	0.08299
Difference Test	C_2	49.6563	50.3506	49.9977	0.08345
Correlation Test	C_1	−0.01681	0.01809	$9.10129e-05$	0.0047077
Correlation Test	C_2	−0.019713	0.01575	0.000124	0.004748
Key Sensitivity	C_1	49.655	50.3194	50.00208	0.082217
Key Sensitivity	C_2	49.69615	50.35781	50.002986	0.084065
Plain-text Sensitivity	C_1	49.68086	50.319984	49.99929	0.08173
Plain-text Sensitivity	C_2	49.649337	50.374259	50.00017	0.0845123

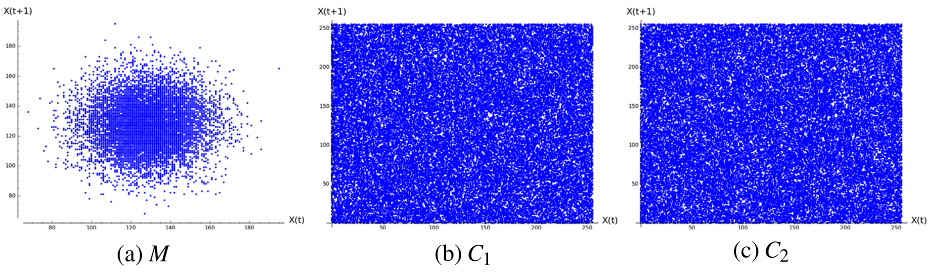


Fig. 8 Recurrence test for a random original message M (a) and for two cipher messages C_1 (b) and C_2 (c) obtained by using the proposed HHSES scheme with a random secret key

message M of size 16 Bytes, Fig. 8b and c present a high level of randomness among the cipher-texts and no clear pattern is shown after the encryption process.

2. **Difference Test:**
   The difference test consists of calculating the difference at the bit level between the plain-texts and their correspondent cipher-texts. To achieve the independence property, a good crypto-system should at least give 50% difference at the bit level between the plain-texts and the cipher-texts. In Fig. 9, the difference test is applied for a plain-texts message M of size 16 Bytes for 10000 iterations. As given in Table 2, mean value for cipher-text C_1 is equal to 49.998 with a low standard deviation equal to 0.08299 and for cipher-text C_2 the mean value is equal to 49.9977 with a low standard deviation equal to 0.08345. Thus, one can deduce that difference values at the bit level between cipher-texts and plain-texts (red dots) are close to the ideal value (50: blue line) and the new scheme provides the independence property.
3. **Correlation Test:**
   The correlation coefficient between a cipher-text message y and its correspondent

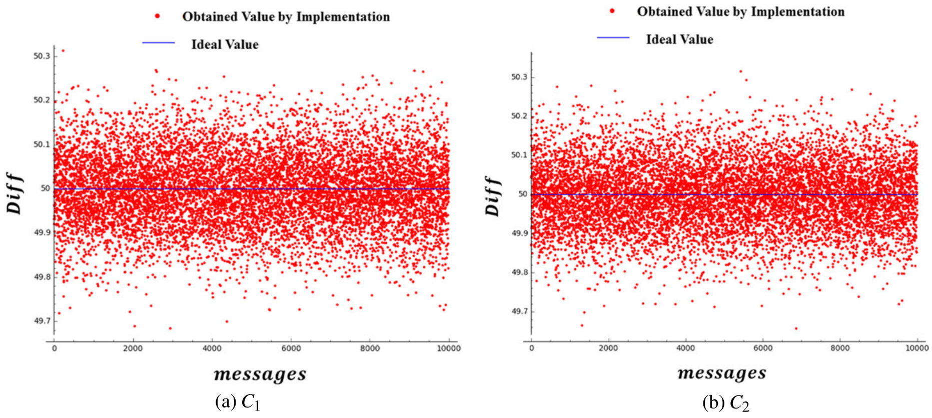


Fig. 9 Variation of the difference test with respect to the plain-text M for the obtained cipher-text $C1$ (a) and C_2 (b) for 10000 random secret keys

plain-text message x is given by the following equation:

$$\rho_{x,y} = \frac{cov(x,y)}{\sqrt{D(x) \times D(y)}} \tag{32}$$
$$\text{where } cov(x,y) = E[\{x - E(x)\}\{y - E(y)\}];$$
$$E(x) = \frac{1}{n} \times \sum_{k=1}^{n} x_i$$
$$\text{and } D(x) = \frac{1}{n} \times \sum_{k=1}^{n} \{x_i - E[x]\}^2$$

A good crypto-system that satisfies the independence property should present a low correlation (close to zero) between the plain-texts and the cipher-texts distribution. In Fig. 10, the correlation for 10000 iterations for a plain-text message M of size 16 Bytes with its corresponding cipher-texts is calculated. As presented in Table 2, the correlation's mean value between cipher-text C_1 and the plain-texts message M is equal to 9.10129×10^{-5} with a low standard deviation equal to 0.0047077. For cipher-text C_2, the correspondent mean value is equal to 0.000124 with a low standard deviation equal to 0.004748. Hence, the resultant scheme provides the required property of in-dependency, since different correlation values (red dots) are close to the ideal correlation value which is zero (blue line).

4.2 Resistance against related key attacks

A crypto-system that ensures a high resistance against related key attacks should give a high level of Key Sensitivity (KS) where the cipher-text should ensure a (KS) result close to 50. Thus, the KS test consists on calculating the difference between the cipher-texts at the bit level after doing a slight change in the encryption key. Indeed, the sensitivity of w^{th} secret key $K^{`}_w$ is calculated as follows:

$$KS_w = \frac{\sum_{k=1}^{T} E_{K_w} \oplus E_{K'_w}}{T} \times 100\%, \tag{33}$$

where all the elements of $K^{`}_w$ are equal to those of K_w, except a random Least Significant Bit (LSB) of a random byte, T is the length of the original and cipher-text packets (in bits), and $w = 1, 2, \ldots, 10000$. Figure 11 presents the result for 10000 iterations for a plain-texts' message M of 16 Bytes. Mean values are close to 50 with a low standard deviation as given in Table 2 (C_1: Mean Value=50.00208, Std=0.082217 and C_2: Mean Value=50.002986, Std=0.084065). Thus, KS values obtained by implementation (red dots) are concentrated close to the ideal KS value (50), and the scheme provides the required Key Sensitivity property.

4.3 Resistance to known plain-text/cipher-text attacks

In this section, the resistance of the HHSES against the known plain-text/cipher-text attacks is first evaluated by testing its avalanche effect while using the Plain-text Sensitivity (PS) test. Second, a theoretical crypt-analysis shows mathematically that the concerned scheme is immune against the concerned attack even with the lowest DF dimension $d = 2$.

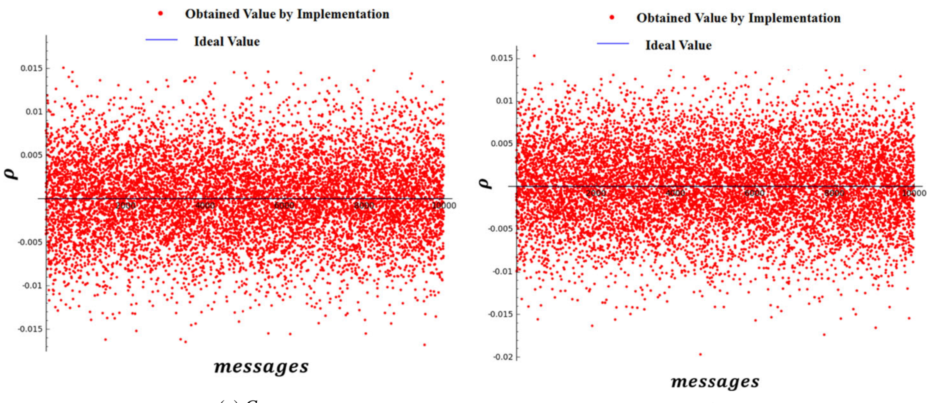


Fig. 10 Variation of the correlation test with respect to the plain-text M for the obtained cipher-text C_1 (a) and C_2 (b) for 10000 random secret keys

4.3.1 Avalanche effect

A good crypto-system should satisfy the avalanche effect. The avalanche effect can be interpreted by a significant change in the cipher-text due to a slight change in the plain-text. A good measure for the avalanche effect is the Plain-text Sensitivity (PS) test which gives the difference at the bit level between the resultant cipher-texts of two plain-texts that differ only in 1 bit. A crypto-system that satisfies the avalanche effect should at least give 50% difference. Figure 12 gives the result of this test for 10000 iterations for a plain-texts message M of 16 Bytes. Mean values are close to 50 with a low standard deviation as given in Table 2 (C_1: Mean Value= 49.99929, Std=0.08173, C_2: Mean Value=50.00017, Std=0.0845123). One can deduce that the HHSES presents the avalanche effect since different PS values (red dots) are concentrated close to the ideal value (50) (blue line).

