Noise Management in Homomorphic Encryption Scheme: Analyzing of Algorithm

https://doi.org/10.3991/ijes.v9i4.25993

Paulin Boale Bomolo(✉), Eugene Mbuyi Mukendi, Simon Ntumba Badibanga
Université de Kinshasa, Kinshasa, République Démocratique du Congo
paulin.boale@unikin.ac.cd

Abstract—Homomorphic encryption is the Swiss army of cryptography. It allows to perform computations on encrypted data. this conjecture of processing on encrypted data was stated by [13]. Until Gentry breakthrough in 2009, only partial solutions were offered. They deal with encrypted data with bounded functions in operations. Gentry raised the bound of number of operations by introducing a new method called the bootstrapping. This method allows to reduce the noise in the ciphertext and to perform more computing on it. Since Gentry's breakthrough, several improvements and several alternatives to the bootstrapping technique have been proposed to improve execution time and reduce resource consumption. This article studies the growth of noise and the noise management strategy in homomorphic encryption. It also presents the trend of hoping strategy from 2009 to 2016. Through the DGHV, it shows the management of noise on a one-bit encrypted message.

Keywords—noise, encryption homomorphic, bootstrapping, modulus switching, noise management

1 Introduction

Homomorphic encryption is based on noise cryptography. The ciphertext is obtained by masking the plaintext by a value called noise. To decrypt, it suffices only to remove this value in the cyphertexts to retrieve the original plaintext. But, in computations on the ciphertexts, this value may exceed the threshold and the immediate consequence is the decryption fails. The management of noise in homomorphic operations is a bottleneck in the practicability of these encryption in everyday applications. A bitter observation is that the noise increases with the depth of the circuit to be evaluated. Several techniques have been proposed by different authors during the period 2009 to 2016. Among these techniques, bootstrapping which makes it possible to reduce the size of the noise in the ciphertext by a homomorphic evaluation of the decryption circuit is the most used. The following lines describe the types of increasing noise in ciphertexts during homomorphic operations and present the most commonly used noise management strategies.

2 Homomorphic encryption scheme

When the encrypted data is stored on cloud [20,21], all kinds of computations [19] such as analyzing data, the calculation of variance, data mining, … are performed after prior decryption. The prior decryption brings an additional cost in terms of resources and time and introduces a possibility of loss of confidentiality and privacy. Homomorphic encryption offers an alternative to traditional cryptography by offering the power of performing operations on encrypted data without any prior conditions.

2.1 Definitions [10]

Definition 1. Public key encryption. A public key encryption is a triplet of polynomial time algorithms *keygen, encrypt and decrypt*. *Keygen* is an algorithm which takes as input the parameter λ and outputs a couple of keys (sk, pk) where pk is public key and sk is private key. The space of plaintext M is defined by public key pk and space of ciphertext C is defined by private key sk. *Encrypt* is a probabilistic algorithm which takes as inputs public key pk, the plaintext m and random value r. It outputs $c \in C$. It is noted as follows: $c \leftarrow Encrypt_{pk}(m,r)$. *Decrypt* is deterministic algorithm which takes as inputs the private key sk and the ciphertext c. It outputs the plaintext m. It is evaluated as follows: $m \leftarrow Decrypt_{sk}(c)$.

Definition 2. Homomorphic encryption. A homomorphic encryption scheme is a public encryption scheme which includes three in polynomial time *keygen, encrypt, decrypt* as described above to which added a fourth algorithm, the *evaluate*. The definition of which is given below: *Evaluate* is an algorithm which takes as inputs the public key pk, ciphertext vector c_i of size n and a function F. It outputs $F(c_1, c_2, \ldots\ldots\ldots\ldots\ldots, c_n) \in C$.

Axiome 1: free from error of homomorphic evaluation. Consider a set of ciphertexts $c_i = \{c_1, c_2, \ldots\ldots\ldots\ldots, c_n\}$ and correspondents decrypted plaintexts $m_i = \{m_1, m_2, \ldots\ldots\ldots\ldots\ldots, m_n\}$. The evaluation is said free from error if $Decrypt((F,\ c_i, pk)) = F(m_i)$.

2.2 Types of homomorphic encryption

There are four types of homomorphic encryption:

The partial homomorphic encryption. The partial homomorphic encryption is which the *encrypt* cannot that evaluating a limited number of operations defined in functions *f*. There is talk of an additive homomorphic it *encrypt* evaluates only additions [16] [15] while multiplicative homomorphic, it evaluates multiplications [16] on encrypted data. In some use cases, these schemes have proven to be useful primitives in constructing secure voting protocols.

