2019 年 9 月 四川大学学报(自然科学版) Sep. 2019
第 56 卷 第 5 期 Journal of Sichuan University (Natural Science Edition) Vol. 56 No. 5

doi：10.3969/j.issn.0490-6756.2019.05.011

基于全同态加密与对称加密融合的批处理研究

陶梦龙，胡 斌
(解放军信息工程大学，郑州 450001)

摘 要：现有的全同态加密方案都具有很大的密文膨胀问题，该问题是制约实际应用的重要瓶颈. 为了提高传输效率，Naehrig 等提出了混合加密的想法，即用户使用密钥为 k 的对称算法 E 加密明文 m，再使用公钥为 pk 的全同态方案加密密钥 k，将缩小尺寸后的密文 $c'=(HE_{pk}(k),E_k(m))$ 发送给云端，云端可以同态运算解密电路 $C_{E^{-1}}$ 解压出同态密文 $HE_{pk}(m)$. 本文将全同态加密与对称加密融合方案推广到批处理形式，利用中国剩余定理将 l 个密文 $E_k(m_0),\cdots,E_k(m_{l-1})$ 打包进一个密文 C 中，将 $C'=(HE_{pk}(k),C)$ 发送给云端. 云端利用 C'，只需要同态运算 $C_{E^{-1}}$ 一次就可以恢复出全部的 $HE_{pk}(m_i)$，这个过程在原方案中需要进行 l 次. 通过这种方式，极大地缩短了原本需要耗费大量计算的同态运算解密电路过程. 文中以批处理 GSW13 全同态加密与 FLIP 流密码融合方案为例详细说明了这一过程. 与原方案相比，对于安全参数为 λ 的 FLIP 流密码方案，批处理方案可以将这个过程的计算复杂性从 $\widetilde{O}(\lambda^3)$ 缩小到 $\widetilde{O}(\lambda^2)$.

关键词：全同态加密；对称加密；批处理加密；中国剩余定理；同态运算
中图分类号：TP309.7 **文献标识码**：A **文章编号**：0490-6756(2019)05-0857-10

Research based on batch fully homomorphic encryption-symmetric encryption

TAO Meng-Long, *HU Bin*
(PLA Information Engineering University, Zhengzhou 450001, China)

Abstract: All homomorphic encryption schemes proposed so far suffer from a very large ciphertext expansion, which is a very significant bottleneck in practice. In order to improve the transmission efficiency, Naehrig *et al*. proposed an idea of hybrid encryption, i. e. a user encrypt some plaintext m with a symmetric encryption scheme E under some private key k, and encrypt the private key k with a homomorphic encryption scheme under some public key pk, transmit a much smaller cipertext $c'=(HE_{pk}(k), E_k(m))$ that cloud decompresses homomorphically into the $HE_{pk}(m)$ through a decryption circuit $C_{E^{-1}}$. In this paper, we extend the Fully Homomorphic Encryption-Symmetric Encryption framework into a batch one, i. e. we use the Chinese Remainder Theorem to pack l ciphertexts $E_k(m_0),\cdots,E_k(m_{l-1})$ into a single C, send $C'=(HE_{pk}(k),C)$ to the cloud. Given C', cloud only needs to homomorphically evaluate $C_{E^{-1}}$ for once to recover all $HE_{pk}(m_i)$, rather than l times in original scheme. By this way, we can greatly reduce the times of homomorphically evaluating decryption circuit, which costs a lot of computation. We also give out an instance of batch GSW13-FLIP scheme to explain in detail. Comparing to original scheme, our batch scheme can reduce the computational complexity from $\widetilde{O}(\lambda^3)$ to $\widetilde{O}(\lambda^2)$, where λ is

收稿日期：2018-11-05
基金项目：国家自然科学基金(61601515)；河南省自然科学基金(162300410332)
作者简介：陶梦龙(1994－)，男，湖北襄阳人，硕士生，研究方向为全同态密码. E-mail：taomenglong1@126.com
通讯作者：胡斌. E-mail：hb2110@126.com

security parameter of FLIP.

Keywords: Fully homomorphic encryption; Symmetric encryption; Bath encryption; Chinese remainder theorem; Homomorphic evaluate

1 Introduction

With the development of communication technology, many users with limited storage and computing facilities emerge, as well as cloud services with extensive storage and computing capabilities. Cloud service and secure multi-party computation (MPC) have become the current hot research. In this case, the outsourcing of data and the delegation of data processing are becoming more and more interesting. In this process, however, for privacy reasons, users do not want the cloud service to know their specific data, allowing only the ciphertext to be processed. The traditional encryption scheme does not allow the homomorphic computation of ciphertext. For this purpose, Gentry proposed the first fully homomorphic encryption (FHE) scheme[1] based on ideal lattices. This scheme proceeds in several steps. First, one constructs a somewhat homomorphic encryption scheme, which only supports a finite number of multiplications. Second, squashing the decryption circuit so that it can be presented as a low degree polynomial. Finally, the key step of Gentry, called bootstrapping, consists in homomorphically evaluating the squashed decryption circuit to refresh the ciphertext, resulting in a new ciphertext associated with the same plaintext, but with reduced noise. Since the breakthrough result of Gentry, many works have been published, introducing new hard problem, improving homomorphic efficiency. Dijk *et al.*[2] proposed fully homomorphic encryption scheme over integers based on approximate GCD problem. Brakerski and Vaikuntanathan[3-4] proposed the scheme based on the learning with errors (LWE) and ring learning with errors (RLWE) problems. Gentry *et al.*[5] introduced the scheme based on LWE using a new technique from approximate eigenvector problem. Full homomorphic encryption plays an important role in privacy protection[6-7]. However, the huge computation and storage cost of homomorphic encryption are still the main limitation for the deployment of cloud services based on such FHE frameworks.