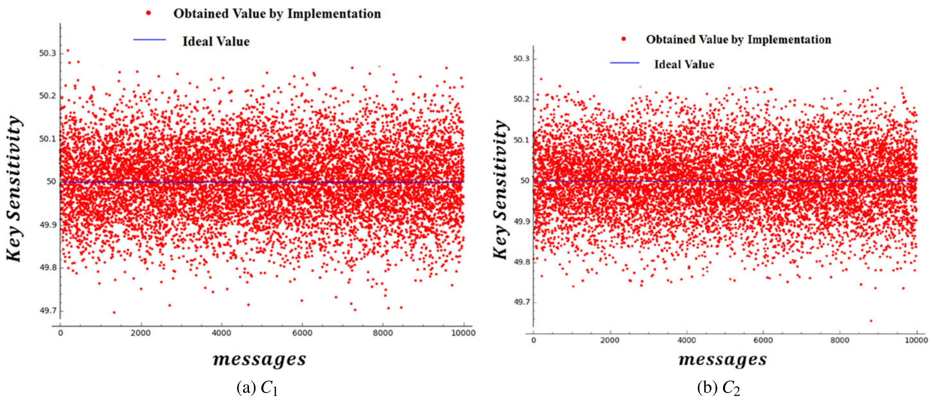
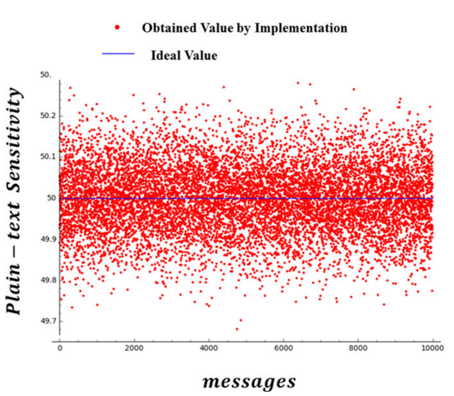
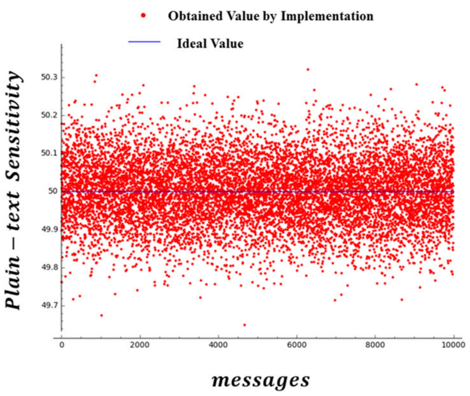


Fig. 11 Variation of the KS test for the obtained cipher-text C_1 (a) and C_2 (b) for 10000 random secret keys.



(a) C_1



(b) C_1

Fig. 12 Variation of the PS test for the obtained cipher-text C_1(a) and C_2 (b) for 10000 random secret keys

4.3.2 Theoretical crypt-analysis

As mentioned previously, HHSES is a new symmetric HE scheme based on mixing the homomorphic behavior of two well known symmetric HE schemes: the MORE approach and the DF scheme.

As given in Sections 2.2.1 and 2.2.3, MORE approach and DF scheme are sensible to known plain-text/cipher-text attacks where revealing secret parameters for both is possible for each with a certain probability.

In this part, it is shown theoretically that the HHSES is resistant against the known plain-text/cipher-text attacks. It is also demonstrated that the implementation of the new scheme is secure even with the lowest possible DF dimension ($d = 2$).

Having Θ plain-texts' vectors $X^j = [x_1^j, x_2^j]$ where $1 \leq j \leq \Theta$ with their respective cipher-texts:

$$C^j = [C^{j,(1)} = \begin{bmatrix} C_{11}^{j,(1)} & C_{12}^{j,(1)} \\ C_{21}^{j,(1)} & C_{22}^{j,(1)} \end{bmatrix} = K^{-1} \begin{bmatrix} x_1^{j,(1)} r & 0 \\ 0 & x_2^{j,(1)} r \end{bmatrix} K,$$

$$C^{j,(2)} = \begin{bmatrix} C_{11}^{j,(2)} & C_{12}^{j,(2)} \\ C_{21}^{j,(2)} & C_{22}^{j,(2)} \end{bmatrix} = K^{-1} \begin{bmatrix} x_1^{j,(2)} r^2 & 0 \\ 0 & x_2^{j,(2)} r^2 \end{bmatrix} K] \quad mod(\Psi)$$

We suppose the existence of an attacker who knows the Θ couples of plain-text/cipher-text ($X^j = [x_1^j, x_2^j], C^j = [C^{j,(1)}, C^{j,(2)}], \quad 1 \leq j \leq \Theta$) and who's trying to reveal the HHSES secret parameters (i.e. the secret modulus ψ, the secret invertible key r and the secret invertible matrix K) by doing the following two steps:

1. Revealing ψ and r: in order to reveal the secret parameters ψ and r, the attacker will try to calculate the $(4 \times \Theta)$ DF values that are $(x_1^{j,(1)} r, x_2^{j,(1)} r, x_1^{(j,2)} r^2, x_2^{j,(2)} r^2)$ that form the eigen values of matrices $C^{j,(1)}$ and $C^{j,(2)}$ respectively. This can be done by calculating the roots of the $(2 \times \Theta)$ characteristic polynomials $P^{j,(1)}(\alpha)$ of $C^{j,(1)}$ and $P^{j,(2)}(\alpha)$ of $C^{j,(2)}$ that are given by:

$$P^{j,(1)}(\alpha) = \alpha^2 - \alpha(C_{11}^{j,(1)} + C_{22}^{j,(1)}) + C_{11}^{j,(1)} C_{22}^{j,(1)} - C_{21}^{j,(1)} C_{12}^{j,(1)} \quad mod(\Psi)$$
$$P^{j,(2)}(\alpha) = \alpha^2 - \alpha(C_{11}^{j,(2)} + C_{22}^{j,(2)}) + C_{11}^{j,(2)} C_{22}^{j,(2)} - C_{21}^{j,(2)} C_{12}^{j,(2)} \quad mod(\Psi)$$

Calculating the roots of $P^{j,(1)}(\alpha)$ and $P^{j,(2)}(\alpha)$ is a hard task due to Rabin's crypto-system that is reduced to the problem of factoring $\Psi = \psi \times \prod_{i=1}^{m}(p_i \times q_i)$. Thus revealing the $(4 \times \Theta)$ DF values $(x_1^{j,(1)}r, x_2^{j,(1)}r, x_1^{j,(2)}r^2, x_2^{j,(2)}r^2)$ is a hard task for the attacker, which makes revealing r and ψ also a hard task.

2. Revealing the Secret Invertible matrix K: the columns of the secret invertible matrix K are the eigen vectors of the $(2 \times \Theta)$ matrices $C^{j,(1)}$ and $C^{j,(2)}$. Calculating those eigen vectors while knowing the eigen values is done by applying (7). The attacker is unable to apply the procedure of (7) with HHSES even while knowing the Θ couples $(x^j = [x_1^j, x_2^j], C^j = [C^{j,(1)}, C^{j,(2)}])$. The latter hardness is based on the fact that the eigen values of $C^{j,(1)}$ and $C^{j,(2)}$ are not x_1^j and x_2^j, otherwise they are the $(4 \times \Theta)$ random values $((x_1^{j,(1)}r, x_2^{j,(1)}r, x_1^{j,(2)}r^2, x_2^{j,(2)}r^2))$ generated based on the DF encryption algorithm and secret parameters. In addition, revealing $((x_1^{j,(1)}r, x_2^{j,(1)}r, x_1^{j,(2)}r^2, x_2^{j,(2)}r^2))$ is a hard task due to Rabin's crypto-system as it is mentioned in the previous step.

Table 2 presented below shows the maximum, the minimum, the mean, and the standard deviation values of different security test results listed previously and implemented to evaluate the security level of the concerned scheme and its immunity against a big variety of attacks.

5 Experimentation and performance analysis

In this section, the performance of the new scheme HHSES is compared with three well known symmetric schemes: the MORE approach, the PORE approach, and the DF scheme. It is also compared with the famous asymmetric scheme BGV. The comparison is done in terms of execution time for the different basic cryptographic functions: encryption, decryption, homomorphic addition, and homomorphic multiplication. Different implementations are done under Python using SageMath library with a personal laptop having the following specifications: OS Ubuntu 14.04, RAM 3.9 GB, Processor Intel Core $i7 - 8550$U CPU @ 1.8 GHZ, 64 bit, Disk 24.1 GB.

5.1 Comparison with symmetric schemes

To accomplish the required comparison, the four different symmetric encryption schemes (HHSES, DF, MORE and PORE) are implemented with the same level of security (security parameter $\lambda = 20$), the same plain-texts message size l in Bytes (varied from 8 Bytes till 128 Bytes) and the same cipher-texts range of storage overhead. Different plain-text inputs are sampled as Bytes from the ring $\mathbb{Z}_{256}$. For different operations, the mean execution time is calculated for 50 iterations.

1. HHSES Performance Analysis: as it was proven in the previous section (HHSES Security Analysis), the concerned scheme is resistant against different types of attacks and especially the known plain-text/cipher-text attacks even with the lowest possible DF dimension $d = 2$. Thus, HHSES is implemented with DF dimension dimension $d = 2$. The evolution of different operations execution time of the HHSES in function of the plain-texts message size l is presented in the two tables below: Tables 3 and 4. KS is adopted for the HHSES in order to preserve the cipher-texts dimension after each homomorphic multiplication (Section 3.3). In Table 3, Mean Total Multiplication time refers

to the basic time of multiplication with the KS operation. In Table 4, different execution time values for multiplication as a function of the message size in Bytes provided with the public matrix M generation (Section 3.3) are present in details.

After examining different results published in both Tables 3 and 4, one can deduce that the execution times of different HHSES cryptographic and arithmetic functions and storage overheads increase with the increase of the message size l in Bytes.

Remark During the HHSES performance analysis, DF dimension is fixed to $d = 2$ based on the security analysis of the scheme. Starting from a security parameter $\lambda = 20$, implementation has shown that the HHSES cipher-text public modulus N (explained in Section 3) size in Bytes is 176 (parameter $m = 5$, prime integers p_i and q_i taken of size 16 Bytes and $N = \prod_{i=1}^{m} f_i$, $\{f_i\}_{i=1}^{m}$ where $f_i = p_i q_i$). Thus, the encryption of a plain-texts message of size l Bytes will output a cipher-text $C = [C_1, C_2]$ formed of 2 matrices (C_1) and (C_2) each of dimension $l \times l$. The cipher-text C size in memory will be $(l \times l \times 2 \times 176)$ Bytes. For example with $l = 8$ the cipher-text C in Bytes will be $8 \times 8 \times 2 \times 176 = 22528$ as mentioned in Table 3.