The somewhat homomorphic encryption. The somewhat homomorphic encryption is which the said algorithm evaluates functions or polynomials of bounded degree, the latter being set by the parameters of the scheme used. It has a computational bounded capacity, but unlike partial homomorphic encryption, the polynomials or functions evaluated include additions and multiplications [18] [10]. [18] is an exception which allows to process encrypted messages with multiple additions and a single multiplication.

The leveled homomorphic encryption. The Leveled Homomorphic Encryption allows to algorithm *encrypt* to evaluate restrictive functions or polynomials. This evaluation deals with functions or polynomials setted by L which allows to evaluate polynomials or functions of degree $L - 1$. The size of used parameter increases linearly with L [4] [9].

The fully homomorphic encryption. The Fully homomorphic Encryption is in which the algorithm *encrypt* evaluate polynomial or function whose degree is arbitrary on the encrypted data [2]. The full Homomorphic Encryption is built from two previous ones using Gentry's technique [11] [12] or the alternative method [6] [8]. To date, no Pure Homomorphic Encryption has been discovered.

2.3 Properties of homomorphic encryption [10]

1. The scheme ε = (*keygen, encrypt, decrypt, evaluate*) is defined correct for any t inputs in circuit C if, for all generated couple of key (*sk*, *pk*) by *keygen*(λ), any plaintext t bits $(m_1, \ldots\ldots\ldots\ldots\ldots\ldots, m_t)$ and any vector ciphertext $\vec{c} = (c_1, \ldots\ldots\ldots\ldots, c_t)$ with $c_i \leftarrow encrypt_{\varepsilon}(pk, m_i)$, it is clear that $decrypt(sk, evaluate(pk, C, \vec{c})) = C(m_1, \ldots\ldots\ldots\ldots\ldots, m_t)$
2. The scheme ε = (*keygen, encrypt, decrypt, evaluate*) is homomorphic for a class of circuit if it is correct for all $C \epsilon \mathrm{C}$, ε is homomorphic for all Boolean circuit.
3. The scheme ε = (*keygen, encrypt, decrypt, evaluate*) is compact if there is fixed bound $b(\lambda)$ such that for any generated key (*sk*, *pk*) by *keygen*(λ), any circuit C and any sequence of ciphertexts $\vec{c} = (c_1, \ldots\ldots\ldots, c_t)$ which generated with respect of *pk*, the size of ciphertext $evaluate(pk, C, \vec{c})$ is less than $b(\lambda)$ bits (regardless the size of circuit C).
4. The scheme ε = (*keygen, encrypt, decrypt, evaluate*) is homomorphic encryption where the decryption is implemented by a circuit which depends only on the security parameter.
5. The scheme (*keygen, encrypt, decrypt, evaluate*) is homomorphic encryption, and for any security value λ be set $C_{\varepsilon}(\lambda)$ a set of circuits with which ε is correct. We say ε is bootstrappable if $Decrypt_{\varepsilon}(\lambda) \subseteq C_{\varepsilon}(\lambda)$ if is valid for each value of λ. It is clear that it is evaluating his own decryption algorithm.

3 The noise management

3.1 Definition of noise

Noise is a hazard introduced into the encryption and aims to destroy the determinism existing between the plaintext and the encrypted message "it is not obvious to know whether posteriori an encrypted message encrypts a given plaintext." It is a concept that was introduced by GoldWasser-Micali in 1982 [14].

(1) Axiom of the determinism of encryption: "An encrypted message corresponds posteriori to a plaintext."

(2) Axiom of the probabilism of encryption: "Several encrypted messages can correspond posteriori to a same plaintext."

The DGHV and the noise [11]

Algorithm of DGHV. Given security parameter λ, *Keygen* (λ) takes as input the security parameter λ and generates a key p which is an integer P bits size. The parameters N, P, Q respectively sets by λ, λ^2 *and* λ^5.

Encrypt(p,m) takes as inputs the key p and a plaintext $m \in \{0,1\}$. To encrypt m, choose a random r of *N–bits* such that $2r < p/2$ and choose randomly an integer q of *Q–bits*. It outputs a fresh ciphertext $c = m + 2r + pq$ where r is pattern noise that masks an actual plaintext m.

Decrypt(p,c) takes as inputs the key p and the ciphertext c. It outputs $mf = (c\, mod\ p) mod\ 2$. Here, $c = (c\, mod\ p)$ is in interval $\left[\frac{-p}{2}, \frac{p}{2}\right]$ with the prerequisite that p divides $c - c'$ without remainder.

Evaluate(f,c) takes as inputs a Boolean function f and a set of ciphertexts $S_c = \{c_1, \ldots\ldots\ldots\ldots, c_n\}$. f is represented by a circuit C composed by NAND and XOR gates.

