International Journal of Security and Its Applications
Vol.8, No.2 (2014), pp.221-230
http://dx.doi.org/10.14257/ijsia.2014.8.2.23

Review of How to Construct a Fully Homomorphic Encryption Scheme

Chen Zhi-gang[1,2,3], Wang Jian[1], Chen Liqun[4] and Song Xin-xia[2]

[1]*Nanjing University of Aeronautics and Astronautics, China*
[2]*Zhejiang Wanli University, China*
[3]*Royal Holloway, University of London,UK*
[4]*HP Labs, UK*

Abstract

Fully homomorphic encryption (FHE) allows one to perform arbitrary computation on encrypted data. From the cloud computing perspective, a FHE scheme can be used to outsource a computation to a remote cloud server without revealing the computed data to the server. This is a very attractive cryptographic primitive. Existing methods used to construct FHE schemes are complicated, and often these methods are difficult to understand except by experts in the field. This paper aims to explain how to construct a FHE scheme in an easily understandable way. We first present a general overview of FHE scheme construction. We then take the van Dijk et al FHE scheme as an example and explain how a FHE scheme is obtained from a simple integer-based encryption scheme. Our explanation gives details of the noise bound, how noise growth can be controlled and how the vanDijk scheme is implemented.

Keywords: *Fully homomorphic encryption, Noise analysis*

1. Introduction

Encryption has traditionally been viewed as a mechanism that enables secure communication [1]. Someone who has the secret decryption key can learn the entire message, but without the key, the ciphertext does not provide any useful information about the message. Can we do arbitrary computations on data while it remains encrypted, without ever decrypting it? This question was posed by Rivest *et al.*, and has since become an open problem in cryptography [2]. This problem is known as fully homomorphic encryption (FHE).

Fully homomorphic encryption enables computation of arbitrary functions on encrypted data. Initially researchers developed a number of schemes that can compute addition or multiplication, on encrypted data [3-12], but not both! Other schemes were then developed, that enabled both additions and multiplications [13-17], but the numbers of these operations were limited. Breakthrough work by Gentry described the first construction of a FHE scheme in 2009 [18]. This scheme supports both addition and multiplication operations without any restriction on the numbers of operations.

Although not efficient Gentry's scheme did demonstrate that FHE could be achieved. Since then, researchers have constructed various other FHE schemes [19-28] and have significantly improved efficiency compared to Gentry's initial work [29].

All the existing methods used to construct a FHE scheme are complicated. They are not easy to understand by anybody except experts in the field. This paper aims to explain the idea of how to construct a FHE scheme in a simple way. We first, in Section 2, present a general overview of FHE scheme construction. We then, in Section 3, take the van Dijk *et al.*, FHE scheme as an example to explain how to obtain a FHE scheme from a simple integer-based encryption scheme. After that, in Section 4, we discuss some details of the noise bound and how to control noise growth. Finally we give a brief description of how

ISSN: 1738-9976 IJSIA

the scheme can be implemented.

2. A General Overview of a FHE Scheme

In a FHE scheme, there are no limitations on the manipulations that can be performed. These manipulations can be represented as circuits (combinations of AND gates and XOR gates) whose inputs are the ciphertexts $\psi_1,\ldots,\psi_t$. In addition to the three conventional algorithms required in any public key encryption scheme (KeyGeneration, Encryption and Decryption), a homomorphic encryption scheme requires a (possibly randomized) algorithm Evaluate. The Evaluate algorithm takes as input the public key *pk*, a circuit *C*, and a tuple of ciphertexts $\Psi = <\psi_1,\ldots,\psi_t>$ which are used as the inputs to *C*. The Evaluate algorithm outputs a ciphertext ψ such that ψ is the ciphertext of $C(m_1,\ldots,m_t)$ under *pk*, where m_i is the plaintext corresponding to the ciphertext of ψ_i.

One may consider that the requirement that a FHE scheme can evaluate arbitrary circuits is too demanding. An important relaxation of FHE is leveled fully homomorphic encryption, which can evaluate all circuits up to a fixed depth.

2.1. How to Construct a FHE Scheme

The first step of Gentry's scheme is to describe a Somewhat Homomorphic Encryption (SHE) scheme that supports a limited number of additions and multiplications on ciphertexts [18]. This is because every ciphertext has a noise component and any homomorphic operation increases this noise. Once the noise reaches a certain bound the resulting ciphertext does not decrypt correctly anymore. For example, if the bit-size of noise is ρ and the noise threshold is k ρ, then the bound is reached after only $log_2 k$ levels of multiplication.