2. DF Performance Analysis: the evolution of different operations execution time for the DF scheme in function of the message size l is shown in Table 5. Parameter d represents the DF cipher-text dimension as given in Section 2.2.3.

   After examining different metrics given in Table 5, one can deduce that with the increase of the plain-texts message size l and the DF parameter d different execution times and storage overheads increase.

   As for the comparison between the HHSES and the DF schemes, different implementations for both schemes should be driven with the same level of security (i.e. security parameter λ is taken 20 for both HHSES and DF schemes) and the same level of cipher-texts storage overhead. Achieving the same level of storage overhead for both DF and HHSES while encrypting two plain-texts messages having the same size l in Bytes is accomplished as follows: during this implementation the secret modulus of DF scheme m' is taken 256 (i.e. plain-texts are taken a Bytes) and $\lambda = 20$, the public modulus is then $m = (256)^{20}$ (as explained in Section 2.2.3). The size of the resultant public modulus m in Bytes is then 20. Thus, starting from a plain-texts message of length l, the DF cipher-texts message size in Bytes will be $(l \times d \times 20)$. Hence, the correspondent DF parameter d required to generate a DF cipher-texts message having the same range of storage overhead for a given HHSES cipher-texts message (also related to a plain-texts message of size l) is calculated by applying the following formula $\lfloor \frac{Required\_Storage\_HHSES\_Overhead\_Bytes}{l \times 20} \rfloor$. For example as given in

Table 3 HHSES execution time in seconds ($\lambda = 20$)

Plain-text Message Size l in Bytes	Mean Enc. Time	Mean Dec. Time	Mean Add. Time	Mean Total Mult. Time	Cipher-text Size in Bytes
$l = 8, d = 2$	0.008484	0.01052056	0.003246	0.1741	22528
$l = 16. d = 2$	0.0375	0.0481	0.013	1.202	89600
$l = 32. d = 2$	0.23643	0.335032	0.053391921	8.50443	360448
$l = 64. d = 2$	1.85349	2.674	0.1836	64.313	1441792
$l = 128. d = 2$	13.6919	14.0495	0.99365	625.248	5767168

Table 4 HHSES KS time in seconds ($\lambda = 20$)

Plain-text Message Size l in Bytes	Mean Basic Mult. Time	Mean KS Time	Public Matrix M Generation
$l = 8, d = 2$	0.153	0.021	0.003192
$l = 16. d = 2$	1.12867	0.07366	0.003305
$l = 32, d = 2$	8.23958	0.2642	0.004334
$l = 64. d = 2$	63.1831	1.12604	0.00436
$l = 128, d = 2$	615.9825	9.2658	0.004094

Table 5, to achieve a DF cipher-texts of dimension close to the correspondent HHSES cipher-texts size (22528 Bytes as given in Table 3) for a plain-texts message of length $l = 8$ Bytes, $d = \lfloor \frac{22528}{8 \times 20} \rfloor = 140$.
A practical way to compare between the performance of both HHSES and DF schemes in terms of different metrics given in Tables 3 and 5 is to calculate the following ratio $\epsilon = \frac{HHSES\_Metric}{DF\_Metric}$ as given in Table 6 below:

By analyzing different results given in Table 6, one can deduce that with the same plain-texts message length (l varied from 8 till 128 Bytes), the same level of security ($\lambda = 20$) and the same range of storage overhead (Storage Overhead ϵ's are close to 1), the HHSES is performing better than the DF scheme in terms of execution time since the majority of the ϵ's values related to execution times are lower than 1. Given that the DF scheme suffers from vulnerability to the known plain-text/cipher-text attacks (Section 2.2.3) and the HHSES presents the immunity against this type of attacks (Section 4.3), it is clear that the HHSES scheme is more implementable and practical than the DF scheme in real world applications.

3. MORE Performance Analysis: different execution times for different functions related to the MORE approach while varying the plain-texts message size l in Bytes are presented in Table 7. $Matrix\_Dim$ represents the matrix dimension of the MORE approach as given in Section 2.2.1.

   It is clear from Table 7, that with the increase of the plain-texts message size l and the MORE matrix dimension $Matrix\_Dim$, different execution times and storage overheads will increase.

   Based on the MORE encryption procedure listed in Section 2.2.1, the encryption of plain-texts message of size l Bytes will output a cipher-texts message of size

Table 5 DF execution time in seconds ($\lambda = 20$)

Plain-text Message Size l in Bytes	Mean Enc. Time	Mean Dec. Time	Mean Add. Time	Mean Total Mult. Time	Cipher-text Size in Bytes
$l = 8, d = 140$	0.201165	0.025562	0.01922908	0.943	22400
$l = 16, d = 280$	2.01739	0.12126	0.086073139	7.73	89600
$l = 32, d = 560$	32.8328	0.4251	0.2976526	51.85	358400
$l = 64, d = 1120$	244.426	1.913	1.243	456.8201254	1433600
$l = 128, d = 2240$	1956.7412	6.464	3.8467	2926.433	5734400

Table 6 Ratio between HHSES and DF schemes

Plain-text Size l in Bytes	Enc. Time (ϵ)	Dec. Time (ϵ)	Add. Time (ϵ)	Mult. Time (ϵ)	Storage Overhead (ϵ)
$l = 8$	0.042174161	0.411608623	0.168805788	0.184645257	1.005714286
$l = 16$	0.018584378	0.397232907	0.151166787	0.155555897	1
$l = 32$	0.007201142	0.788167086	0.179376629	0.164038868	1.005714286
$l = 64$	0.007583038	1.397727968	0.14769073	0.140783538	1.005714286
$l = 128$	0.006997344	2.173398172	0.258313437	0.2136555	1.005714286

$(l \times Matrix\_Dim \times Matrix\_Dim \times N\_Size\_Bytes)$ Bytes where N_Size_Bytes is the size in Bytes of N (the MORE cipher-text public modulus). Implementation has shown, that with a security parameter $\lambda = 20$, public modulus N size in Bytes (N_Size_Bytes) is 159 (parameter $m = 5$, prime integers p_i and q_i are taken of size 16 Bytes and $N = \prod_{i=1}^{m} f_i$, $\{f_i\}_{i=1}^{m}$ where $f_i = p_i q_i$). Hence, to generate a MORE cipher-texts message with a storage overhead close to a given HHSES cipher-texts message while both having the same plain-texts message size l, $Matrix\_Dim$ should be estimated as follows: $\lfloor \sqrt{\frac{Required\_Storage\_HHSES\_Overhead\_Bytes}{l \times 159}} \rfloor$.
The comparison between both HHSES and MORE schemes in terms of different metrics presented in Table 3 and Table 7 is accomplished like the case of DF scheme and presented in Table 8 below:

As given in Table 8, for the same plain-texts message size l in Bytes, the same security parameter λ and the same range of storage overhead (Storage Overhead ϵ) for both MORE and HHSES schemes, MORE is performing better than the HHSES in terms of execution time since the majority of ϵ's values related to execution times are greater than 1. Based on the crypt-analysis of the MORE approach discussed in Section 2.2.1, the latter scheme is vulnerable to the known plain-text/cipher-text attacks while the HHSES presents simultaneously the efficiency in implementation and the high immunity against a big variety of attacks and specifically the known plain-text/cipher-text attacks.

4. PORE Performance Analysis: the variation of execution time of different cryptographic and analytic functions related to the PORE scheme while varying the plain-texts message size in Bytes is given in the following Table 9. N_bits represents the number of bits related to the PORE public modulus N (Section 2.2.2).

Table 7 MORE execution time in seconds ($\lambda = 20$)

Plain-text message Size l in Bytes	Mean Enc. Time	Mean Dec. Time	Mean Add. Time	Mean Total Mult. Time	Cipher-text Size in Bytes
$l = 8$, $Matrix\_Dim = 4$	0.014141	0.0075746	0.00346	0.00470318	20352
$l = 16$, $Matrix\_Dim = 6$	0.0482647	0.03507	0.01336	0.01893	91008
$l = 32$, $Matrix\_Dim = 8$	0.1493	0.127399	0.051996	0.079289	325632
$l = 64$, $Matrix\_Dim = 12$	0.678075	0.7081144	0.2134	0.3611	1465344
$l = 128$, $Matrix\_Dim = 17$	3.071547	3.59209	0.956166	1.771	5844736

Table 8 Ratio between HHSES and MORE schemes

Plain-text Size l in Bytes	Enc. Time (ϵ)	Dec. Time (ϵ)	Add. Time (ϵ)	Mult. Time (ϵ)	Storage Overhead (ϵ)
$l = 8$	0.599954741	1.388926148	0.937943908	37.02514896	1.106918239
$l = 16$	0.776799193	1.373520477	0.973883859	63.52906777	0.984528833
$l = 32$	1.583396129	2.629786402	1.02684747	107.2587309	1.106918239
$l = 64$	2.733463791	3.775903498	0.859997435	178.0769284	0.983927324
$l = 128$	4.457685251	3.911249871	1.039207039	353.0220788	0.986728571

After analyzing different values present in Table 9, one can deduce that with the increase of plain-texts message size l, different execution times and storage overheads will increase.