Given a circuit C+ a copy of circuit C which the NAND and XOR are replaced respectively from addition and multiplication on integers.

Let be f^+ a multivariate polynomial which corresponds to circuit C+. It outputs $c = f^+(c_1, \ldots\ldots\ldots, c_n)$ or the evaluation of the function f^+ on ciphertexts c_i.

Homomorphic encryption operation and the noise in the DGHV. A homomorphic encryption performs arithmetic operations which are addition and multiplication. Let be two ciphertexts $c_1 = q_1 p + 2r_1 + m_1$ and $c_2 = q_2 p + 2r_2 + m_2$.

Let's calculate: $c_1 + c_2 = q_1 p + 2r_1 + m_1 + q_2 p + 2r_2 + m_2 = p(q_1 + q_2) + 2(r_1 + r_2) + m_1 + m_2$. The evaluation of $c_1 + c_2$ produce a valid decryption of $m_1 + m_2$ if $r_1 + r_2 < p/2$. The increase in noise in the addition of ciphertext is equivalent to $2\sum_{i=1}^{n} r_i$ with $n \geq 2$. As long as $\sum_{i=1}^{n} r_i < p/2$ the decryption of homomorphic addition of ciphertext $\sum_{i=1}^{n} c_i$ produce the sum of valid plaintext $\sum_{i=1}^{n} m_i$.

$c_1 \times c_2 = (q_1 p + 2r_1 + m_1) \times (q_2 p + 2r_2 + m_2) = q_1 q_2 p + 2r_2 q_1 p + q_1 m_2 p + 2r_1 q_2 p + 4r_1 r_2$ $+2r_1 m_2 + m_1 q_2 p + 2r_2 m_1 + m_1 m_2$. The evaluation of $c_1 \times c_2$ produce a decryption $m_1 m_2$ if $2r_1 r_2 + r_1 m_2 + r_2 m_1 < p/2$. The increase in noise in the homomorphic multiplication of ciphertexts $\prod_{i=1}^{n} 2^{i-1} r_i$ with $n \geq 2$. If the approximation of $p/2 = 2^t$ with a positive integer t, then $t – 1$ multiplications can be performed on ciphertexts before the decryption fails.

3.2 Noise management strategies in homomorphic encryption

Given two ciphertexts c_1 and c_2 which respectively contain 10 and 20 as noise value. The noise threshold in this encryption scheme for retrieving a valid plaintext after decryption is 300. Beyond this value, the decryption fails.

The homomorphic encryption of $(c_1 + c_2) \times c_2$ (1) gives a noise with value of 600. The resulted ciphertext c_3 from expression (1) contains a resulting noise greather than threshold value which is 300. The decryption of c_3 fails by providing a different value of result of evaluation of (1).

Indeed, the homomorphic encryption in expression (1) add a noise at level of 30. Said value is less than the threshold of 300. The decryption of $c_1 + c_2$ produce a plaintext which worth the sum of m_1 + m_2. Subsequently, this result is homomorphically multiplied by c_2. This operation brings a noise at level of 600 which is greater than the threshold. From this example, each typical operation introduces a specific type of increasing noise in ciphertext.

For a ciphertext with noise beyond the threshold, its decryption violates the correct property. Thus, to reduce noise in the ciphertext, several management strategies noise were proposed whose commonly used are defined below.

Growth of noise in homomorphic encryption. Noise in ciphertext can increase in several ways in homomorphic encryption. The growth of noise is the value which is added to the initial noise after one or several homomorphic operations between two or more ciphertexts. Thus, a growth of noise can be described as a: exponential noise, polynomial noise, linear noise, constant noise and logarithmic noise.

Exponential noise. [10,11]. Given two ciphertexts c_1 and c_2. Let be *op*, an arbitrary homomorphic operation on two or more plaintexts.

A noise is said exponential if there exists a value b'' in c_3 such that $b'' = 2^n b$ where b is the initial noise contain in c_1 and c_2 and $c_3 = op(c_1, c_2, n)$.

The growth in [10,11] is illustrated as follow: the homomorphic multiplication of two ciphertexts $mod - p$ messages chiffrés with noise at level ρ results a ciphertext with noise at level 2ρ; after second level of same operation the noise becomes $2^2\rho$, then $2^3\rho$ and so on; the noise exponentially increases with number of multiplications. The modulus is threshold value before decryption fails; therefore, if the bit size of p is $k.\rho$; the threshold of noise is reached after $\log_2 k$ multiplication levels.

Polynomial noise. Given two ciphertexts c_1 and c_2. Let be *op*, an arbitrary homomorphic operation on two or more plaintexts.