If one wants to achieve FHE, this noise growth problem must be solved. Gentry's solution is to decrypt the ciphertext, but homomorphically! The input to the decryption circuit is the bits of the secret key and the bits of the ciphertext, each encrypted under another public key. As long as the decryption circuit can be handled by the SHE scheme, the output is another ciphertext for the same plaintext. If the degree of the decryption polynomial is small enough, the new ciphertext's noise will be less than that of the old ciphertext; this is called the "ciphertext refresh" procedure. One can refresh ciphertexts before every homomorphic operation. The decryption circuit is augmented by some gates, *e.g.*, AND or XOR; call this the augmented decryption circuit. If the SHE scheme can handle augmented decryption circuits, it can do arbitrary computations on ciphertexts. Since an AND gate and a XOR gate have the property of functional completeness [30]. That is, any other logic function can be implemented by combining AND gates and XOR gates.

If a SHE scheme can handle AND gates and XOR gates, the next critical question is whether the decryption circuit can be handled. If it can, this is called "bootstrapping"; unfortunately it cannot. Hence one needs to squash the decryption circuit so that the degree of the decryption polynomial is small enough, and then the SHE scheme can do arbitrary computation, and one gets a FHE scheme. This process is called ciphertext refresh.

The refresh algorithm is performed every time before homomorphic addition or multiplication of two ciphertexts. The algorithm can be seen as a re-encryption operation with a public and secret key pair, and the secret key itself is encrypted under the public key. Recall that the entirely homomorphic manipulation is represented as a circuit with multiple levels. As showed in Figure 1, at Level *i*, the refresh algorithm takes as input the encrypted secret key (s_i') and a pair of ciphertexts (c1 and c2) that are the output from the previous level, and outputs a pair of refreshed ciphertexts that are inputted to the gate in this level. The evaluation operation associated with the gate manipulates these two refreshed ciphertexts and creates a new ciphertext under pk_{i+1} that will be inputted to the next level. The number of public/secret keys depends on the depth of the circuit.

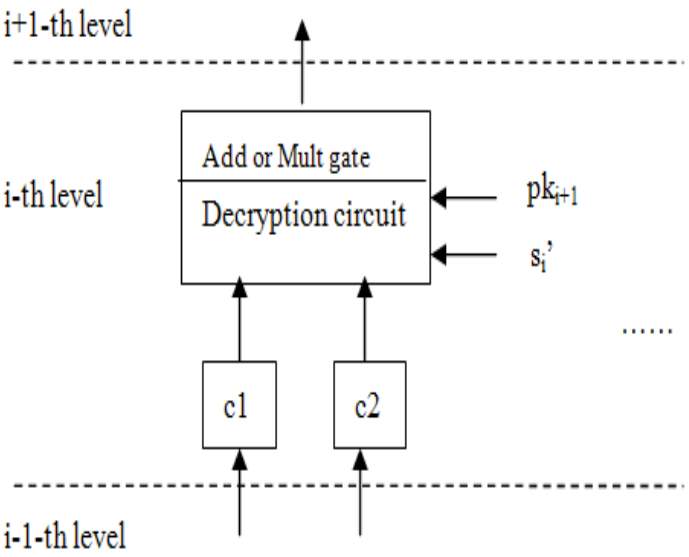

Figure 1. Ciphertext Refresh before Homomorphic Addition or Multiplication of Two Ciphertexts

2.2. A general Interpretation

We now give a general interpretation of how to construct a FHE scheme in term of a flow chart as illustrated in Figure 2. The interpretation consists of a module that was used in Gentry's scheme. It is interesting that every possible path in the flow diagram could be a possible way to construct a FHE scheme. The path presented as a solid line is the Gentry scheme. Dotted lines show alternative ways to construct a FHE scheme. There are some challenges as follows.