In these cloud applications, we often send some data encrypted under a FHE scheme to the cloud to process in further. Typically, a user (Alice) encrypts some plaintext m under some other's public key pk (Bob) and sends some homomorphic ciphertext $c = HE_{pk}(m)$ to a third-party evaluator in the Cloud (Charlie). However, all homomorphic encryption schemes suffer from a very large ciphertext expansion, which greatly restricts the practical application. Naehrig *et al.* first considered the problem of reducing the size of c as efficiently as possible in Ref. [8]. We could encrypt m with a symmetric encryption scheme E under some secret key k, then send a much smaller compressed ciphertext $c' = (HE_{pk}(k), E_k(m))$ to the cloud. By utilizing the homomorphic property of FHE, cloud can homomorphically evaluate the decryption circuit $C_{E^{-1}}$ to recover the original $c = HE_{pk}(m) = C_{E^{-1}}(HE_{pk}(k), E_k(m))$, this can be assimilated to compressed encryption method and ciphertext decompression procedure. Obviously, for long messages, the expansion rate of ciphertext

$$|(HE_{pk}(k), E_k(m_1), E_k(m_2), \ldots)| / |(m_1, m_2, \ldots)|$$

is approaching 1. In this way, we greatly reduce the storage and transmission complexity for the long messages.

The framework of homomorphic encryption - symmetric encryption leaves an open question of how to choose symmetric encryption scheme E to minimize the decompression overhead, while preserving the same security level as originally intended.

Some work has been done to investigate the cost of a homomorphic evaluation of AES and several optimized implementations[9-11], including some lightweight block ciphers Simon[12] and Prince[13] *etc*. These lightweight block ciphers seem to be natural candidates for the framework, however, they may lead to worse result than AES. Specifically speaking, existing homomorphic encryption schemes use noisy encryption, where the noise of ciphertexts grows along with homomorphic operations. Normally, homomorphic multiplication increases the noise much faster than homomorphic addition. The noise of homomorphic evaluation depends mostly on the multiplicative depth of the circuit. However, many lightweight block ciphers have a large number of rounds for security, which is prohibitive in the FHE context. Albrecht *et al*.[14] observed that the standards of designing lightweight block ciphers does not applies to design a symmetric encryption with a low-cost homomorphic evaluation. Both the number of rounds and the number of binary multiplications required to evaluate an Sbox effect the complexity of homeomorphic evaluation of decryption circuit. Taking both criteria into account, Albrecht *et al*.[14] design a family of block ciphers called LowMC with very small multiplicative size and depth. Later, stream ciphers are also seemed to be candidates, Trivium and Kreyvium[15] are proposed for low multiplicative depth of decryption that is appropriate for homomorphic evaluation.

Unfortunately, these two types of schemes are still limited by complementary drawbacks. The noise produced by homomorphic evaluation of the decryption circuit of block ciphers is high and constant, while stream ciphers have a low initial noise that grows with each successive encryption. In consideration of combining the best of these two previous works, Méaux *et al*.[16] proposed a new stream cipher construction named filter permutator and a family of stream ciphers denoted as FLIP, based on the filter generator construction. It combines both advantages from the constant noise property offered by block ciphers, and the lower noise levels of stream ciphers. Overall, filter permutators in general and FLIP instances in particular open a large design space of new symmetric constructions with low-cost homomorphic evaluation.

In practical application, every time we transmit a compressedciphertext, we need to homomorphically evaluate the decryption circuit $C_{E^{-1}}$ once to recover the original ciphertext $HE_{pk}(m)$. When we transmit a long message, this procedure brings a considerable amount of computation. Inspired by previous works, packing technique can combine many operations into one operation, which greatly improves the efficiency of calculation. For this reason, we consider extending the homomorphic encryption-symmetric encryption framework into a batch one, *i. e.*, we use the Chinese remainder theorem to pack l ciphertexts $E_k(m_0),\cdots,E_k(m_{l-1})$ into a single $C=CRT(E_k(m_0),...,E_k(m_{l-1}))$[10], then we sent new compressed ciphertext $(HE_{pk}(k),C)$ to the cloud, any operation we do with C can be reflected on $E_k(m_i)$. At last, given $(HE_{pk}(k),C)$, utilizing the property of FHE, we get C' by homomorphically evaluate the decryption circuit $C_{E^{-1}}$ with $(HE_{pk}(k),C)$. Then we can simply recover each $HE_{pk}(m_i)$ by $(C' \bmod p_i) \bmod 2$. That is to say, by introducing a new calculation of packing, we can reduce the homomorphic evaluation from l times to once. We also give out an instance of GSW13-FLIP framework for more specific analysis. When we need to transmit l ciphertexts, in original scheme, it requires l times of homomorphic evaluation, the computational complexity of each homomorphic evaluation of decryption circuit is roughly $\widetilde{O}(\lambda)$, so the computational complexity of recovering all $HE_{pk}(m_i)$ is roughly $\widetilde{O}(\lambda^3)$. In batch scheme, we reduce the homomorphic evaluation of decryption circuit for only once. But meanwhile, we also introduce a new calculation $C=CRT(E_k(m_0),...,E_k(m_{l-1}))$, the computational complexity of batch scheme is roughly $\widetilde{O}(\lambda^2)$. In this instance, we reduce the computational complexity from $\widetilde{O}(\lambda^3)$ to $\widetilde{O}(\lambda^2)$. Actually, the more ciphers we pack, the more computational costs we save.