A noise is said to be polynomial if there exists a value b'' in c_3 such that $b'' = b^n$ where b is the initial noise contained in c_1 and c_2 and $c_3 = op(c_1, c_2, n)$. If n=2, the growth is said quadratic. If n=3, the growth is said cubic. The growth is illustrated in [8] as follows: Let a modulus q such that q is approximated by x^k, and given two ciphertexts $mod - p$ with a noise at level x. In Homomorphic multiplication, the noise becomes x^2. After four levels of homomorphic multiplication, the noise becomes x^{16}. And so on but at almost log k levels of multiplications, the noise reaches threshold value and the decryption fails.

Linear noise. Given two ciphertexts c_1 and c_2. Let be *op*, an arbitrary homomorphic operation on two or more plaintexts.

The noise is said linear if there exists a value b'' in c_3 such that $b'' = nb$ where b is the initial noise contained in c_1 and c_2 and $c_3 = op(c_1, c_2, n)$. After k multiplications, this growth is proportional to the noise whose magnitude is defined by $(N+1)^k r$.

In [4] Given two ciphertexts C_1 and C_2 are matrix-type B-bounded ciphertexts, in the sense that and the coefficients of μ_i and C_i and $\overrightarrow{e_i}$ all have an order of magnitude at most B. Then, C^+ represents the addition of two encrypted matrices whose noise in the elements is $2B$-borné, and $C^\times$ represents the multiplication of two encrypted matrices whose the noise in the elements is $(N+1)B^2$. In short, the level of error increases more than B^{2^L}, twice exponentially with multiplicative depth of circuit.

To compensate for this growth, an implementation of NAND gate on two ciphertexts C_1 *and* C_2 whose the noise B – strongly bounded, makes it possible to obtain new ciphertexts $C_3 \leftarrow I_N - C_1C_2$ (where I_N is identity matrix), then thesaid messages remains in $\{0,1\}$, and the coefficients of error vector C_3 have a magnitude at most $(N + 1)B$. The function $Flatten(C_3)$ makes it possible to obtain C_3 having the elements 0/1 in strongly bounded. This property ensures that circuits with L depth can be processed as long as the magnitude of error is at most $(N + 1)^L B$ and the homomorphic NAND of [4] generates noise of linear order of magnitude.

Constant noise. In this strategy, a multiplication on ciphertexts increases the noise by adding a constant value. The growth of noise is independent of the number of multiplications on ciphertexts.

Given two ciphertexts c_1 and c_2. Let be *op*, an arbitrary homomorphic operation on two or more plaintexts.

Noise is said to be constant if there exists a value b'' in c_3 such $b'' = b$ where b is the initial noise contained in c_1 and c_2 and $c_3 = op(c_1, c_2, n)$.

In [1], a ciphertext is defined as follows $AAE_s(m_i, q/16)$ where m_i is the plaintext in $\{0,1\}$, s is the private key and $q/16$ is magnitude of noise. When the homomorphic NAND is performed on ciphertexts c_1 and c_2 defined as follows $AAE_s(m_1, q/16)$ and $AAE_s(m_2, q/16)$ outputs ciphertext $c_3 = AAE_s(m_3, q/4)$.

Logarithmic noise. Given two ciphertexts c_1 and c_2. Let be *op*, an arbitrary homomorphic operation on two or more plaintexts.

A noise is said logarithmic if there exists a value b'' in c_3 such that $b'' = \log(b)$ where b is the initial noise contained in c_1 and c_2 and $c_3 = op(c_1, c_2, n)$.

Strategy management of noise. A noise strategy is a technique which manages the noise generated by n homomorphic operations on $n + 1$ ciphertexts to preserve the validity of the resulting ciphertexts after decryption and the homomorphic structure on *encrypt* algorithm.

A noise strategy is said to be compact if it allows the ciphertext to remain compact within the limit of the security parameter during homomorphic operations. It is clear that the expansion parameter is close to 1.

A noise strategy is said to be efficient if the execution time of the homomorphic operations on a set of ciphertext is reduced.

Two main noise management strategies have caught our attention, which are bootstrapping and modulus switching.

The bootstrapping. The majority of current schemes have a common property which adds short-sized noise to the plaintext message in the encryption operation. The homomorphic evaluation on ciphertexts increases noise to a threshold value where the decryption fails. [10] The bootstrapping proposed and conceptualized on the term *recrypt* by Gentry, is used to reduce noise in the encrypted message to a level which is determined by the complexity of the decryption circuit.

(1) Definition of function of reencryption: *Recrypt*.