(1) In order to achieve full homomorphism, Gentry went through a so-called "squashing step" which relies on an additional very strong hardness assumption, namely, the hardness of the (average-case) sparse subset-sum problem. While the sparse subset-sum assumption is not a well-studied cryptographic assumption. This additional assumption was considered to be the main caveat of Gentry's solution and removing it has been, perhaps, the main open problem in the design of FHE schemes.
(2) In a "leveled" FHE scheme, to do ciphertext refresh, the number of the public keys is linear to the depth of the circuits that the scheme can evaluate. Note that if the encryption scheme is **circular-secure**, namely it is "safe" to reveal the encryption of a secret key under its own associated public key. Then a single public key can be used to all levels of the circuit. One can obtain a "pure" FHE scheme (with a constant-size public key) from a leveled FHE scheme. However, for all known SHE schemes, it is not know how to meet this requirement under any well-known cryptographic assumption.
(3) To obtain FHE, Gentry provided a remarkable bootstrapping theorem which states that given a SHE scheme that can evaluate its own decryption function (plus an additional operation), one can transform it into a "leveled" FHE scheme. Bootstrapping "refreshes" a ciphertext by running the decryption function on it homomorphically, using an encrypted secret key (given in the public key), resulting in a reduced noise. If there is a technique that can refresh ciphertext and meanwhile can reduce noise, one can obtain FHE without bootstrapping.
(4) Of course, the perfect way to construct a FHE scheme should be directly to obtain FHE without from a SHE scheme. But how to achieve this is still an open problem.

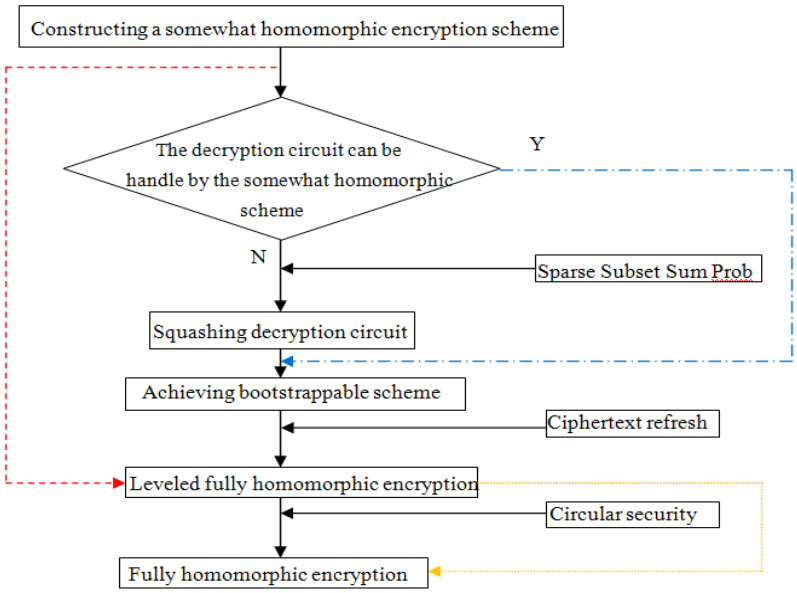

Figure 2. A General Interpretation of How to Construct a FHE Scheme

3. A FHE Scheme over the Integers

At Eurocrypt 2010 Van Dijk *et al.*, described the first FHE scheme over the integers (DGHV for short) [19]. We use the DGHV scheme as an example to show how a FHE scheme can be constructed. Just as in the Gentry scheme the starting point is a SHE scheme.

3.1. Somewhat Homomorphic Encryption Scheme

The scheme parameters. Let λ be the security parameter, γ, η and ρ the bit-lengths of the ciphertext x_i', the secret key p and the noise r_i respectively. Let τ be the number of x_i's in the public key, and ρ' *be* a secondary noise parameter used for encryption.
For a specific η-bit odd integer p, we use the following distribution over γ-bit integers:

$$\mathcal{D}_{\gamma,\ \rho}(p)=\{\text{Choose } q\leftarrow \mathbb{Z}\cap[0,2^{\gamma}/p),\ \ r\leftarrow \mathbb{Z}\cap(-2^{\rho},\ 2^{\rho})\text{: Output } x=q\cdot p+r\}.$$

KeyGen(1^{λ}): Generate a random odd integer p of size η bits. For $1\le i\le\tau$ sample $x_i\leftarrow \mathcal{D}_{\gamma,\ \rho}(p)$. Relabel that x_0 is the largest. Restart unless x_0 is odd and $[x_0]_p$ is even. Let $pk = (x_0, x_1 \ldots x_\tau)$ and $sk = p$.

Encrypt(*pk, m*): Choose a random subset $S\subseteq\{1,2,\ldots,\tau\}$and a random integer r in $(-2^{\rho'},\ 2^{\rho'})$, and output the ciphertext: $c = (m + 2r + 2\sum_{i\in S} x_i) \bmod x_0$.