2 Background

In this section, we introduce the filterpermutator, a new stream cipher construction. We describe the general filter permutator structure and specify a family of filter permutators (donated as FLIP). Next, we introduce the batch DGHV scheme what we utilize to extend the fully homomorphic encryption-symmetric encryption framework into a batch one.

2.1 Description of the FLIPfamily of stream ciphers

In order to design a symmetric construction compatible for FHE applications, we consider both block and stream ciphers, each with advantages and disadvantages, as discussed in Refs. [17-18].

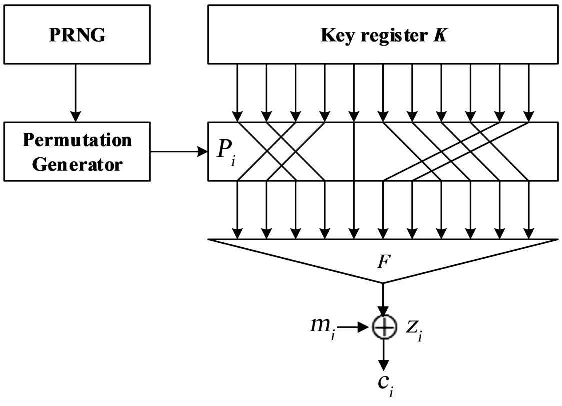

Fig. 1　The general structure of FLIP

Since the growth of noise is one of the most important factors restricting the application of FHE, our main purpose is to limit the growth of noise as much as possible. In other words, we need a symmetric construction whose circuit has a lower multiplicative depth. This ideal property is also considered to be a high homomorphic capacity, which is difficult to obtain with block ciphers because of the large algebraic degree of the output due to the round iterations. However, the good point is that noise is a constant for each block, which means that noise does not add any limitation to the number of generated ciphertext blocks. On the other hand, the homomorphic capacity of stream ciphers is usually very high for the first ciphertext bit, but as more bits are generated, this requires reinitializing the cipher or using techniques such as bootstapping.

Taking the best from both types of schemes, Méaux *et al.*[16] designed an innovative stream cipher, which has a very good homomorphic capacity and remains the noise of homomorphic evaluation constant with time. The new stream cipher is based on the filter generator construction, the general structure of filter permutators is represented in Fig. 1.

FLIP is composed of three main components:

(1) A register storing the N-bit K.

(2) A (bit) permutation generator controlled by a Pseudo Random Number Generator (PRNG), which is initialized with a public IV, producing an N-bit permutation P_i at each clock i.

(3) A filtering function which generates a key stream z_i.

Comparing to the filter generator construction, FLIP drops the register update part to avoid the algebraic degree increase. Instead, the register bits are permuted rather than updated by means of LFSR. In details, when the PRNG is initialized with an IV, at eachclock i, the permutation generator produces a permutation P_i, key bits are permutated by P_i before they enter the filter function F. XORing m_i and z_i generated by F, we produce the c_i.

The decryption process is similar to encryption. We generate the keystreams z_i, XOR them with c_i to recover the m_i. When homomorphically evaluating decryption circuit, the permutation part produces none noise, the main component that affects homomorphic efficiency is filter function. F is an N-variable Boolean function defined by the direct sum of three specific Boolean functions f_1, f_2 and f_3[16].

Definition 1 (Linear functions) Let $n>0$ be a positive integer, the L_n linear function is an n-variable Boolean function defined as:

$$L_n(x_0, \ldots, x_{n-1}) = \sum_{i=0}^{n-1} x_i$$

Definition 2 (Quadratic functions) Let $n>0$ be a positive integer, the Q_n quadratic function is a $2n$-variable Boolean function defined as:

$$Q_n(x_0, \ldots, x_{2n-1}) = \sum_{i=0}^{n-1} x_{2i} x_{2i+1}$$

Definition 3 (Triangular functions) Let $k>0$ be a positive integer, the k-th triangular function T_k is a $\frac{k(k+1)}{2}$ variable Boolean function defined as:

$$T_k(x_0, \ldots, x_{\frac{k(k+1)}{2}-1}) = \sum_i^k \prod_{j=0}^{i-1} x_{j+\sum_{l=0}^{i-1} l}$$

For example, the 4th triangular function T_k is

$$T_4 = x_0 \oplus x_1 x_2 \oplus x_3 x_4 x_5 \oplus x_6 x_7 x_8 x_9$$

These three types of function have good property of several security criteria, non-linearity, resiliency and algebraic immunity etc. Three parts of filter function respectivelyhave n_1, n_2, n_3 variables.

$$f_1(x_0, \ldots, x_{n_1-1}) = L_{n_1} = \sum_{i=0}^{n_1-1} x_i$$

$$f_2(x_{n_1}, \ldots, x_{n_1+n_2-1}) = Q_{n_2/2} = \sum_{i=0}^{n_2/2-1} x_{2i} x_{2i+1}$$

$f_3(x_{n_1+n_2}, \ldots, x_{n_1+n_2+n_3-1})$ is the direct sum of nb triangular function T_k, each T_k has different and independent variables, we donate f_3 as nbT_k.