Let c_1 ciphertext of m bit under the key pk_1 and $\overline{sk_1}$ the encrypted private key under the pk_2 using the function *encrypt* as follows: *encrypt*(pk_2, sk_{1j}) on bits of sk_1. It is defined as follows:

Recrypt takes as inputs pk_2, the circuit of decryption *decrypt*, the encrypted private key $\overline{sk}_1$ and the ciphertext c_1. At first glance, it generates a vector $\overline{c}_1$ using *encrypt*(pk_2, c_{1j}) which operates on bits of c_1, it encrypts at bit level the elements of c_1 under pk_2. It outputs $Evaluate(pk_2, decrypt, \overline{sk}_1, \overline{c}_1)$ that effectively eliminates the portion pk_1 of c_1 and keep it under pk_2.

The bootstrapping refreshes ciphertext by performing a homomorphic decryption. usually, a plaintext is encrypted two times as follows: ($\overline{c} = encrypt_{pk_2}(encrypt_{pk_1}(m))$) is decrypted from the outside to inside. But in *recrypt*, the inner layer of ciphertext is homomorphically decrypted by correspondent key $\overline{c}_1 = encrypt_{pk_2}(decrypt_{\overline{sk}_1, pk_2}(c_1))$. As long as this technique reduce the noise in the scheme by eliminating noise in inner layer of ciphertext, despite the noise being added by *evaluate*. As long as the added noise by *evaluate* is less than the removed noise from inner encryption layer, the decryption continues to produce valid results.

Bootstrapping is extremely costly in time and resources. The complexity of several approaches to bootstrapping is at least as complex as the number of times of decryption of each ciphertext which encrypts the private key [10]. This requires in the bootstrapping the homomorphic evaluation of the decryption, so each bit of the private key is replaced by a larger encrypted representation.

Modulus switching. Since bootstrapping is a greedy process, other alternatives have been proposed. The most promising alternative is switching modules [8]. The modulus switching uses an evaluator which knows the size of the private key not the key itself. It transforms a ciphertext *c mod q* to another *c′ mod p* without compromise the propriety of correct of scheme. The transformation simply divide *c* by a factor p/q. If *sk* is short and $p \ll q$ the noise in ciphertext decreases. This technique allows the evaluator to minimize noise without the private key and bootstrapping makes it a flexible mode of noise management.

Definition 3. Modulus switching. Let p and q two odd moduli and a vector of integers c. Let define an integer c' close to $(p/q)c$ such that $c' = c \, mod \, 2$. Then, for all s with $\left| \left[\langle c, s \rangle_q \right] \right| < q/2 - (q/p).l_1(s) : \left[\langle c', s \rangle \right]_p = \left[\langle c, s \rangle \right]_q mod \, 2 \; and \left| \left[\langle c', s \rangle \right]_p \right| < (p/q).$ $\left| \left[\langle c, s \rangle \right]_q \right| + l_1(s)$. or $l_1(s)$ est $l_1 - norm$ de s and $\langle c, s \rangle$ represents the cross product of two vectors, $[.]_q$ represents the modulo q elements.

The modulus switching reduces a ciphertext $c \in \mathbb{Z}_q$ by a factor β after each homomorphic multiplication. To do this, it painstakingly chooses a decreasing modulus q for each multiplication level that keeps the noise at small level and constant from low level to high level. The refreshed ciphertext is $c/\beta \in \mathbb{Z}_{q/\beta}$ in which the noise is reduced to level e/β. By performing this procedure, the absolute size of noise in the ciphertext decreases. Through this technique, K multiplications levels can be performed before reaching the noise threshold.

4 Noise management in homomorphic DGHV [11]

4.1 Somewhat homomorphic encryption scheme of DGHV

Given a safety parameter λ, the following assumptions are made: $\rho = \lambda$, $\rho' = \lambda$, $\eta = \lambda^2, \gamma = \lambda^5$ *and* $\tau = \gamma + \lambda$. These give a scheme having complexity of λ^5.

(1) *keygen*(λ).

- Choose at random an integer p of η bits as private key. Generate the public key as follows: $x_i = pq_i + r_i$ Where $q_i \in \mathbb{Z} \cap \left[0, 2^{\gamma/p}\right)$ *and* $r_i \in \mathbb{Z} \cap (-2^{\rho}, 2^{\rho})$ are randomly selected for $i = 1$ *to* τ.
- Rename x_0 as it is largest. Repeat if x_0 is odd and x_0 *mod p* is even.
- Publish the key $pk = \langle x_0, x_1, \ldots\ldots\ldots\ldots\ldots, x_\tau \rangle$.