Evaluate(*pk, C,* c_1, ……,c_t): Given the circuit C with t input bits, and t ciphertexts c_i, apply the addition and multiplication gates of C to the ciphertexts, by performing all the additions and multiplications over the integers, and return the resulting integer.

Decrypt(*sk,c*): Output $m\leftarrow(c \bmod p) \bmod 2$. Note that since $c \bmod p = c - p\cdot\lceil c/p\rfloor$, and p is odd, one can compute instead: $(c \bmod 2) \oplus (\lceil c/p\rfloor \bmod 2)$.

We will call $(c \bmod p)$ the noise associated to ciphertext c under key p. Decryption succeeds as long as the magnitude of the noise stays smaller than $p/2$. Homomorphic addition and multiplication increase the noise in the ciphertext. Given two ciphertexts with noise at most B addition results in a ciphertext with noise at most $2B$, while multiplication results in a noise as large as B^2.

3.2. Analysis of Circuit Depth

Clearly, the bottleneck is the multiplicative depth of the circuit, or the degree of the multivariate polynomial computed by the circuit. We analyze the maximum degree of the multivariate polynomial that is allowed to compute in the above scheme. Let $f(x_1,\ldots,x_t)$ be the multivariate polynomial computed by the circuit, and let d be its degree. Let x_i be the noise of a fresh ciphertext, and a fresh ciphertext has noise at most B. The bound $f(x_1,\ldots, x_t) < p/2$ would suffice for correct decryption. If $|\vec{f}|\cdot B^d < p/2$ (where $|\vec{f}|$ is the l_1 norm of the coefficient vector of f) then $C \in C_E$. In particular, E can handle f as long as $d < (\log p - \log |\vec{f}|)/\log B$. We consider settings where $\log |\vec{f}|$ is small in relation to $\log p$ and $\log B$, then we get

$$d < \log p / \log B. \qquad (1)$$

Equation (1) implies the homomorphic capacity of this scheme. Below we refer to the polynomials that satisfy Equation (1) as suitable polynomials and we denote by P_E the set of suitable polynomials and by $C(P_E)$ the set of associated circuits. The discussion above implies that $C(P_E) \in C_E$. The definition of the set C_E from above is rather indirect, while suitable polynomials are very easy to understand and compute. Note that E is only correct for circuits in C_E, but we want E to be correct for all circuits. Recall that if decryption circuit is in C_E, our purpose can be achieved. In other words, if the degree of a polynomial computed by decryption circuit is less than d from Equation (1), FHE will be achieved.

Next we analyze the depth of decryption circuit. Because computing $\lceil c \cdot p^{-1} \rfloor$ is complicated, E cannot handle the function $f(p, c) = \lceil c \cdot p^{-1} \rfloor$. We now explain the reasons as follows. The numbers c and p^{-1} can be express in binary bits, and p^{-1} must have $\log c > \log p$ bits of precision to ensure the rounding is correct. There are some facts we will use: Multiplying two t-bit numbers involves adding t numbers. The sum of two t-bit numbers is an up to t-degree polynomial of input bits. The 3-for-2 trick: adding 3 numbers can transfer to add 2 numbers with same sum, which output bits are up to degree-2 in input bits.

Hence, adding t numbers can finally get 2 numbers with same sum by 3-for-2 trick, which output bits are degree $2^{\log 3/2\ t} = t^{\log 3/2\ 2} = t^{1.71}$. Then the degree of adding final 2 numbers is $2t^{1.71}t = 2t^{2.71}$. Hence, f has a degree at least $2(\log p)^{2.71}$. Comparing this result to equation (1), it shows that in this scheme E cannot handle a polynomial f of such a high degree. The only solution is to squash the decryption circuit.

3.3. Analysis of New Decryption Circuit Depth

The idea of squashing the decryption circuit is similar to server-aided cryptography, where a user offloads some portion of a computationally intensive cryptographic task, such as decryption, onto an mistrusted server; in our case, the encrypter itself plays the server's role.