We define F as the direct sum of f_1, f_2 and f_3, such that:

$$F(x_0, \ldots, x_{n_1+n_2+n_3-1}) = f_1 \oplus f_2 \oplus f_3 = L_{n_1} + Q_{n_2/2} + \sum_{i=1}^{nb} T_k,\ n_3 = nb \cdot \frac{k(k+1)}{2}$$

Since the filterpermutators is similar with a filter generator, Méaux *et al*. provide an initial analysis of FLIP filter permutators against a couple of the most common attacks on filter generators. Some instances of attack complexities and the selection of parameters aiming at 80- and 128-bit are presented in Tab. 1[14], where FLIP (n_1, n_2, nbT_k) donates the FLIP with n_1, n_2, n_3 variables for each part.

Tab. 1 Attack complexities and parameters of two concrete instances of FLIP

	Key size N	AA	l	FAA	l	HOC	l	Security λ
FLIP(46,136,4T_{15})	662	91	52	81	52	90	48	80
FLIP(86,238,5T_{23})	1704	149	105	137	105	128	74	128

AA, FAA, HOC stand for Algebraic Attacks, Fast Algebraic Attacks, Higher-Order Correlation attacks, respectively. l represents the number of bits to guess that leads to the optimal complexity of guess and determine attacks. Finally, λ stands for the security parameter of F, which is taken as the minimum between these attacks.

Low multiplicative depth of circuit is the largest advantage of FLIP. In most previous schemes, the noise produced byhomomorphically evaluating decryption circuit is too high to perform more homomorphic operation.

2.2 Batchfully homomorphic encryption over the integers

In the original DGHV scheme[2], a ciphertext has the form $c = q \cdot p + 2r + m$, where p is the secret key, q is a large random integer, and r is a small random integer (noise), the plaintext bit $m \in \{0,1\}$ is recovered by computing $m = [c \bmod p] \bmod 2$.

We use the Chinese Remainder Theorem to pack l bits m_i into a single ciphertext with a set of $l+1$ coprime integers q_0, $p_0, \cdots, p_{l-1}$. The batch ciphertext has the form

$$c = CRT_{q_0, p_0, \ldots, p_{l-1}}(q, 2r_0 + m_0, \ldots, 2r_{l-1} + m_{l-1})$$

where $q = c \bmod q_0$, $2r_{i-1} + m_{i-1} = c \bmod p_{i-1}$ for $1 \leqslant i \leqslant l$. Therefore, the addition or multiplication of two ciphertext produces a new ciphertext, which is decrypted to the componentwise sum or product modulo 2 of the original plaintexts.

As for public-key encryption, the original DGHV scheme masks the message m with a random subset sum of the public key elements $x_j = q_j \cdot p + r_j$, has the form

$$c = [m + 2r + 2\sum_{j \in S} x_j]_{x_0}$$

with $x_j = q_j \cdot p + r_j$. Decryption outputs $m \leftarrow [c \bmod p]_2$.

Next we describe the batch DGHV scheme (BDGHV) [10]. For a real number x, we respectively denate $\lceil x \rceil$, $\lfloor x \rfloor$ and $\lceil x \rfloor$ as the upper, lower, and nearest integer part of x. For integers z and p, we denate z mod p by $[z]_p$ with $-p/2 < [z]_p \leqslant p/2$. Let λ be the security parameter, τ be the number of elements in the public key, γ the bit-length of integers x_i in the public key, η the bit-length of secret key p and ρ the bit-length of the noise r_i, ρ' the bit-length of the noise in a refresh ciphertext.

BDGHV. KeyGen(1^λ). Generate a collection of l random η-bit prime $p_j, 0 \leqslant j < l$, and denote π their product. We define a noise-free public key element $x_0 = q_0 \cdot \pi$, that does not contain prime factor and is less than 2^{λ^2}, where $q_0 \leftarrow \mathbb{Z} \cap [0, 2^\gamma / \pi)$.

Generate the integers x_i, x_i' and Π_i with a quotient by π uniformly and independently distributed in $\mathbb{Z} \cap [0, q_0)$, and with the following distribution modulo p_j for $0 \leqslant j < l$:

$1 \leqslant i \leqslant \tau$, $x_i \bmod p_j = 2r_{i,j}$

$0 \leqslant i \leqslant l-1$, $x_i' \bmod p_j = 2r_{i,j}' + \delta_{i,j}$

$0 \leqslant i \leqslant l-1$, $\Pi_i \bmod p_j = 2\tilde{\omega}_{i,j} + \delta_{i,j} \cdot 2^{\rho'+1}$, with $r_{i,j} \leftarrow \mathbb{Z} \cap (-2^{\rho'-1}, 2^{\rho'-1})$, and $r_{i,j}', \tilde{\omega}_{i,j} \leftarrow \mathbb{Z} \cap (-2^\rho, 2^\rho)$

Finally, let $pk = \{x_0, (x_i)_{1 \leqslant i \leqslant \tau}, (x_i')_{0 \leqslant i \leqslant l-1}, (\Pi_i)_{0 \leqslant i \leqslant l-1}\}$ and $sk = (p_j)_{0 \leqslant j \leqslant l-1}$.