(2) $Encrypt(pk, m)$

Given $m \in \{0,1\}$ and the public key *pk*, choose a random subset $S \subseteq \{1, 2, \ldots\ldots\ldots, \tau\}$ and a random integer $r \in (2^{-\rho'} 2^{\rho'})$. It outputs $c = m + 2r + 2\sum_{i \in S} x_i (mod\ x_0)$.

(3) *Decrypt*(*c,sk*) evaluates *m* as follows (*c mod p*)(*mod* 2)

4.2 Noise management in DGHV (reencryption and bootstrapping)

Let three new parameters be a function of λ noted κ, θ *and* Θ where $\kappa = \gamma\eta/\rho'$, $\theta = \lambda$ *and* $\Theta = \omega(\kappa \log \lambda)$. Given a private key $s_k = p$ and a public key p_k, add to the public key a vector of real y in [0, 2) in size Θ and κ precision such that there exists a subset $s \subset \{1, 2, \ldots\ldots\ldots, \Theta\}$ *with* $\sum y_i = 1/p$ *mod* 2 (the inverse of private key is replaced by sum of subset *s*).

Key generation. Let two keys from somewhat encryption scheme of s_k^* and p_k^*, the generation of keys to make this scheme fully homomorphic encryption follows the algorithm defined below:

- Set $x_p = \left[2^\kappa / p\right]$;
- Choose at random a vector of Θ bits $(S_1, S_2, \ldots\ldots\ldots\ldots, S_\Theta)$ with a hamming weight θ and set $s = \{i:\ s_1 = 1\}$;
- Choose at random integers $u_i \in \mathbb{Z} \cap \left[0, 2^{\kappa+1}\right)$ *and* $\sum_{i \in S} u_i = x_p\ mod\ 2^{\kappa+1}$ *avec* $i = 1, 2, \ldots\ldots\ldots\ldots\ldots, \Theta$;
- compute $y_i = u_i / 2^\kappa$ *and* $y = \{y_1, y_2, \ldots\ldots\ldots\ldots\ldots, y_\Theta\}$;
- keep secret the private key $s_k = s_k^*$ and publish the public key $\{p_k = p_k^*, y\}$

Reencryption: $reencrypt(p_k, c^*)$

for $i = 1, 2, \ldots\ldots\ldots, \Theta$, compute $z_i = c^* y_i\, mod\, 2$; publish the refreshed ciphertext z_i *and* c^*.

Decryption: $decrypt(s_k, c^*, z_i)$

The plaintext is computed as follows: $m = \left(c^* - \left[\sum_{i \in s} z_i\right]\right) mod\ 2$

4.3 Toy example of noise management in DGHV [12]

The DGHV undergoes an increase in noise exponential for the multiplication and constant for the addition or linear for the addition. The example given below shows how this algorithm can be applied to noise management for a one-bit encrypted message. With toy parameters, this example shows the difficulty of implementing it in everyday applications. The example below shows how to manage noise in [12] for unbounding processing on encrypted data.

Let $sk = p = 10001$. The public key p_k is built as follows:

Given $[q_0, q_1, q_2, q_3] = [36, 27, 34, 6]$ *and* $[r_0, r_1, r_2, r_3] = [8, 5, 4, 2]$. The public key $pk = [360044, 270032, 340038, 60008]$ where $x_i = q_i p + r_i$ and x_0 is the greatest.

The plaintext m = 0 is encrypted as follows with subset S = [1,3] with r = 31: $c = 0 + 2.31 + 2(270032 + 60008) = 660142\, mod\, 360044 = 300098$.

Refreshment of ciphertext $c_0 = 300098$ of plaintext $m_0 = 0$: reencryption of c_0 :

1) Let $s_k = p = 10001$ and $\kappa = 24$, compute $x_p = [2^{24}/10001] = 1678$;
2) Choose at random of vector of Θ bits with Θ = 9 with hamming weight of 3, $s = \{0, 0, 1, 0, 0, 1, 0, 1, 0\}$ and set $s = \{3, 6, 8\}$;
3) Choose at random integers $u_i \in \mathbb{Z} \cap [0, 2^{25})$ for i = 1 to 9 as follows: $u_i = \{281782, 1892147, 589103, 487403, 491831, 1093482, 293813, 3187325, 5718711\}$ where $\sum_{i \in s} u_i = u_3 + u_6 + u_8 = 589103 + 1093482 + 31873525 = 1678$
4) Set $y_i = u_i / 2^{25}$ with i = 1 to 9 as follows: $y_i = \{0.0167955, 0.1127807, 0.035133, 0.0290515, 0.0293154, 0.0651766, 0.0175126, 1.8998101, 0.3408617\}$ where $\sum_{i \in s} y_i = y_3 + y_6 + y_8 = 0.0351133 + 0.0651766 + 1.8998101 = 2.0001 \approx 1/p\, mod\, 2$
5) Compute $z_i = c_0 y_i\, mod\, 2$ avec $i = 1$ to 9 as follows: $z_i = \{0.295959, 1.2625086, 1.4311034, 1.4962348, 0.297047, 1.4929092, 1.3673068, 1.4962348, 1.2113898, 1.9144466\}$ with $\sum z_3 + z_6 + z_8 = 1.4311034 + 1.3673068 + 1.2113898 = 4.0098$