As for the comparison between the performance of HHSES and PORE schemes, the same principle adopted for both DF and MORE schemes is applied. The two schemes (HHSES and PORE) should be implemented with the same level of security ($\lambda = 20$), the same plain-texts message size l in Bytes and the same range of storage overhead. Given that the PORE cipher-text dimension is limited to 2 numbers ($C = (c_1, c_2)$), a practical way to control the PORE cipher-texts message size is to vary the size of the PORE public modulus N (explained in Section 2.2.2). Hence, encrypting a plain-texts message of size l Bytes under the PORE scheme will output a cipher-texts message having the same range of storage overhead of that of a given HHSES cipher-texts message related also to a plain-texts message of size l Bytes if the PORE public modulus N size in Bytes is estimated by using the following formula $\lfloor \frac{Required\_Storage\_HHSES\_Overhead\_Bytes}{2 \times l} \rfloor$. For example, starting from a plain-texts message size of $l = 8$ Bytes, to achieve under the PORE a cipher-texts message of size close to 22528 Bytes (HHSES cipher-texts message size in Bytes for $l = 8$ Bytes as given in Table 3) the size of the PORE public modulus $N = \lfloor \frac{22528}{2 \times 8} \rfloor = 1408$ Bytes (i.e $1408 \times 8 = 11264$ bits). The parameter m is taken equal to 5, both prime integers p_i and q_i are taken of size 140 Bytes in this case and $N = \prod_{i=1}^{m} f_i$, $\{f_i\}_{i=1}^{m}$ where $f_i = p_i q_i$. The comparison between the performance of HHSES and PORE is accomplished similar to that of DF and PORE and presented in Table 10.

Table 9 PORE execution time in seconds ($\lambda = 20$)

Plain-texts message Size l in Bytes	Mean Enc. Time	Mean Dec. Time	Mean Add. Time	Mean Total Mult. Time	Cipher-text Size in Bytes
8, $N\_bits = 11264$	0.00371348	0.00090168	$8.296e^{-}5$	0.0057458	22512
16, $N\_bits = 22400$	0.01741764	0.004438	0.00014442	0.03072498	89568
32, $N\_bits = 45056$	0.08871658	0.02221452	0.00036358	0.1560644	358400
64, $N\_bits = 90112$	0.49892978	0.12547544	0.0010088	0.90894358	1441792
128,$N\_bits = 180224$	4.59223732	1.36599674	0.00344232	8.34415842	5767168

Table 10 Ratio between HHSES and PORE schemes

Plain-text Size l in Bytes	Enc. Time (ϵ)	Dec. Time (ϵ)	Add. Time (ϵ)	Mult. Time (ϵ)	Storage Overhead (ϵ)
$l = 8$	2.284639	11.6677	39.127	30.3066	1.000710732
$l = 16$	2.152529	10.85388	90.094	39.1400	1.00035727
$l = 32$	2.665	15.08166	146.85	54.49310169	1
$l = 64$	3.7149	21.309	181.967	70.75549547	1
$l = 128$	2.9815	10.2852	288.658	74.93248734	1

Based on Table 10, one can deduce that the PORE approach is performing better than the HHSES in terms of execution times since all ϵ's values related to execution times are greater than 1. We recall that the PORE Approach is vulnerable to known plain-text/cipher-text attack (Section 2.2.2), where the attack can be driven while knowing only one couple of plain-text/cipher-text. On the other hand, the HHSES presents efficiency in implementation and high immunity against attacks.

5.2 Comparison with asymmetric schemes

In this Section, the performance of the HHSES scheme is compared with the well known asymmetric BGV scheme. Starting from the same level of security ($\lambda = 10$), a plain-texts message is taken from the ring $\mathbb{Z}_{256}$ where its size is varied from 2 Bytes till 7 Bytes. Hence, different results are published as follows:

1. HHSES Performance Analysis: implementations given in Tables 11 and 12 are similar to Tables 3 and 4 (the security parameter λ is taken 10 instead of 20). It is clear that with increase of the plain-texts message size l, different execution times of different operations and storage overheads will also increase.
2. BGV Performance Analysis: Table 13 presents the execution times of the BGV scheme for different operations are given (similar to the HHSES scheme). $Latt\_Dim$ represents the lattice dimension (BGV cipher-text dimension). In Table 14, detailed execution times for homomorphic multiplication (basic multiplication and KS technique) and public matrix M generation (Section 2.3) are shown. Based on these two tables, different execution times and storage overheads are increasing with the increase of the plain-texts message Size l and the lattice dimension ($Latt\_Dim$).

Table 11 HHSES execution time in seconds ($\lambda = 10$)

Plain-texts Message Size l in Bytes	Mean Enc. Time	Mean Dec. Time	Mean Add. Time	Mean Total Mult. Time	Cipher-texts Message Size in Bytes
$l = 2, d = 2$	0.00109	0.00122	0.0001778	0.00429	896
$l = 3, d = 2$	0.001476	0.00178	0.0003655	0.00983	1998
$l = 4, d = 2$	0.00206	0.00217	0.0006366	0.018067	3584
$l = 5, d = 2$	0.002784	0.00279	0.00098074	0.0323	5650
$l = 6, d = 2$	0.00378	0.0038	0.00187	0.0666	8064
$l = 7, d = 2$	0.00503	0.00516	0.002052	0.0866	10878

Table 12 HHSES KS time in seconds ($\lambda = 10$)

Plain-texts Message Size l in Bytes	Mean Basic Mult. Time	Mean KS Time	Public Matrix M Generation
$l = 2, d = 2$	0.00279784	0.00143208	0.003979
$l = 3, d = 2$	0.00725622	0.00257298	0.003569
$l = 4, d = 2$	0.01406394	0.00393076	0.003361
$l = 5, d = 2$	0.02672842	0.0055413	0.003339
$l = 6, d = 2$	0.0562989	0.01031014	0.003382
$l = 7, d = 2$	0.0754255	0.01118048	0.0034694

As mentioned in the previous Section 5.1, to achieve a good comparison between different encryption schemes, implementations should be accomplished with the same security parameter λ, the same plain-texts message size l in Bytes and the same range of storage overhead. Achieving for a BGV cipher-texts message the same size in Bytes of a given HHSES cipher-texts ($Required\_Storage\_HHSES\_Overhead\_Bytes$) starting from the same plain-texts message size l is done as follows: based on [4], a secure implementation of the BGV scheme is assured as long as $Latt\_Dim \cong poly(\lambda)$ and the BGV public modulus $q \cong poly(Latt\_Dim)$. Hence starting from $Latt\_Dim \cong poly(\lambda)$ and $q \cong (Latt\_Dim)^g$ where $g \in \mathbb{Z}$, for a plain-texts message of size l in Bytes, the resultant cipher-texts message size in Bytes after applying the BGV scheme is equal to $\lceil \frac{l \times Latt\_Dim \times log(Latt\_Dim)^g}{8 \times log(2)} \rceil$. Thus, to achieve the required level of storage overhead for a BGV cipher-texts message, g is estimated by the following formula $g = \lfloor \frac{Required\_Storage\_HHSES\_Overhead\_Bytes \times 8 \times log(2)}{l \times Latt\_Dim \times log(Latt\_Dim)} \rfloor$. For example, as given in Table 13 for $l = 2$ and $Latt\_Dim = 13$ to achieve for BGV a cipher-texts message of size 896 Bytes ($Required\_Storage\_HHSES\_Overhead\_Bytes$ given in Table 11), $\lfloor g = \frac{896 \times 8 \times log(2)}{2 \times 13 \times log(13)} \rfloor = 74$ and $q \cong (13)^{74}$.
The comparison between the HHSES and the BGV schemes is accomplished a given in Tables 6, 8 and 10 and given in Tables 15 and 16.

Based on the results given in Tables 15 and 16, it is obvious that the HHSES is performing better than the BGV scheme in terms of execution time given that different ϵ's values related

Table 13 BGV execution time in seconds ($\lambda = 10$)

Plain-texts Message Size l in Bytes	Mean Enc. Time	Mean Dec. Time	Mean Add. Time	Mean Total Mult. Time	Cipher-text Size in Bytes
$l = 2$, $Latt\_Dim = 13$	4.829757	0.00158	0.003559	2.077387	858
$l = 3$, $Latt\_Dim = 18$	16.4838	0.0023671	0.006197	6.85549	1998
$l = 4$, $Latt\_Dim = 23$	42.5665	0.003132	0.01189	17.1927	3680
$l = 5$, $Latt\_Dim = 27$	77.7437	0.004255	0.01932	40.8725	5670
$l = 6$, $Latt\_Dim = 33$	140.04524	0.04189	0.0257078	66.29662	8910
$l = 7$, $Latt\_Dim = 36$	224.43017	0.05372	0.0302415	90.92897	11592

Table 14 BGV KS time in seconds ($\lambda = 10$)

Plain-texts Message Size l in Bytes	Mean Basic Mult. Time	Mean KS Time	Public Matrix M Generation
$l = 2$, $Latt\_Dim = 13$	0.04309216	2.03428524	9.247966
$l = 3$, $Latt\_Dim = 18$	0.11542216	6.7400588	30.241611
$l = 4$, $Latt\_Dim = 23$	0.26575618	16.9269577	65.136014
$l = 5$, $Latt\_Dim = 27$	0.48831334	40.38422894	124.276685
$l = 6$, $Latt\_Dim = 33$	0.8502763	65.44632018	254.536334
$l = 7$, $Latt\_Dim = 36$	1.14594242	89.78301398	295.984628

to execution times are lower than 1. Given that both schemes are immune against the known plain-text/cipher-text attacks, one can conclude that the HHSES is considered as a good competent for the well known BGV scheme.

5.3 Multiplicative circuit evaluation

Another way to compare the performance of the different encryption schemes (DF, HHSES and BGV) is to evaluate the multiplicative circuit of depth l given in Fig. 13. The 3 schemes listed previously suffer from cipher-text expansion after homomorphic multiplication and KS technique can be adopted as a practical solution to reduce this expansion and improve their efficiency.