BDGHV. Encrypt($pk, m \in \{0,1\}^l$). Choose random integers vectors $b = (b_i) \in (-2^\alpha, 2^\alpha)^\tau$ and $b' = (b_i')_{0 \leqslant i \leqslant l-1} \in (-2^{\alpha'}, 2^{\alpha'})^l$ and output the ciphertext:

$$c = [\sum_{i=0}^{l-1} m_i \cdot x_i' + \sum_{i=0}^{l-1} b_i' \cdot \Pi_i + \sum_{i=1}^{\tau} b_i \cdot x_i]_{x_0}$$

BDGHV. Decrypt (sk, c). Output $m = (m_0, \ldots, m_{l-1})$ where $m_j \leftarrow [c]_{p_j} \bmod 2$.

BDGHV. Add(pk, c_1, c_2). Output $c_1 + c_2 \bmod x_0$.

BDGHV. Mult (pk, c_1, c_2). Output $c_1 \cdot c_2 \bmod x_0$.

The parameters in the scheme must satisfy the following constraints:

(1) $\rho = \Omega(\lambda)$ to avoid brute force attack on the noise[19-20],

(2) $\eta \geqslant \rho \cdot \Theta(\lambda \log^2 \lambda)$ for homomorphically evaluating the "squashed decryption" circuit,

(3) $\eta \geqslant \alpha' + \rho' + 1 + \log(l)$ for correct decryption,

(4) $\gamma = \omega(\eta^2 \cdot \log \lambda)$ to avoid lattice-based attacks[2,21],

(5) $\rho' \geqslant \rho + \lambda$ and $\alpha' \geqslant \alpha + \lambda$ in order to prove the semantic security,

(6) $\alpha \cdot \tau \geqslant \lambda + \gamma$ and $\tau \geqslant l \cdot (\rho' + 2) + \lambda$ in order to apply leftover hash lemma to prove the semantic security of the scheme.

To satisfy these constraints above, we can take $\rho = 2\lambda$, $\eta = \tilde{O}(\lambda^2)$, $\gamma = \tilde{O}(\lambda^5)$, $\alpha = \tilde{O}(\lambda^2)$, $\tau = \tilde{O}(\lambda^3)$ as in References [18], with $\rho' = \tilde{O}(\lambda)$, $\alpha' = \tilde{O}(\lambda^2)$ and $l = \tilde{O}(\lambda^2)$, where λ is the security parameter.

3 Batch FHE-symmetric encryption framework

In this section, we describe our contribution on batch FHE-symmetric encryption framework. Using Chinese remainder theorem, we can pack l ciphertexts $E_k(m_0), \cdots, E_k(m_{l-1})$ into a single C, then send $C' = (HE_{pk}(k), C)$ to the cloud. Furthermore, we give out an instance of batch GSW13-FLIP framework, and analyze the effect of this improvement on computational efficiency.

3.1 Description

In original compressed encryption, we encrypt m with a symmetric encryption scheme E, and secret key k with a fully homomorphic encryption. Then the compressed ciphertext $c' = (HE_{pk}(k), E_k(m))$ is sent to the cloud instead of $HE_{pk}(m)$. Receiving c', Cloud utilizes the homomorphic property of FHE to homomorphically evaluate the decryption circuit,

$$C_{E^{-1}}(HE_{pk}(k), c) = E^{-1}(HE_{pk}(k), HE_{pk}(c)) = HE_{pk}(E^{-1}(k, c)) = HE_{pk}(m)$$

As for l messages $m_i, 0 \leqslant i \leqslant l-1$, we send compressed ciphertext $c'=(HE_{pk}(k),E_k(m_i))$, then l times homomorphic evaluation are needed to recover the $HE_{pk}(m_i)$.

Considering that ciphertexts $c_i=E_k(m_i)$ are single-bit, we can imitate batch FHE over integers, and use the Chinese remainder theorem to pack c_i into a C,

$C=CRT_{q_0,p_0,\ldots,p_{l-1}}(q,2r_0+c_0,\ldots,2r_{l-1}+c_{l-1})$ with a set of $l+1$ coprime integers $q_0,p_0,\cdots,p_{l-1}$, and $c\equiv q \bmod q_0$, $C\equiv q\cdot p_{i-1}+2r_{i-1}+c_{i-1}$ for $1\leqslant i\leqslant l$[10]. The new compressed ciphertext is $(HE_{pk}(k),C)$, we can recover c_i by $(C \bmod p_i) \bmod 2$.

Next, the main problem is how to recover the $HE_{pk}(m_i)$ when given the $(HE_{pk}(k),C)$. It is clear that C and c_i have homomorphic property, our operation on C can be reflected on c_i, i. e. $Dec(Evaluat_f(C))=f(c_i)$. Treating decryption circuit as a function, ciphertext and secret key are inputs. Similarly, $C_{E^{-1}}$ can be seen as a function, $HE_{pk}(k)$ and C are inputs. Utilizing the homomorphic property, there are two steps to get $HE_{pk}(m_i)$:

(1) $C'=C_{E^{-1}}(HE_{pk}(k),C)$,

(2) $HE_{pk}(m_i)=(C' \bmod p_i) \bmod 2$

First, we homomorphically evaluate decryption circuit with $HE_{pk}(k)$ and C to get C', it means that we take a functional operation on C, which can be reflected on c_i. Then, according to the relationship between C and c_i, we can simply recover $HE_{pk}(m_i)$ by $(C' \bmod p_i) \bmod 2$. Introducing an extra packing operation, we only need homomorphically evaluating decryption circuit for once, which is l times in previous view.