$\lceil c \cdot p^{-1} \rfloor$ is the product that is computed by the decryption circuit whose complexity is higher. The intuition is to transfer the product into a sum if we want to have a shallow decryption circuit. Since c is a ciphertext, it can't be changed. The only way is to transfer p^{-1} into a sum. However, p is a secret key that can't be revealed, we can only hide the value p in the sum. So we need a trapdoor for securing the secret key, where the trapdoor used is the **sparse subset sum problem** (namely, given a set of β numbers $\vec{y}$ and another number s, find the sparse (α-element) subset of $\vec{y}$ whose sum is s.). In this transformation, we add to the public key some extra information about the secret key, and use this extra information to "post process" the ciphertext. The post-processed ciphertext can be decrypted more efficiently than the original ciphertext, thus making the scheme bootstrappable. We pay for this saving by having a larger ciphertext, and also by introducing another hardness assumption (basically assuming that the extra information in

the public key does not help an attacker break the scheme). We finally obtain a new decryption circuit: (c *mod* 2) $\oplus$ ($\lceil \sum s_i \cdot z_i \rfloor$ *mod* 2). We refer the reader to the paper [19] for the formal description of this scheme.

Next we analyze the depth of the new decryption circuit obtained by squashing. There are only n bits of precision after the binary point for each z_i according to the setting from the DGHV scheme. It can be shown in Figure 3 how to compute the formula $\lceil \sum s_i \cdot z_i \rfloor$ *mod* 2:

If we can add t numbers in every column applying "3-for-2 trick", the decryption circuit still cannot be handled by this scheme. So we need to find another way to solve this problem. We observe a facts that the sum of every column is the Hamming weight of the column of bits ($a_{1,-n}$,…,$a_{t,-n}$). The following is a useful lemma for solving the Hamming weight.

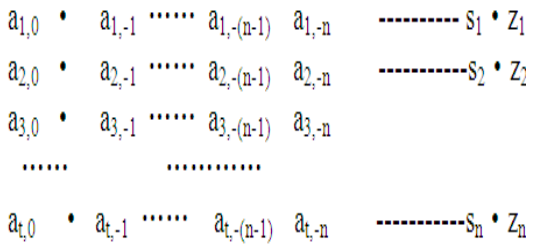

Figure 3. The Way of Computing the Formula $\lceil \sum s_i \cdot z_i \rfloor$ *mod* 2

Lemma 3.1 Let $\vec{\sigma}$ = <σ_1, σ_2,.., σ_t> be a binary vector, let $W = W(\vec{\sigma})$ be the Hamming weight of $\vec{\sigma}$, and denote the binary representation of W by W_n . . .W_1W_0. (That is, $W = \sum_{i=0}^{n} 2^i W_i$ and all the W_i's are bits.)

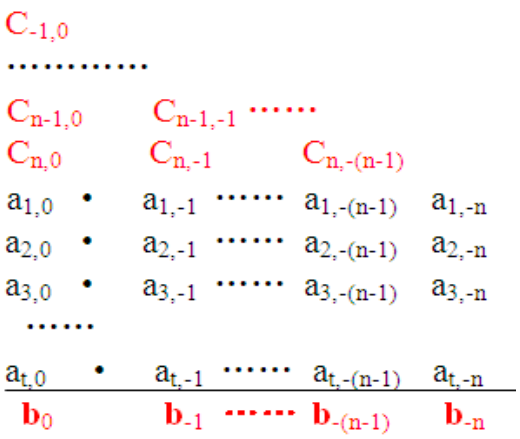

Figure 4. The Hamming Weight of Each Column and Carries from other Columns

For example, the Hamming weight of least column is W. The least bit of W is $e_{2^0}(a_{1,-n}$,…,$a_{t,-n})$ module 2 that is the sum of this column, where e_k (•) is the k-th elementary symmetric polynomial. We note the degree of e_k is exactly k. The i-th bit of W is laid in i-th column, it refer as $C_{n,i}$. Hence, as illustrated in Figure 4 we obtain b_{-n} by computing $e_{2^0}(a_{1,-n}$,…,$a_{t,-n})$ and $b_{-(n-1)}$ by computing $e_{2^0}(a_{1,-n}$,…,$a_{t,-n}, C_{n,-(n-1)})$, and so on.

So we finally get the result of $\lceil \sum s_i \cdot z_i \rfloor$ module 2 by computing (b_0 + b_{-1}) *mod* 2. This method enables the scheme to handle the decryption circuit. To verify this we consider the degree of multivariate polynomial $f(a_{i,j})$ computed by the circuit. At the beginning, the degree of the polynomial is one. After we compute the Hamming weight of each column, the degree will be changed. This procedure can be shown in Figure 5.