In Fig. 14, the circuit evaluation using the HHSES is compared with an evaluation that adopts the basic DF scheme given in [7] and another version of it that adopts the DF scheme implemented with KS as presented in [16]. Different schemes are implemented with the same security level (security parameter $\lambda = 60$) over a plain-text message of size 2 Bytes and by varying the circuit depth l from 2 till 14 and taking the DF dimension d respectively 2, 3 and 5 for DF and HHSES. Different execution times in Fig. 14 represent the mean values of 10 iterations and illustrated in seconds using a log scale. After examining different results, it is clear that with the increase of the circuit depth different execution times are increasing linearly. A deep analysis of the concerned results shows that the main importance of the HHSES comes with complex circuits of high depth ($l \geq 6$) since DF with KS takes the lowest execution time afterward comes HHSES and the highest execution time is preserved for DF without KS. The main contribution of HHSES is that with circuits of high depth, such as the case of real-life applications, the latter is a symmetric scheme that provides, simultaneously, the efficiency in implementation and the immunity against different types

Table 15 Ratio between HHSES and BGV schemes

Plain-text Size l in Bytes	Enc. Time (ϵ)	Dec. Time (ϵ)	Add. Time (ϵ)	Mult. Time (ϵ)	Storage Overhead (ϵ)
$l = 2$	0.000226073	0.775642726	0.049963477	0.002066192	1.044289044
$l = 3$	$8.95206e - 05$	0.750403447	0.058970644	0.001433965	1
$l = 4$	$4.84253e - 05$	0.695759897	0.053540099	0.001050851	0.973913043
$l = 5$	$3.582e - 05$	0.65758614	0.050757108	0.000790416	0.996472663
$l = 6$	$2.70033e - 05$	0.090837996	0.072788025	0.001004712	0.905050505
$l = 7$	$2.24536e - 05$	0.096105018	0.067859729	0.000952521	0.938405797

Table 16 Ratio between BGV and HHSES for KS operation

Plain-texts Message Size l in Bytes	Mean Basic Mult (ϵ)	Mean KS (ϵ)	Public Matrix Generation (ϵ)
$l = 2$	0.064926892	0.000703972	0.000430257
$l = 3$	0.062866784	0.000381744	0.000117719
$l = 4$	0.052920463	0.000232219	$5.15997e-05$
$l = 5$	0.054736207	0.000137214	$2.68675e-05$
$l = 6$	0.066212477	0.000157536	$1.32869e-05$
$l = 7$	0.065819625	0.000124528	$1.17202e-05$

of attacks including the known plain-text/cipher-text attacks. On the other hand, DF with KS is efficient but it is compromised where a known plain-text/cipher-text attack is possible while knowing at least d couples of plain-text/cipher-text as given in [16, 35].

In Fig. 15, the evaluation procedures of the circuit given in Fig. 13 using both asymmetric BGV and symmetric HHSES are illustrated. Starting from the same security parameter ($\lambda = 10$) and a plain-text message of size 2 Bytes for both, the circuit depth is varied from 1 till 5. As for HHSES, DF dimension d is varied between 10, 25, and 50 respectively. For BGV the lattice dimension is chosen 5, 7, and 10 respectively. Different results in Fig. 15 also present the mean execution time of 10 iterations given in seconds and illustrated using a log scale. It is clear from the given results that with the increase of circuit depth l, execution times are increasing linearly. A deep examination of the obtained results shows that HHSES with $d = 10$ is taking the lowest execution time then come respectively HHSES with $d = 25$, BGV with lattice dimension=5, HHSES with $d = 50$, BGV with lattice dimension=7 and finally BGV with lattice dimension=10 is taking the highest execution time. In conclusion, HHSES can be considered as a good candidate in comparison with the well known and famous BGV scheme. Hence, the proposed solution is a symmetric HE cipher, and it requires simple operations and a lower number of rounds. Thus, the proposed solution can ensure better efficiency compared to existing asymmetric ones. Moreover, the proposed

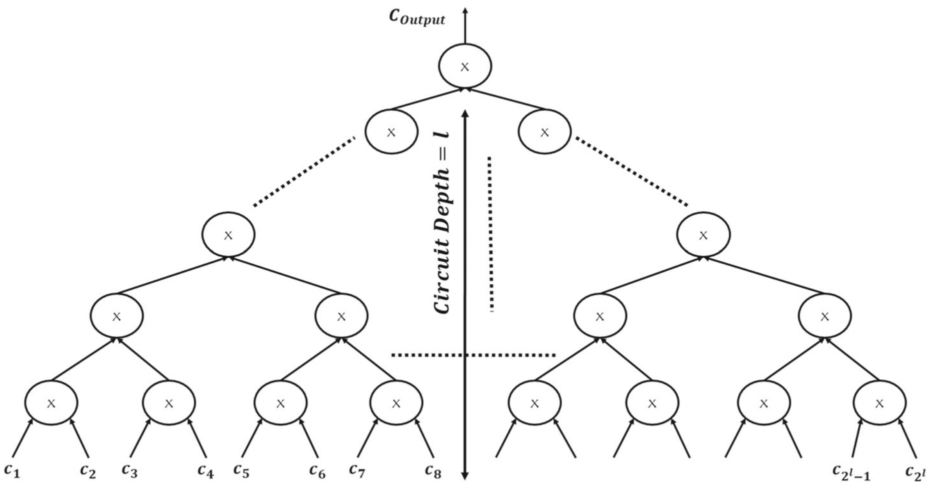


Fig. 13 Multiplicative circuit of depth l

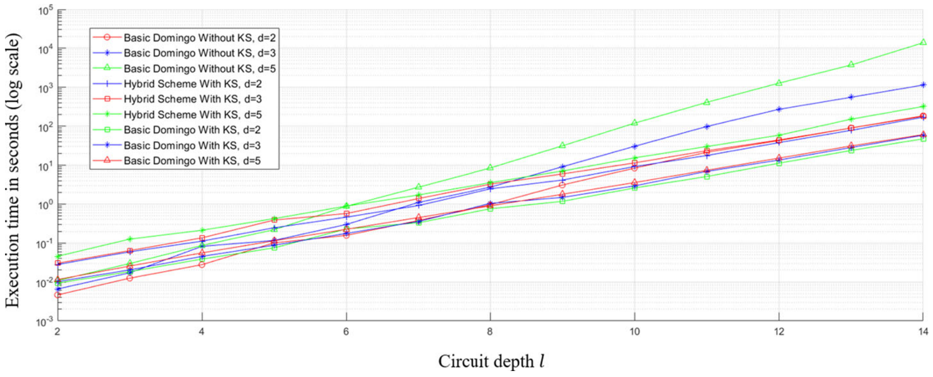


Fig. 14 Multiplicative circuit evaluation for HHSES and DF

solution reaches resistance against crypt-analysis as discussed in Section 4. Equally important, the storage overhead can be reduced according to the selected encryption parameters (configuration). This discussion indicates clearly that the proposed solution can reach a good level of efficiency and robustness.

6 HHSES optimization using CRT

HE schemes are characterized by having a high storage overhead in comparison with traditional encryption schemes, which reflects a high computational complexity in the encryption and decryption operations. Brakerski et al. [2] and Smart et al. [32] proposed an optimization technique for HE and decryption over an array of plain-texts instead of a single plain-text at a time. This technique uses the Chinese Remainder Theorem (CRT) [31] and factorizes any plain-text ring R_f into a product of many small prime factors where $f = \prod_{i=1}^{t} f_i$, where $f_1, f_2, ..., f_t$ are distinct prime numbers.

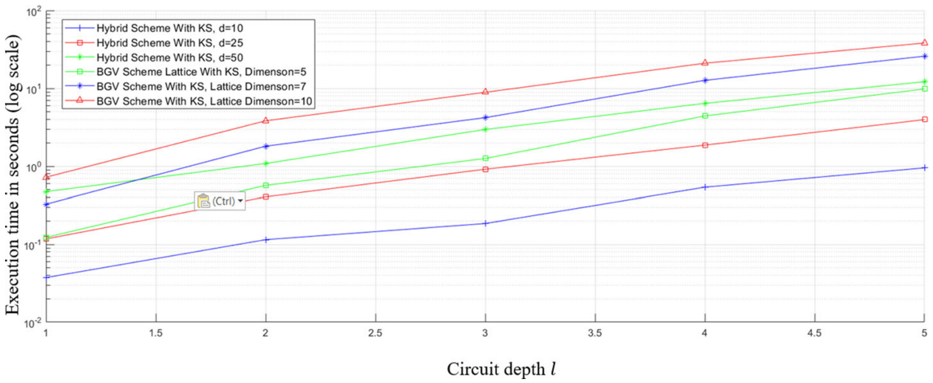


Fig. 15 Multiplicative circuit evaluation for HHSES and BGV

6.1 Chinese remainder theorem (CRT)

In this section, a brief overview about CRT is introduced as given in [31]: Given n_1, n_2, ..., n_t, t integers greater than 1 and $N = n_1 \times n_2 \times .... \times n_t$ the product of the n_i. The CRT asserts that if the n_i are pairwise co-prime and if a_1, a_2,...., a_t are t integers such that $0 \leq a_i \leq n_i$ for every i, then there is one and only one integer x, such that $0 \leq x < N$ and the remainder of the Euclidean division of x by n_i is a_i for every i.