We can also achieve the operation between the slots to get $HE_{pk}(m_i)+HE_{pk}(m_j)$ and $HE_{pk}(m_i)\cdot HE_{pk}(m_j)$. For two packed ciphertexts C_1 and C_2, we pack $(c_0,c_1,\ldots,c_{l-1})$ into C_1 and $(c'_0,c'_1,\ldots,c'_{l-1})$ into C_2, i. e.

$$C_1=CRT_{q_0,p_0,\ldots,p_{l-1}}(q,2r_0+c_0,\ldots,2r_{l-1}+c_{l-1})$$
$$C_2=CRT_{q_0,p_0,\ldots,p_{l-1}}(q,2r_0+c'_0,\ldots,2r_{l-1}+c'_{l-1})$$

Here, we respectivelydenote $E_k(m_i)$ and $E_k(m'_i)$ as c_i and c'_i, $1\leqslant i\leqslant l$. As introduced above, there are homomorphic properties between C_1 and c_i, as well as C_2 and c'_i.

$$C_1=p_i\cdot q+2r_i+c_i,\ C_2=p_i\cdot q+2r'_i+c'_i$$

It is clear that,

$$C_1+C_2\xrightarrow{(\bmod p_i)\bmod 2}c_i+c'_i$$

$$C_1\cdot C_2\xrightarrow{(\bmod p_i)\bmod 2}c_i\cdot c'_i$$

According to homomorphic properties, the operation we process on C_1+C_2 and $C_1\cdot C_2$ can be reflected on $c_i+c'_i$ and $c_i\cdot c'_i$. That is to say, we can homomorphically evaluate the decryption circuit $C_{E^{-1}}$ with C_1+C_2 and $C_1\cdot C_2$ to get $C_{E^{-1}}(c_i+c'_i)$ and $C_{E^{-1}}(c_i\cdot c'_i)$.

$$C_{E^{-1}}(C_1+C_2)\xrightarrow{(\bmod p_i)\bmod 2}C_{E^{-1}}(c_i+c'_i)$$

$$C_{E^{-1}}(C_1\cdot C_2)\xrightarrow{(\bmod p_i)\bmod 2}C_{E^{-1}}(c_i\cdot c'_i)$$

In the process of recovering the $HE_{pk}(m_i)$, we use the homomorphic properties. When it comes to operate $C_{E^{-1}}$ on $c_i+c'_i$.

$$\begin{aligned}C_{E^{-1}}(c_i+c'_i)=&\\ &E^{-1}(HE_{pk}(k),HE_{pk}(c_i+c'_i))=\\ &E^{-1}(HE_{pk}(k),HE_{pk}(c_i)+HE_{pk}(c'_i))=\\ &E^{-1}(HE_{pk}(k),HE_{pk}(c_i))+\\ &E^{-1}(HE_{pk}(k),HE_{pk}(c'_i))=\\ &HE_{pk}(m_i)+HE_{pk}(m'_i)\end{aligned}$$

It is the same for $c_i\cdot c'_i$, $C_{E^{-1}}(c_i\cdot c'_i)=HE_{pk}(m_i)\cdot HE_{pk}(m'_i)$. By this way, we can parallel operate on l slots at the same time, which reduces a lot of extra calculation. We can get all of the $HE_{pk}(m_i)+HE_{pk}(m'_i)$ and $HE_{pk}(m_i)\cdot HE_{pk}(m'_i)$ through only one homomorphic evaluation.

Next, we give out an instance of batch GSW13-FLIP framework, in this context, the efficiency of batch scheme and original scheme is analyzed in detail.

3.2 Efficiency analysis

3.2.1 Original GSW13-FLIP scheme. In the absence of packing operation, when we need to transmit l compressed ciphertexts, we usually take l times homomorphic evaluation of decryption circuit to recover $HE_{pk}(m_i)$, which causes a great deal of calculation.

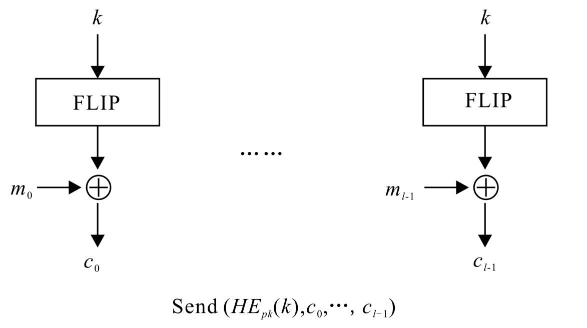

Fig. 2 l Compressed ciphertexts

Given compressed ciphertext $(HE_{pk}(k), c_0, \ldots, c_{l-1})$, we then homomorphically evaluate the decryption circuit $C_{E^{-1}}$ to recover the $HE_{pk}(m_i) = C_{E^{-1}}(HE_{pk}(k), c_i)$. In this construction, the cost of decomp-ression per plaintext is fixed and roughly equals one single evaluation of the decryption circuit; more specifically, the multiplicative depth of the decompression circuit is also fixed and equals to the decryption circuit.

As showing in Tab. 1, FLIP $(46, 136, 4T_{15})$ and FLIP $(86, 238, 5T_{23})$ separately satisfy 80- and 128-bit security. That is to say, when security parameter $\lambda = 80$, we choose parameters as FLIP $(46, 136, 4T_{15})$. It denotes the instance of FLIP with 662 inputs, linear part of $n_1 = 46$ bits, quardratic part of $n_2 = 136$ bits and 4 triangular functions of degree $k = 15$; when security parameter $\lambda = 128$, FLIP $(86, 238, 5T_{23})$ denotes the instance with 1704 inputs, the linear and quardratic parts respectively of $n_1 = 86$ and $n_2 = 238$ bits, and 5 triangular functions of degree $k = 21$.