$$c_0 - \left[\sum_{i \in s} z_i \right] = 300098 - 4.0098 = 300094$$

6) Reencrypt c_0 as follows: $c_0 - \left[\sum_{i \in s} z_i \right] = 300098 - \sum z_3 + z_6 + z_8$

300094 is refreshed ciphertext of ciphertext 300098 whose the noise is 4 units. The decryption of 300094 outputs the same plaintext as 300098 which is 0: 300098 $mod\, 2$ = 300094 $mod\, 2$ = 0 $mod\, 2$.

5 Trend of noise management strategy from 2009 to 2016

5.1 Noise management strategy in homomorphic encryption from 2009 to 2016

In [BGV12] [7]. This algorithm uses the modulus switching as described below:

1) Choose a gradual decrease in modulus ($q_i = q/x^i$) for $i < k$;
2) Perform a multiplication of two $mod - p$ ciphertexts;
3) Switch the ciphertext to small modulus $q_1 = q/x$. The noise level of refreshed ciphertext decreases from x^2 *to* x.
4) Repeat step (2), the multiplication of two ciphertexts with noise at level x, the noise becomes again x^2;
5) Repeat step (3) by switching to modulus q_2 to reduce noise at x level. Each multiplication reduces the ratio $threshodnoise/levelnoise$ by x. This approch makes it possible to perform k levels of multiplication before reaching the noise threshold.

In [DM15] [1]. This algorithm uses bootstrapping and modulus switching to manage noise as described below:

1) Assume that the inputs bits $m_0, m_1 \in \{0,1\}$ are encrypted as ciphertexts $c_i \in AAE_s^4(m_i, q/16)$ with modulus $t = 4$ and error bound $q/16$;
2) Perform a homomorphic NAND as follows: $AAE_s^4(m_0, q/16) \times AAE_s^4(m_i, q/16) \rightarrow AAE_s^2(m_3, q/4)$ where $m_3 = 1 - m_0 m_1$;
3) Refresh $AAE_s^2(m_3, q/4) \rightarrow AAE_s^4(m_3, q/16)$ by using homomorphic accumulator as follows: $\left[2(b - a.E(s))\right] mod\, 2 = E(m_3)$ where $E(m_3)$ is the encrytion of the same plaintext under different encryption scheme E. It is output of homomorphic evaluation of decryption circuit under encrypted private key s.

5.2 Trend of noise management strategy from 2009 to 2016

Table 1. Summary of trend of noise management strategy

No	Scheme	Efficient	Noise Management	Noise Growth	Observations
1	[Gen09]	NO	Bootstrapping	Exponential	Ressources hungry and high expansion.
2	[DGHV10]	NO	Bootstrapping	Exponential	Ressources intensive and high expansion.
3	[BGV12]	YES	Modulus switching Bootstrapping	Polynomial	Optimal implmentation in Helib. [3,5] [17]
4	[GSW13]	NON	Leveled encryption bootstrapping	Linear	High expansion
5	[DM15]	YES	Bootstrapping/modulus swithing	Constant	Compute homomorphic NAND in less than a second. High expansion.

The Table 1 gives the summary of the noise direction over the period from 2009 to 2016. In this table, the row 5 shows the improvement of bootstrapping in terms of efficiency through the scheme [1] when performs it in less than a second.

In contrast, the column observations of Table 1 shows that expansion is still high despite the numerous techniques for reducing it.

6 Conclusion

The refreshing of the noise in the message is the challenge of homomorphic cryptography. Its reduction makes it possible to perform several operations on an encrypted message before each homomorphic operation. This has been shown through the DGHV scheme for one bit on an experimental basis and difficult to use in practice.

Noise is the bottleneck in homomorphic operations. Its rapid growth especially in multiplication reduces the possibility of running larger circuits on encrypted messages. The fastest algorithms today benefit from a constant increase in noise. The efficiency of homomorphic cryptography is answered in the management of noise through a logarithmic increase.