The CRT can be written in term of congruence's: If the n_i are pairwise co-prime, and if a_1, a_2,..., a_t are any integers, then there exist integers x such that:

$$\begin{aligned} x &\equiv a_1 \quad (mod \quad n_1) \\ x &\equiv a_2 \quad (mod \quad n_2) \\ &\vdots \\ x &\equiv a_t \quad (mod \quad n_t) \end{aligned} \tag{34}$$

Any given two solutions, for example x_1 and x_2, are congruent modulo N (i.e $x_1 \equiv x_2 \quad (mod \quad N)$)
CRT enables homomorphic operations, since having two integers x and y in N that verify the two following relations:

$$\begin{array}{llll} x \equiv a_1 & (mod \quad n_1) & y \equiv b_1 & (mod \quad n_1) \\ x \equiv a_2 & (mod \quad n_2) & y \equiv b_2 & (mod \quad n_2) \\ \vdots & \vdots & \vdots & \vdots \\ x \equiv a_t & (mod \quad n_t) & y \equiv b_t & (mod \quad n_t) \end{array}$$

Both addition and multiplication properties are simply verified:

$$\begin{array}{llll} x + y \equiv (a_1 + b_1) & (mod \quad n_1) & x \times y \equiv (a_1 \times b_1) & (mod \quad n_1) \\ x + y \equiv (a_2 + b_2) & (mod \quad n_2) & x \times y \equiv (a_2 \times b_2) & (mod \quad n_2) \\ \vdots & \vdots & \vdots & \vdots \\ x + y \equiv (a_t + b_t) & (mod \quad n_t) & x \times y \equiv (a_t \times b_t) & (mod \quad n_t) \end{array}$$

6.2 HHSES under CRT

In order to enhance the HHSES efficiency, the latter scheme is implemented under CRT as follows:

1. Security Parameters:
   - (a) λ: security parameter, based on the security level.
   - (b) t: CRT's number of equations.
   - (c) ψ_i, $1 \leq i \leq t$: t secret co-primes pairwise modulus (ψ_1, ψ_2, ..., ψ_t) are generated such that $\psi_i = random\_prime(2^{high\_bound}, 2^{low\_bound})$.
   - (d) ψ: global secret modulus given by the product of the t (ψ_i) (i.e. $\psi = \psi_1 \times \psi_2 \times .... \times \psi_t$).
   - (e) N_i, $1 \leq i \leq t$: t RSA ring modulus are generated (N_1, N_2, .., N_t).
   - (f) Ψ_i: t public modulus where $\Psi_i = \psi_i \times N_i$, where $i \in \{1, 2, 3, .., t\}$.

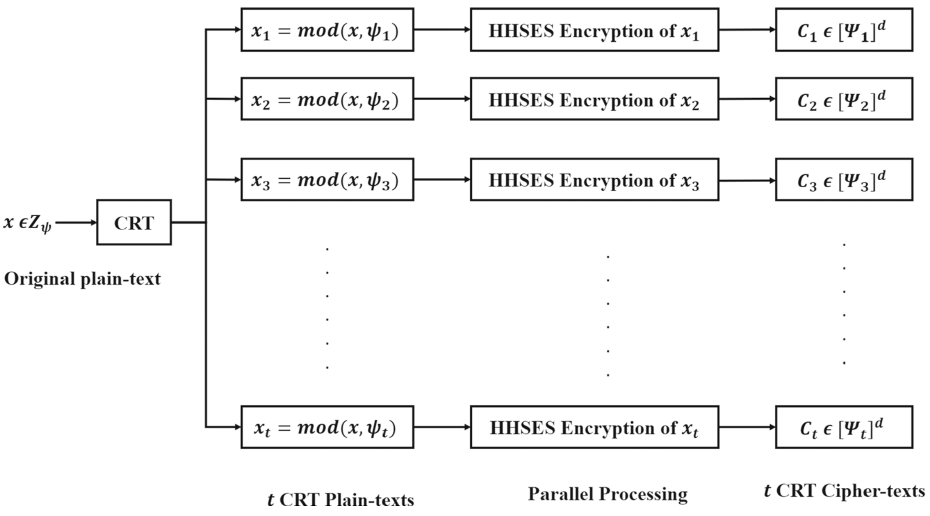


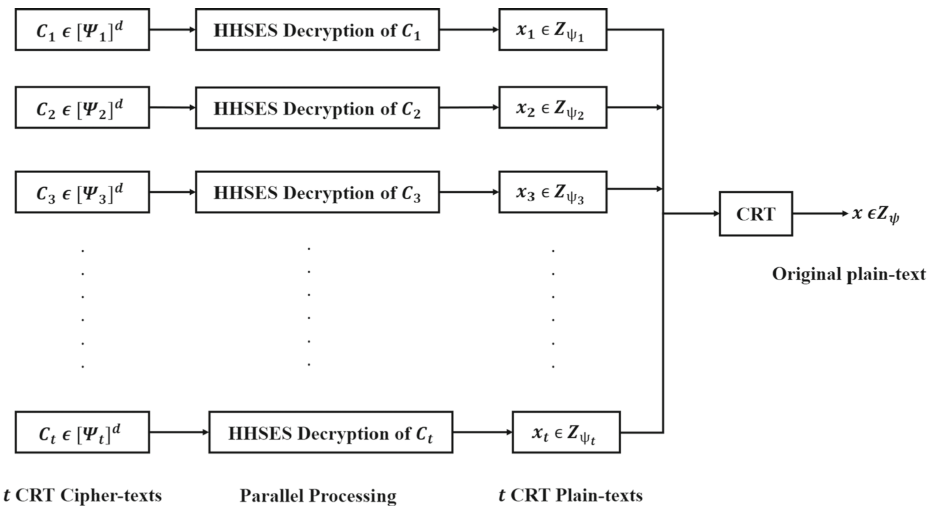
Fig. 16 Proposed HHSES parallel encryption scheme

(g) Ψ: global public modulus is given by the product of t public modulus Ψ_i. (i.e. $\Psi = \Psi_1 \times \Psi_2 \times .... \times \Psi_t$)
(h) r_i, $1 \le i \le t$: t invertible secret keys respectively in the public rings $\mathbb{Z}_{\Psi_i}$.
(i) K_i, $1 \le i \le t$: t invertible secret matrices respectively in the public rings (i.e $K_i = (k^i_{j,h} \in \mathbb{Z}_{\Psi_i})$ generated as given in Section 3.1.1.
(j) d: Cipher-text dimension is kept the same as given in Section 3.1.1.

2. Encryption Procedure: the computation with respect to the different secret modulus $(\psi_1, \psi_2, ..., \psi_t)$ are independent of each and hence encryption can be implemented in parallel as given in Fig. 16.
3. Decryption Procedure: similar to the parallel encryption procedure given in Fig. 16, decryption can also be implemented in parallel as given in Fig. 17.
4. Implementations: to validate the correctness of the HHSES under CRT for both encryption and decryption procedures, different implementations are done under Python using SageMath library. A personal laptop having the following specifications: OS Ubuntu 14.04, RAM 3.9 GB, Processor Intel Core $i7 - 8550$U CPU @ 1.8 GHZ, 64 bit, Disk 24.1 GB is used. For parallel processing, the multiprocessing pool of threads available in Python is adopted. In the following implementations, security parameter λ is taken equal to 10, t the number of CRT equations is taken equal to 6. Thus, 6 secret modulus ψ_i such that $1 \le i \le 6$ are generated as random primes between 2^{500} and 2^{100} ($high\_bound = 500$ and $low\_bound = 100$). Since 6 CRT equations are present, 6 processors are mandatory to implement the 6 processes of encryption and decryption in parallel to achieve the required optimization.

   Different results of optimization are given as follow:

   (a) Encryption' Optimization: in Fig. 18, the mean execution time for 40 iterations of HHSES encryption procedure with and without CRT are respectively calculated. In the related implementation, DF dimension d is taken 50 and the plain-text message size is varied from 10 to 30 with a step equal to 5. By examining the obtained result in Fig. 18, one can see that the parallel processing decreases the execution



$C_1 \epsilon [\Psi_1]^d$
HHSES Decryption of C_1
$x_1 \in Z_{\psi_1}$
$C_2 \epsilon [\Psi_2]^d$
HHSES Decryption of C_2
$x_2 \in Z_{\psi_2}$
$C_3 \epsilon [\Psi_3]^d$
HHSES Decryption of C_3
$x_3 \in Z_{\psi_3}$
CRT
$x \epsilon Z_\psi$
Original plain-text
$C_t \epsilon [\Psi_t]^d$
HHSES Decryption of C_t
$x_t \in Z_{\psi_t}$
t CRT Cipher-texts
Parallel Processing
t CRT Plain-texts

Fig. 17 Proposed HHSES parallel decryption scheme

time of encryption efficiently where ϵ represents the relative enhancement of the encryption procedure with CRT with respect to the encryption procedure without CRT as given below:

$$\epsilon = \frac{Execution\_time\_with\_CRT - Execution\_time\_without\_CRT}{Execution\_time\_with\_CRT} \quad (35)$$

While varying the plain-texts message size from 10 till 30 with a step equal to 5, ϵ's values increase respectively as follows: 0.166303856595, 0.3495, 0.4803, 0.51203 and 0.5895.

Figure 19 illustrates the same contribution for HHSES under CRT but with DF dimension $d = 60$.