Since the final result is computed as (b_0 + b_{-1}) *mod* 2 that is related to both the first column and the second column, we need only to consider the degree of these two columns. The degree of computing the Hamming weight of the second column is 2^{n-1}, and the degree of computing the Hamming weight of the first column is 2^n. So the degree of multivariate polynomial $f(a_{i,j})$ computed by the circuit is 2^n. Recall that the maximum degree of f is less than $\log p / \log B$, and set $\|p\| \sim \lambda^2$, $\|r\| \sim \lambda$ according to parameters in the DGHV scheme. We get $\log p / \log B \sim \lambda$. If the scheme can handle decryption circuit, z_i should be only keep $\log \lambda$ bits of precision. After the decryption circuit is squashed and evaluated by using the elementary arithmetic technique， the scheme now can evaluate decryption circuit that results in a FHE scheme.

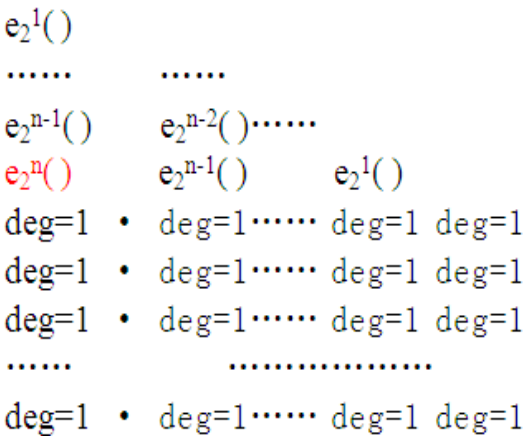

Figure 5. The Degree of each Column

3.4. Analysis of Noise in the Refresh Procedure

The refresh method is very important for constructing a FHE scheme. However, it also is bottleneck since noise of ciphertext grows explosively in the refresh procedure. We next analyze the noise in the refresh procedure. Let c be a ciphertext and （pk, $\vec{y}$）be public key. Let s_i' be an encryption of the i-th bit of the secret key s under the public key pk. The refresh procedure can be described as follows.

(1) For i=1,…,Θ, set $z_i \leftarrow (c \cdot y_i)$ *mod* 2, keeping only n bits of precision after binary points for each z_i. z_i can be express a matrix $(b_{i,j})$ with Θ row and n column, where $b_{i,j}$ is bit of z_i.

(2) In the case of the DGHV scheme, 0 and 1 are valid encryptions of themselves, so the ciphertext bits can be passed as is to the decryption circuit. Each entry $b_{i,j}$ in a matrix which does not need to be encrypted.

(3) For i=1,…, Θ, s_i' multiple each entry of the i-th row in matrix $(b_{i,j})$, then a new matrix A can be obtained. We note that each entry in the matrix A is an integer rather than a bit.

(4) The Hamming weight of each column in the new matrix A can be computed by the method showed in the section above. The least bit of Hamming weight of each column in matrix A is $e_{2^0}(\ldots)$, where the numbers in the parentheses are integers.

(5) We obtain b by computing (b_0 + b_{-1}) *mod* 2, this result is the value of this formula (「$\sum s_i z_i$ 」 *mod 2*).

(6) Let $\overline{c}$ be encryption of c modulo 2 under pk.

(7) Let c^* be the sum of $\overline{c}$ and b. Finally, then the refresh ciphertext of c is c^*.

Let B be bound of noise in fresh ciphertext， and the size of noise is ρ. We analyze the growth of noise in the steps of refresh procedure.

(1) There is no noise in this step.
(2) There is no noise in this step.
(3) The result of this step is that we obtain a $\Theta \times n$ matrix A containing clean ciphertexts with noise at most B, since s_i' has noise at most B.
(4) We only need to consider growth of noises when we compute the Hamming weight of both the 0-th column and 1-th column. The noises of these two columns derived from other column's carry. The entry with maximum degree in matrix A is $e_{2^{n-1}}(\ldots)$ that is come from the least column, so the noise of $e_{2^{n-1}}(\ldots)$ is at most $B^{2^{n-1}}$. The noise of $e_{2^{n-2}}(\ldots)$ in the 1-th column is also at most $B^{2^{n-1}}$, and so on. We can know similarly the noise of 0-th column is at most B^{2^n}.
(5) This step is the sum of two integers, so the noise of this sum is approximately B^{2^n} by computing $B^{2^n}+B^{2^{n-1}}$.
(6) Since $\overline{c}$ is a fresh ciphertext, the noise of this step is at most B.
(7) Finally, we obtain the noise of result ciphertext that is approximately B^{2^n}.