7 References

[1] L. Ducas and D. Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. Cryptology ePrint Archive, Report 2014/816. https://eprint.iacr.org/2014/816.2014

[2] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds, pages 3–33. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. https://doi.org/10.1007/978-3-662-53887-6_1

[3] A. Khedr, Member, IEEE, G. Gulak, Senior Member, IEEE, and V. Vaikuntanathan. SHIELD: Scalable homomorphic implementation of encrypted data-classifiers. Cryptology ePrint Archive, Report 2014/838. https://eprint.iacr.org/2014/838. 2014.

[4] C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Crypto '13, 2013. https://doi.org/10.1007/978-3-642-40041-4_5

[5] A. Khedr, G. Gulak, and V. Vaikuntanathan. SHIELD: Scalable homomorphic implementation of encrypted data-classifiers. IEEE Transactions on Computers, PP(99):1– 1, 2015.

[6] Y. Doröz and B. Sunar. Flattening NTRU for evaluation key free homomorphic encryption. Cryptology ePrint Archive, Report 2016/315, 2016.

[7] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In ITCS, pages 309–325, 2012. https://doi.org/10.1145/2090236.2090262

[8] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive, 2012:144, 2012.

[9] Joppe W. Bos, K. Lauter, J. Loftus, and M. Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In Stam [Sta13], pages 45–64.

[10] C. Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, STOC, pages 169–178. ACM, 2009. https://doi.org/10.1145/1536414.1536440

[11] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In Eurocrypt, pages 24–43, 2010. https://doi.org/10.1007/978-3-642-13190-5_2

[12] Homomorphic application and applications, Springer, 2014, New York.

[13] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In Foundations of Secure Computation, pages 169–180, 1978.

[14] S. Goldwasser and S. Micali. Probabilistic encryption. J. Comput. Syst. Sci., 28(2):270–299, 1984. https://doi.org/10.1016/0022-0000(84)90070-9

[15] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. Eurocrypt ’99, pp. 223–238. https://doi.org/10.1007/3-540-48910-X_16

[16] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. Crypto ’84, pp. 469–472. https://doi.org/10.1109/TIT.1985.1057074

[17] S. Halevi and V. Shoup, Algorithms in Helib, 2014. https://doi.org/10.1007/978-3-662-44371-2_31

[18] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. TCC ’05, LNCS 3378, pp. 325–341. https://doi.org/10.1007/978-3-540-30576-7_18

[19] H.-C. Hsu, Z.-Y. Liu, R. Tso, and K. Chen. Multi-value private information retrieval using homomorphic encryption, IJES, August 2020, 15th AsiaJCIS

[20] R. Motawie, Mahmoud, M. El-Khouly, and Samir A. El-Seoud. Security problems in cloud computing, IJES, December 2016. https://doi.org/10.3991/ijes.v4i4.6538

[21] F. O. Idepefo, O. S. Aderibigbe, B. S. Afolabi, and B. I. Akhigbe. Towards on architecture-based ensemble methods for online social network sensitive data privacy protection, IJES, Vol. 9, No. 1, 2021. https://doi.org/10.3991/ijes.v9i1.20819

8 Authors

Paulin Boale Bomolo is senior lecturer and PhD Student at university of Kinshasa in Mathematics and Computers sciences department. His field of research is cryptography, in particular homomorphic cryptography. He works to improve algorithms in everyday applications. He contributed to the publication of articles respectively in the journal IJCSI and IJSR such as “Study of Master-Slave Database replication in distributed database”, IJCSI, 2011.

Eugene Mbuyi Mukendi is professor and head of Mathematic and computers sciences department of the University of Kinshasa. As publications, Author of many publications, such as: “Enhanced Parallel Skyline on multi-core architecture with lax Memory space Cost”, IJCSI, volume 13, Issue 5, September 2016, Data mart approach for stock management model with a calendar under budgetary constraint, IJCSI, volume 15, Issue 5, September 2018, Poster and the 2nd International conference on Big Data Analysis and Data Mining, San Antonio, USA, December 1, 2015; Data Mart Approach for Stock Management Model with a calendar Under Budgetary constraint, IJCSI, volume 15, Issue 5, September 5, 2016. E-mail: eugene.mbuyi@unikin.ac.cd

Simon Ntumba Badibanga is professor at the Mathematic and Computers Sciences department of the University of Kinshasa. Director of informatics laboratory of the faculty of sciences at the university of Kinshasa. He is author of many articles in many scientific journals like in IJCSI, Poster and the 2nd International conference on Big Data Analysis and Data Mining, San Antonio, USA, December 1, 2015. E-mail: simon.ntumba@unikin.ac.cd

Article submitted 2021-08-04. Resubmitted 2021-10-11. Final acceptance 2021-10-14. Final version published as submitted by the authors.