We recall that three parts of filter function respectively have n_1, n_2, n_3 variables.

$$f_1(x_0, \ldots, x_{n_1-1}) = L_{n_1} = \sum_{i=0}^{n_1-1} x_i$$

$$f_2(x_{n_1}, \ldots, x_{n_1+n_2-1}) =$$

$$Q_{n_2/2} = \sum_{i=0}^{n_2/2-1} x_{2i} x_{2i+1}$$

$f_3(x_{n_1+n_2}, \ldots, x_{n_1+n_2+n_3-1})$ is the direct sum of nb triangular function T_k,

$$T_k(x_0, \ldots, x_{\frac{k(k+1)}{2}-1}) = \sum_i^k \prod_{j=0}^{i-1} x_{j+\sum_{l=0}^{i-1} l}$$

As for FLIP $(46, 136, 4T_{15})$, security parameter λ is 80, multiplication comes mainly from quardratic and triangular parts of filter functions, that is 432. Moreover, the number of addition is 172. Similarly, security parameter λ of FLIP $(86, 238, 5T_{23})$ is 128, the number of multiplication is 1274 and the number of addition is 313. Obviously, gate computational complexity of homomorphically evaluating decryption is $\widetilde{O}(\lambda)$. To satisfy the constraints of batch scheme, we take $l = \widetilde{O}(\lambda^2)$. So that, the computational complexity of decompression is roughly $\widetilde{O}(\lambda^3)$.

3.2.2 Batch GSW13-FLIP scheme. Utilizing the Chinese remainder theorem to pack c_i into a C, we only need to homomorphically evaluate decryption circuit for once, but introduce the new computation of BDGHV.

Tab. 2 Capability of somewhat and fully BDGHV scheme

Scheme	Security	Hard problem	Gate computational complexity
Somewhat BDGHV	IND-CPA	Error-free approximate-GCD, SSSP	$\widetilde{O}(\lambda^{1.5})$
Fully BDGHV	IND-CPA	Error-free approximate-GCD, SSSP	$\widetilde{O}(\lambda^{1.5})$

As showing in Tab. 2[22], both schemes have gate computational complexity of $\widetilde{O}(\lambda^{1.5})$.

Comparing the original scheme with the batch scheme, although the computational complexity of a homomorphic evaluation is $\widetilde{O}(\lambda)$, less than $\widetilde{O}(\lambda^{1.5})$ that of BDGHV scheme, we have to homomor phically evaluate decryption circuit for $l-1$ more times to recover all $HE_{pk}(m_i)$. So that the computational complexity of original scheme is roughly $\widetilde{O}(\lambda^3)$. By packing c_i into a C, we only need once homomorphic evaluation to get C', the extra introduced operation has computational complexity of $\widetilde{O}(\lambda^{1.5})$. Given C', We can simply get $HE_{pk}(m_i)$ by $(C' \bmod p_i) \bmod 2$. Twice multiplication is required for once modular operation, therefor the computational complexity of batch scheme is roughly $\widetilde{O}(\lambda^2)$. Through packing operation, we reduce the computational complexity from $\widetilde{O}(\lambda^3)$ to $\widetilde{O}(\lambda^2)$.

When we need $HE_{pk}(m_i)+HE_{pk}(m_i')$ and $HE_{pk}(m_i)\cdot HE_{pk}(m_i')$, utilizing the parallel operation on slots, we can just operate once homomorphic evaluation of decryption circuit on C_1+C_2 and $C_1\cdot C_2$ instead of $2l$ times in original scheme.

4 Conclusions

In the context of fully homomorphic encryption-symmetric encryption framework, we usually send compressedciphertext $c'=(HE_{pk}(k), E_k(m_i))$ to the cloud to get over the expansion of homomorphic ciphertexts, then cloud homomorphically evaluate decryption circuit l times to recover all $HE_{pk}(m_i)$, where $0\leqslant i\leqslant l-1$. In this process, a great deal of operations are consumed on homomorphic evaluation.

In this paper, utilizing the Chinese remainder theorem to pack c_i into a C, we extend the fully homomorphic encryption-symmetric encryption framework into a batch one, we only need to homomorphically operate $C_{E^{-1}}$ on C, which can be reflected on c_i by the homomorphic property between c_i and C. So that, while introducing an extra operation of BDGHV, we can reduce the homomorphic evaluation of decryption to only once. Meanwhile, we can also parallel operate components in packed ciphertexts to get $HE_{pk}(m_i)+HE_{pk}(m_i')$ and $HE_{pk}(m_i)\cdot HE_{pk}(m_i')$ through $C_{E^{-1}}(C_1+C_2)$ and $C_{E^{-1}}(C_1\cdot C_2)$.

More specifically, we give out an instance of batch GSW13-FLIP framework, analyze the computational complexity of original scheme and the batch one. Although the computational complexity of once homomorphic evaluation of decryption is less than the complexity of BDGHV, we have to operate l times to recover all $HE_{pk}(m_i)$, $0\leqslant i\leqslant l-1$. Compared to the original scheme, our contribution of batch scheme can reduce the computational complexity from $\widetilde{O}(\lambda^3)$ to $\widetilde{O}(\lambda^2)$. And the more message we transmit, the more benefit we get from the batch scheme.