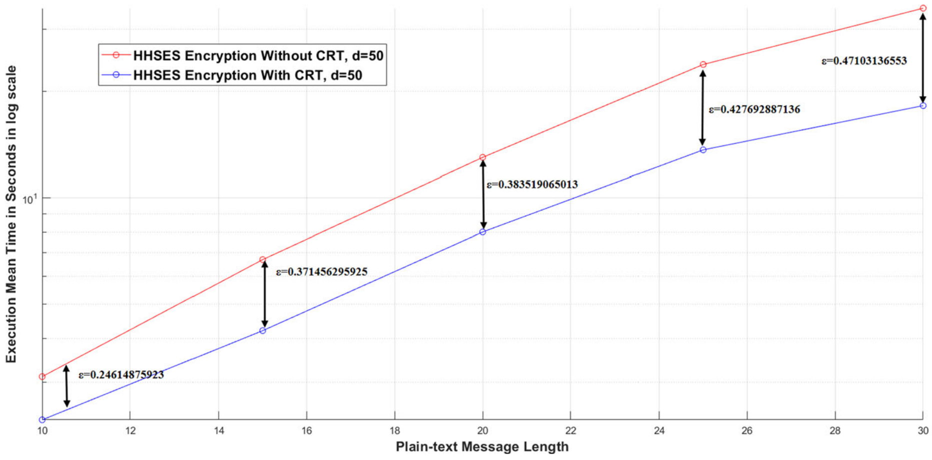


Fig. 18 Variation of HHSES encryption execution time with and without CRT for $d = 50$ (Parallel Processing)

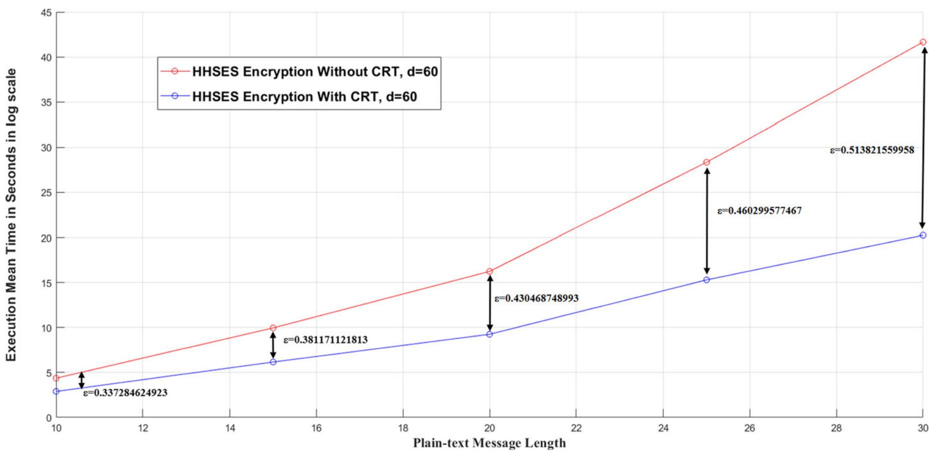


Fig. 19 Variation of HHSES encryption execution time with and without CRT for $d = 60$ (Parallel Processing)

(b) Decryption's Optimization: in Fig. 20, the mean execution time for 40 operations of decryption by taking $d = 50$ are illustrated. The plain-text message size is also varied from 10 to 30 with a step equal to 5. By examining different values of ϵ, it is obvious that decryption is also reduced efficiently using the parallel processing implementation since ϵ's values are increasing with the increase of the plain-texts message size as follows: 0.24027130944, 0.385004909377, 0.4572527777372, 0.500034926174 and 0.549887710517.

Figure 21 also illustrates the same contribution for HHSES under CRT with DF dimension $d = 60$.

An overall evaluation for both encryption and decryption optimization procedures leads us to conclude that with the increase of the plain-text message size, the relative enhancement ϵ tends to be close to 0.5. Hence, the proposed optimization improves twice the efficiency in implementation in terms of execution time.

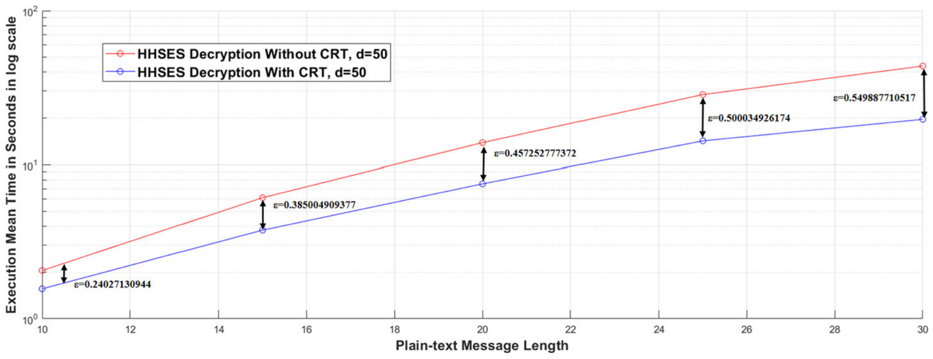


Fig. 20 Variation of HHSES decryption execution time with and without CRT for $d = 50$ (Parallel Processing)

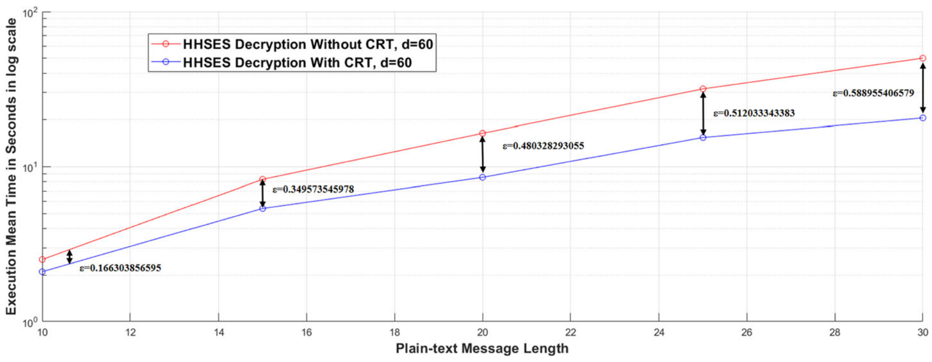


Fig. 21 Variation of HHSES decryption execution time with and without CRT for $d = 60$ (Parallel Processing)

7 Conclusion and future work

In this paper, we have designed the first fully symmetric HE scheme that resists the known plain-text/cipher-text attacks. As far as we know this is the first work in this direction. The new scheme is referred to as Homomorphic Hybrid Symmetric Encryption Scheme (HHSES). HHSES is based on mixing the homomorphic behavior of two well known symmetric encryption schemes, which are the MORE approach and the DF scheme. Different implementations and performance evaluations have shown that the new scheme (HHSES) is a good candidate in comparison to the well known symmetric (MORE, PORE and DF) and asymmetric (BGV and DGHV) encryption schemes. Security tests have shown that the new variant presents a high immunity against several types of attacks such as statistical attacks and related key attacks. The scheme also fulfills some important properties such as uniformity, in-dependency, and the avalanche effect. Theoretical crypt-analysis has shown that the scheme is robust against the known plain-text/cipher-text attacks even with the lowest possible dimension ($d = 2$), while other symmetric schemes (MORE, PORE and DF) are sensitive to this type of attacks. Another contribution in this work is optimizing the HHSES encryption and decryption procedures under CRT using parallel processing. The correctness of the optimization is also validated by the implementation. In conclusion, HHSES is a new efficient, robust, and practical symmetric HE scheme that can be adopted as a secure solution for a wide range of emerging and future applications. For future work, we will design a symmetric fully homomorphic message authentication to achieve data integrity and source authentication security services.

Acknowledgements This work has been funded by the EIPHI Graduate School (contract "ANR-17-EURE-0002").

Appendix A: Invertible Secret Matrix Generation

Different Steps for generating the invertible matrix K in [12] are given by the following: Starting from a matrix

$$k = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

its corresponding determinant $Det(k) = ad - bc$, assume that $Det(k) = ad - bc = 1$ and $a = d$. Under the conditions listed above we have:
$a^2 - bc = 1$, so $bc = a^2 - 1 = (a+1)(a-1)$, than we can write $b = a + 1$ and $c = a - 1$. As a result, the matrix k is given by the following form:

$$k = \begin{bmatrix} a & a+1 \\ a-1 & a \end{bmatrix} \tag{36}$$

Since $Det(k) = 1$, the matrix k is obviously invertible, and k^{-1} is

$$k^{-1} = \begin{bmatrix} a & -(a+1) \\ -(a-1) & a \end{bmatrix} \tag{37}$$

If the parameter a is replaced by sub matrix A, we get

$$K = \begin{bmatrix} A & A+I \\ A-I & A \end{bmatrix} \tag{38}$$

where I and A are the identity matrix and a non-zero matrix of size $n/2$ respectively. Additionally, the elements of A can be freely chosen from any Galois field such that K is full rank. However, having a matrix K constructed from four sub-matrices (A, B, C and D), $K = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$, the inverse of this matrix when $A = D$, $B = A + I$ and $C = A - I$ can be proven since the determinant of K is given by:

$$\begin{aligned} Det(K) &= Det(A) \times Det(D - CA^{-1}B) \\ &= Det(A) \times Det(A - (A - I)A^{-1}(A + I)) \\ &= Det(A) \times Det(A - (I - A^{-1})(A + I)) \\ &= Det(A) \times Det(A - (A + I - I - A^{-1})) \\ &= Det(A) \times Det(A^{-1}) \\ &= Det(A) \times \tfrac{1}{Det(A)} \\ &= 1 \end{aligned} \tag{39}$$

Since $Det(K) = 1$, the matrix K is always invertible and its inverse integer matrix K^{-1} is:

$$K^{-1} = \begin{bmatrix} A & -A-I \\ -A+I & A \end{bmatrix} \tag{40}$$

As a result, building a secret invertible matrix K of dimension $n \times n$, where n is always even is done by selecting a nonzero random sub matrix A of dimension $n/2$, and by applying the matrix forms listed respectively in (38) and (40), K and K^{-1} are obtained.