We can see that noise grows explosively in the refresh procedure from above analysis.

3.5. The Steps of Implementing an Augmented Decryption Circuit

As discussed before, arbitrary functions in a FHE scheme are expressed as a set of augmented decryption circuits. An augmented decryption circuit takes as input two ciphertexts *(*c_1, c_2) and a public key *PK=*（*pk*, $\vec{y}$） and a ciphertext of encrypting the secret key under *pk, say s'*. The algorithm of an augmented decryption circuit consists of the following steps:

Step 1: we encrypt c_1 mod 2 and c_2 mod 2 respectively under *pk* to obtain c_1' =Encrypt (*pk*, c_1 *mod* 2) and c_2'= Encrypt (*pk*, c_2 *mod* 2).

Step 2: In order to reduce noises of c_1' and c_2', we need to do refresh. The refresh procedure has been described in the previous section. The results of this procedure is called fresh ciphtertexts, and they are denoted by c_1* and c_2* respectively.

Step 3: We then input c_1* and c_2* to a gate in the circuit (Add or Mult gate). Note that all the operations are performed over the integers, and return the resulting integer.

Step 4: If this is the last circuit of the whole set, run refresh once again and output the result. Otherwise, carry on to the next level circuit.

Acknowledgements

The first author would like to thank for the Fund of Jiangsu Innovation Program for Graduate Education (No.CXLX12_0162), the Fundamental Research Funds for the Central Universities, and Ningbo Natural Science Foundation (No.2012A610067) and the Chinese National Scholarship fund, and also appreciate the benefit to this work from projects in science and technique of Ningbo municipal. The forth author would like to thank for Ningbo Natural Science Foundation (No.2013A610071). The third author thanks Chris Newton for the useful discussions.

References

[1] W. Diffie and M. Hellman, "New directions in cryptography", Information Theory, IEEE Transactions on, vol. 22, no. 6, pp. 644-54.
[2] R. L. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", Communications of the ACM, vol. 21, no. 2, **(1978)**, pp. 120-6.
[3] S. Goldwasser and S. Micali, "Probabilistic encryption", Journal of computer and system sciences, vol. 28, no. 2, **(1984)**, pp. 270-99.
[4] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", IEEE Trans Inf Theor, vol. 31, no. 4, **(1985)**, pp. 469-72.