Beside this, we can operate our batch technique on other fully homomorphic encryption-symmetric encryption frameworks for different FHE schemes or symmetric encryption schemes. As for parallel operation, we have implemented the calculation of the corresponding components in slots of different packingciphertexts. Utilizing the pattern in Ref. [23], we can also realize the calculation of different components in a packing ciphertext.

References:

[1] Gentry C. Fully homomorphic encryption using ideal lattices [J]. Stoc, 2009, 9: 169.

[2] Dijk M V, Gentry C, Halevi S, *et al*. Fully homomorphic encryption over the integers [M]. Advances in Cryptology- EUROCRYPT 2010. Berlin: Springer Berlin Heidelberg, 2010.

[3] Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (Standard) LWE [C]// Foundations of Computer Science. [S. l.]: IEEE, 2011.

[4] Brakerski Z, Vaikuntanathan V. Fully homomorphic encryption from ring-lwe and security for key dependent messages [C]//Cryptology Conference. Berlin: Springer Berlin Heidelberg, 2011.

[5] Gentry C, Sahai A, Waters B. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based [C]//Advances in Cryptology-CRYPTO 2013. Berlin: Springer Berlin Heidelberg, 2013.

[6] Peng X B, Li Q S, Wang L Z, *et al*. Research progress of privacy preserving support vector machines [J]. J Jiangsu Univ: Nat Sci Ed, 2017, 38: 78 (in Chinese).

[7] Ji Y M, Zhu T H, Chai B Z, *et al*. Hybrid encryption scheme and performance analysis for user's privacy in cloud [J]. J Chongqing Univ Posts Telecommun: Nat Sci Ed, 2015, 27: 631 (in Chinese).

[8] Naehrig M, Lauter K, Vaikuntanathan V. Can homomorphic encryption be practical? [C]// ACM Cloud Computing Security Workshop, Ccsw 2011. Chicago: DBLP, 2011.

[9] Gentry C, Halevi S, Smart N P. Homomorphic evaluation of the AES circuit [J]. Lect Notes Comp Sci, 2012, 7417: 850.

[10] Cheon J H, Coron J S, Kim J, *et al*. Batch fully homomorphic encryption over the integers [C]// In-

ternational Conference on the Theory and Applications of Cryptographic Techniques. Berlin: Springer Berlin Heidelberg, 2013.

[11] Doröz Y, Hu Y, Sunar B. Homomorphic AES evaluation using the modified LTV scheme [M]. Hague, Holland: Kluwer Academic Publishers, 2016.

[12] Lepoint T, Naehrig M. A comparison of the homomorphic encryption schemes FV and YASHE [C]// Progress in Cryptology -AFRICACRYPT 2014. Berlin: Springer International Publishing, 2014.

[13] Doröz Y, Shahverdi A, Eisenbarth T, *et al*. Toward practical homomorphic evaluation of block ciphers using prince [C]// International Conference on Financial Cryptography & Data Security. Berlin: Springer Berlin Heidelberg, 2014.

[14] Albrecht M R, Rechberger C, Schneider T, *et al*. Ciphers for MPC and FHE [M]// Advances in Cryptology-EUROCRYPT 2015. Berlin: Springer Berlin Heidelberg, 2015.

[15] Canteaut A, Carpov S, Fontaine C, *et al*. Stream ciphers: a practical solution for efficient homomorphic-ciphertext compression [J]. J Cryptol, 2016 (3): 313.

[16] Méaux P, Journault A, Standaert F X, *et al*. Towards stream ciphers for efficient fhe with low-noise ciphertexts [C]// Advances in Cryptology-EUROCRYPT 2016. Berlin: Springer Berlin Heidelberg, 2016.

[17] Canteaut, A, Carpov, S, Fontaine, C, *et al*. How to compress homomorphic ciphertexts [C]// Fast Software Encryption FSE 2016. [S. l.]: [s. n.], 2016.

[18] Duval S, Lallemand V, Rotella Y. Cryptanalysis of the FLIP, family of stream ciphers [C]// Cryptology Conference. Berlin: Springer Berlin Heidelberg, 2016.

[19] Chen Y, Nguyen P Q. Fasteralgorithms for approximate common divisors: breaking fully-homomorphic-encryption challenges over the integers [C]// Advances in Cryptology-EUROCRYPT 2012. Berlin: Springer Berlin Heidelberg, 2012.

[20] Coron J, Naccache D, Tibouchi M. Public key compression and modulus switching for fully homomorphic encryption over the integers [M]// Advances in Cryptology-EUROCRYPT 2012. Berlin: Springer Berlin Heidelberg, 2012: 446.

[21] Coron J, Mandal A, Naccache D, *et al*. Fully homomorphic encryption over the integers with shorter public keys [C]// Conference on Advances in Cryptology. Berlin: Springer-Verlag, 2011.

[22] Luo B C, Liu Q, Yuan M A, *et al*. Batch fully homomorphic encryption over integers with shorter public keys [J]. Appl Res Comput, 2014, 8:1180.

[23] Gentry C, Halevi S, Smart N P. Fully homomorphic encryption with polylog overhead [C]// Advances in Cryptology-EUROCRYPT 2012. Berlin: Springer Berlin Heidelberg, 2012.

引用本文格式:

中　文：陶梦龙，胡斌. 基于全同态加密与对称加密融合的批处理研究[J]. 四川大学学报：自然科学版，2019，56：857.

英　文：Tao M L，Hu B. Research based on batch fully homomorphic encryption-symmetric encryption [J]. J Sichuan Univ: Nat Sci Ed, 2019, 56: 857.