Appendix B: Attack on the Invertible Matrix Model

The vulnerability detected by the authors of [34] resides in using (38) while creating the invertible secret key K. As explained previously in Appendix A, the invertible matrix K has the following form: $\begin{bmatrix} A & A+I \\ A-I & A \end{bmatrix}$ where $dim(K)$ is $[n \times n]$ and $dim(A) = dim(I) = [\frac{n}{2} \times \frac{n}{2}]$, thus the attack is based on the the two following principles:

1. Principle 1: the attacker does not know the matrix A, but given a plain-text vector $m = [m_1, m_2, ..., m_n]$ with its corresponding cipher-text matrix C of dimension $[n \times n]$, he knows that the eigen-vector of C associated to the i^{th} plain-text m_i has the following

form:

$$\begin{bmatrix} A(i) \\ -A(i)+e_i \end{bmatrix} \text{ for } 1 \leq i \leq \frac{n}{2} \text{ and } \begin{bmatrix} -A(i-\frac{n}{2})-e_{i-\frac{n}{2}} \\ A(i-\frac{n}{2}) \end{bmatrix} \text{ for } \frac{n}{2} < i \leq n.$$

2. Principle 2: setting $C_{1,1}$, $C_{1,2}$, $C_{2,1}$ and $C_{2,2}$ four sub-matrices of $C = \begin{bmatrix} C_{1,1}, C_{1,2} \\ C_{2,1} C_{2,2} \end{bmatrix}$, the attacker can deduce the following equalities for $1 \leq i \leq \frac{n}{2}$ and $\frac{n}{2} < j \leq n$:

$$\begin{bmatrix} C_{1,1}, C_{1,2} \\ C_{2,1}, C_{2,2} \end{bmatrix} . \begin{bmatrix} A(i) \\ -A(i)+e_i \end{bmatrix} = m_i . \begin{bmatrix} A(i) \\ -A(i)+e_i \end{bmatrix} \tag{41}$$

$$\begin{bmatrix} C_{1,1}, C_{1,2} \\ C_{2,1}, C_{2,2} \end{bmatrix} . \begin{bmatrix} -A(j-\frac{n}{2})-e_{j-\frac{n}{2}} \\ A(j-\frac{n}{2}) \end{bmatrix} = m_j . \begin{bmatrix} -A(j-\frac{n}{2})-e_{j-\frac{n}{2}} \\ A(j-\frac{n}{2}) \end{bmatrix} \tag{42}$$

By linearizing the two (41) and (42) and setting up $j = i + \frac{n}{2}$, the attacker can build the following relation given by:

$$(-C_{1,1} + C_{1,2} - C_{2,1} + C_{2,2}) = (m_i - m_{i+\frac{n}{2}}).e_i \tag{43}$$

Let $D = (-C_{1,1} + C_{1,2} - C_{2,1} + C_{2,2})$. It is clear that for every $1 \leq i \leq \frac{n}{2}$, the $i-th$ column of D is the $i-th$ canonical vector multiplied by a difference of two eigen-values of C. Thus the matrix D, that can be computed using only one cipher-text, is a diagonal matrix, whose entries leak differences of eigen-values of C.

Revealing a plain-text vector, in this case, is given by the following steps:

1. Step1: let $\delta_i = D(i,i)$ denotes the $i-th$ entry on the diagonal of D.
2. Step2: let the characteristic polynomial $p(x) = CharPoly(C)(x)$ of the cipher-text C.
3. Step3: given that the roots of $p(x)$ are the elements of the plain-text vector m, the attacker defines n new polynomials $p_{-i}(x)$ and $p_i(x)$ for $1 \leq i \leq \frac{n}{2}$ as $p_{-i}(x) = p(x-\delta_i)$ and $p_i(x) = p(x+\delta_i)$
4. Step 4: for any $1 \leq i \leq \frac{n}{2}$, m_i must be a root of $p_{-i}(x)$, Thus m_i will also be root of $gcd(p(x), p_{-i}(x))$.

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 Proc Mag 30(2):108–117
2. Brakerski Z, Gentry C, Halevi S (2013) Packed ciphertexts in LWE-based homomorphic encryption. In: Kurosawa K, Hanaoka G (eds) Public-Key Cryptography – PKC 2013. Springer, Berlin, pp 1–13
3. Brakerski Z, Gentry C, Vinod V (2014) (Leveled) fully homomorphic encryption without bootstrapping. ACM Trans. Comput. Theory 6(3)
4. Brakerski Z, Vaikuntanathan V (2011) Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway P (ed) Advances in cryptology – CRYPTO 2011. Springer, Berlin, pp 505–524
5. Catalano D, Cramer R, Damgård IB (2005) Contemporary cryptology. Advanced courses in mathematics: CRM Barcelona Birkhäuser Di Crescenzo G, Pointcheval D (eds)

6. 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
7. Domingo-Ferrer J (2002) A provably secure additive and multiplicative privacy homomorphism. In: Chan AH, Gligor V (eds) Information security. Springer, Berlin, pp 471–483
8. Gentry C (2009) A fully homomorphic encryption scheme. PhD thesis, Stanford University. crypto.stanford.edu/craig
9. Gentry C (2009) Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on theory of computing, STOC '09. Association for Computing Machinery, New York, pp 169–178
10. Gentry C, Sahai A, Waters B (2013) Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti R, Garay JA (eds) Advances in cryptology – CRYPTO 2013, Springer, pp 75–92
11. Hariss K, Chamoun M, Samhat AE (2017) On DGHV and BGV fully homomorphic encryption schemes. In: 2017 1St cyber security in networking conference (CSNet), pp 1–9
12. Hariss K, Noura H, Samhat AE (2017) Fully enhanced homomorphic encryption algorithm of MORE approach for real world applications. Journal of Information Security and Applications 34:233–242
13. 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 J-L, Legay A, Garcia-Alfaro J (eds) Risks and Security of Internet and Systems. Springer International Publishing, Cham, pp 127–139
14. Im J, Choi J, Nyang D, Lee M (2016) Privacy-preserving palm print authentication using homomorphic encryption. In: 2016 IEEE 14Th intl conf on dependable, autonomic and secure computing, 14th intl conf on pervasive intelligence and computing, 2nd intl conf on big data intelligence and computing and cyber science and technology congress(DASC/picom/datacom/cyberscitech), pp 878–881
15. Katz J, Lindell Y (2014) Introduction to Modern Cryptography. Chapman & hall/CRC, 2nd edn
16. Khalil H, Samhat AE, Chamoun M (2019) An efficient fhe scheme to secure cloud computing. In: Proceedings of the 16th International Joint Conference on e-Business and Telecommunications - Vol 2: SECRYPT, INSTICC, SciTePress, pp 341–349
17. Kim Miran, Lauter KE (2015) Private genome analysis through homomorphic encryption. IACR Cryptol ePrint Arch 2015:965
18. Kim T, Oh Y, Kim H (2020) Efficient privacy-preserving fingerprint-based authentication system using fully homomorphic encryption. Commun Networks 2020:4195852:1–4195852:11
19. Kipnis A, Hibshoosh E (2012) Efficient methods for practical fully homomorphic symmetric-key encrypton, randomization and verification. IACR Cryptology ePrint Archive 2012:637
20. Leng L, Teoh ABJ, Li M, Khan MK (2014) Analysis of correlation of 2dpalmhash code and orientation range suitable for transposition. Neurocomputing 131:377–387
21. Leng L, Zhang J (2011) Dual-key-binding cancelable palmprint cryptosystem for palmprint protection and information security. J Netw Comput Appl 34(6):1979–1989
22. Leng L, Zhang J (2013) Palmhash code vs. palmphasor code. Neurocomputing 108:1–12
23. Liao Xin, Li Kaide, Yin Jiaojiao (2017) Separable data hiding in encrypted image based on compressive sensing and discrete fourier transform. Multimed Tools Appl 76(20):20739–20753
24. Liao X, Yin J, Chen M, Qin Z (2020) Adaptive payload distribution in multiple images steganography based on image texture features. IEEE Transactions on Dependable and Secure Computing
25. Liao X, Yu Y, Li B, Li Z, Qin Z (2019) A new payload partition strategy in color image steganography. IEEE Trans Circuits Syst Video Technol 30(3):685–696
26. Micciancio D, Peikert C (2013) Hardness of SIS and LWE with small parameters. In: Canetti R, Garay JA (eds) Advances in cryptology – CRYPTO 2013. Springer, Berlin, pp 21–39
27. Noura H, Chehab A, Sleem L, Noura M, Couturier R, Mansour MM (2018) One round cipher algorithm for multimedia IoT devices. Multimedia Tools and Applications 77(14):18383–18413
28. Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes. In: Stern J (ed) Advances in cryptology — EUROCRYPT '99. Springer, Berlin, pp 223–238
29. Regev Oded (September 2009) On lattices, learning with errors, random linear codes, and cryptography. j. ACM 56(6)
30. Rivest RL, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21(2):120–126
31. Schwarzweller C (2009) The chinese remainder theorem, its proofs and its generalizations in mathematical repositories. Studies in Logic, Grammar and Rhetoric, 18(31)
32. Smart NP, Vercauteren F (2014) Fully homomorphic SIMD operations. Des Codes Cryptography 71(1):57–81

33. Torres WAA, Bhattacharjee N, Srinivasan B (2015) Privacy-preserving biometrics authentication systems using fully homomorphic encryption. International Journal of Pervasive Computing and Communications 11(2):151–168
34. Vizár D, Vaudenay S (2019) Cryptanalysis of enhanced more. Tatra Mountains Mathematical Publications 73(1):163–178
35. Wagner D (2003) Cryptanalysis of an algebraic privacy homomorphism. In: CBoyd D, Mao W (eds) Information security. Springer, Berlin, pp 234–239
36. Xiao L, Bastani O, Yen I-L (2012) An efficient homomorphic encryption protocol for multi-user systems. IACR Cryptology ePrint Archive 2012:193
37. Yi X, Paulet R, Bertino E (2014) Homomorphic encryption and applications. Springer Publishing Company, Incorporated
38. van Dijk M, Gentry C, Halevi S, Vaikuntanathan V (2010) Fully homomorphic encryption over the integers. In: Gilbert H (ed) Advances in cryptology – EUROCRYPT 2010. Springer, Berlin, pp 24–43
39. Öztürk E, Doröz Y, Sunar B, Savas E (2015) Accelerating somewhat homomorphic evaluation using FPGAs. IACR Cryptology ePrint Archive 2015:294

Affiliations

Khalil Hariss[1,2] · Hassan Noura[3]

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

[2] ESIB-CIMTI, Saint Joseph University, Mar Roukoz, Lebanon

[3] FEMTO-ST Institute, Université Bourgogne Franche-Comté (UBFC), CNRS, Belfort, France