[5] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes", Advances in Cryptology -EURPCRYPT 1999, Springer-Verlag, **(1999)**, pp. 223-38.
[6] I. Damgrd and M. Jurik, "A generalisation, a simplification and some applications of Paillier's probabilistic public-key system", Public Key Cryptography – PKC 2001. Springer-Verlag, **(2001)**, pp. 119-36.
[7] M. S. Ajtai and C. Dwork, "A public-key cryptosystem with worst-case/average-case equivalence", Proceedings of STOC'97, ACM, **(1997)**, pp. 284-93.
[8] O. Regev, "New lattice-based cryptographic constructions", J ACM, vol. 51, no. 6, **(2004)**, pp. 899-942.
[9] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography", Proceedings of STOC 2005, ACM, **(2005)**, pp. 84-93.
[10] J. D. Cohen and M. J. Fischer, "A robust and verifiable cryptographically secure election scheme", Proceedings of FOCS 1985, IEEE Computer Society, **(1985)**, pp. 372-82.
[11] D. Naccache and J. Stern, "A new public key cryptosystem based on higher residues", Proceedings of ACMCCS 1998, ACM, **(1998)**, pp. 59-66.
[12] T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring", Advances in Cryptology - EUROCRYPT'98, Springer Berlin / Heidelberg, **(1998)**, pp. 308-18.
[13] D. Boneh, E. J. Goh and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts", Theory of Cryptography, **(2005)**, pp. 325-41.
[14] C. Gentry, S. HalevI and V. Vaikuntanathan, "A simple BGN-type cryptosystem from LWE", Advances in Cryptology–EUROCRYPT 2010, **(2010)**, pp. 506-22.
[15] C. Gentry, S. HalevI and V. Vaikuntanathan, "A Simple BGN-Type Cryptosystem from LWE", Advances in Cryptology – EUROCRYPT 2010 [M]. Springer Berlin / Heidelberg, **(2010)**, pp. 506-22.
[16] C. A. Melchor, P. GaboriT and J. HerranZ, "Additively homomorphic encryption with d-operand multiplications", Advances in Cryptology – CRYPTO 2010. Springer-Verlag, **(2010)**, pp. 138-54.
[17] T. Sander, A. YounG and M. Yung, "Non–interactive cryptoComputing for NC^1", Foundations of Computer Science, 1999 40th Annual Symposium on, **(1999)**, pp. 554-6.
[18] C. Gentry, "Fully homomorphic encryption using ideal lattices", Proceedings of STOC 2009, ACM, **(2009)**, pp. 169-78.
[19] M. Van Dijk, C. Gentry, S. HalevI and V. Vaikuntanathan, "Fully Homomorphic Encryption over the Integers", Advances in Cryptology - EUROCRYPT 2010 [M], Springer Berlin / Heidelberg, **(2010)**, pp. 24-43.
[20] N. Smart and F. VercautereN, "Fully homomorphic encryption with relatively small key and ciphertext sizes", Public Key Cryptography – PKC 2010, Springer Berlin / Heidelberg, **(2010)**, pp. 420-43.
[21] D. Stehl and R. Steinfeld, "Faster fully homomorphic encryption", Advances in Cryptology – ASIACRYPT 2010, Springer Berlin / Heidelberg, **(2010)**, pp. 377-94.
[22] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages", Advances in Cryptology – CRYPTO 2011, Springer, **(2011)**, pp. 505-24.
[23] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE", Proceedings of FOCS 2011, IEEE Computer Society, **(2011)**, pp. 97-106.
[24] C. Gentry and S. Halevi, "Implementing Gentry's Fully-Homomorphic Encryption Scheme", Advances in Cryptology – EUROCRYPT 2011 [M]//PATERSON K. Springer Berlin / Heidelberg, **(2011)**, pp. 129-48.
[25] J.-S. Coron, A. Mandal, D. Naccache and M. Tibouchi, "Fully homomorphic encryption over the integers with shorter public keys", Advances in Cryptology – CRYPTO 2011, Springer, **(2011)**, pp. 487-504.
[26] J.-S. Coron, D. Naccache and M. Tibouchi, "Public key compression and modulus switching for fully homomorphic encryption over the integers", Advances in Cryptology – EUROCRYPT 2012, Springer, **(2012)**, pp. 446-64.
[27] A. BogdanoV and C. H. LEE, "Homomorphic encryption from codes", Cryptology ePrint Archive, Report 2011/622.
[28] Z. Brakerski, C. GentrY and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping", Proceedings of ITCS 2012. ACM, **(2012)**, pp. 309-25.
[29] V. Vaikuntanathan, "Computing Blindfolded: new developments in fully homomorphic encryption", Proceedings of FOCS 2011. IEEE Computer Society, **(2011)**, pp. 5-16.
[30] H. Enderton and H. B. Enderton, "A mathematical introduction to logic", Academic press, **(2001)**.

Authors

Zhigang Chen, is an associate professor at Zhejiang Wanli University. He received his BSc in Mathematics in 1994 and MSc in Computer Software and Theory in 2004. He is now working on PhD in the Nanjing University of Aeronautics and Astronautics. Currently his researches focus on fully homomorphic encryption and lattice-based cryptography. He is the corresponding author of this work and his email address is chzg99@21cn.com.

Jian Wang, is a full professor in the Nanjing University of Aeronautics and Astronautics. He received his BSc, MSc and PhD in 1989, 1992 and 1998 respectively. From 2001 to 2003, he did research on applied cryptography and information security in the University of Tokyo. His current research interests include broadcast encryption, security issues on wireless sensor network and ad-hoc, identification authentication, security metrics of software and system, etc. He has published more than 30 papers and has had research funding from Natural Science Foundation of China, from Natural Science Foundation of Jiangsu, and from the National Cryptography Development Foundation.

Liqun Chen, is a researcher at Hewlett-Packard Laboratories. She has developed several cryptographic schemes adopted by International Standards. She has an extensive publication record and holds over 50 granted patents in cryptography and information security. Liqun serves as an associate editor of IEEE Transactions on Information Forensics and Security and is an editorial board member of the International Journal of Information Security. Liqun obtained her BSc, MSc and PhD degrees in Engineering from Southeast University. Prior to joining HP, she worked at Southeast University, the University of Oxford and Royal Holloway, University of London.

Xinxia Song, is an associate professor at Zhejiang Wanli University. Her received her BSc in Mathematics in 1995 and MSc in Algebra Theory in 2004. Her researches currently focus on cryptography and algebra.

