A compression method for homomorphic ciphertexts

S. Carpov, R. Sirdey

CEA, LIST,
L3S
Point Courrier 172, 91191 Gif-sur-Yvette Cedex, France

Abstract. In this work we describe a message packing and unpacking method for homomorphic ciphertexts. Messages are packed into the coefficients of plaintext polynomials. We propose an unpacking procedure which allows to obtain a ciphertext for each packed message. The packing and unpacking of ciphertexts represents a solution for reducing the transmission bottleneck in cloud based applications, in particular when sending homomorphic calculations results. The results we obtain (packing ratio, unpacking time) are compared to existing packing methods based on trans-ciphering.

1 Introduction

A homomorphic encryption scheme allows to perform computations on encrypted data such that during the computation the evaluator never has access to the decrypted data. Since the seminal work of Gentry [Gen09], introducing the first fully homomorphic encryption (FHE) many other simpler and more efficient schemes have been proposed. The spectrum of applications of homomorphic encryption is rather large, in particular in the domain of cloud-based applications [NLV11,GLN12,LLAN]. We start by giving some literature reference on message packing methods for homomorphic ciphertexts, on existing homomorphic encryption schemes and operations they support. Afterwards we describe message packing and unpacking methods together with some experimental results. Finally, some conclusions and future works are given.

1.1 Ciphertext message packing

In the context of cloud-based applications the client sends data encrypted under a homomorphic encryption (HE) scheme to the cloud. The cloud evaluates an algorithm on encrypted data and sends the obtained ciphertexts (algorithm execution results) back to the client. All the HE schemes proposed so far have a huge ciphertext expansion factor (ciphertext size to payload ratio). In practice the uplink (client to cloud) and the downlink (cloud to client) transmission of encrypted data is a bottleneck in the applicability of HE.

The first ideas for decreasing HE ciphertext expansion factor were described in [NLV11], in particular the authors proposed to send uplink data encrypted using a symmetric encryption scheme. On the cloud side the data is *trans-ciphered* from the symmetric encryption to the HE. The trans-ciphering works only in one direction (symmetric encryption to HE).

The AES can be used as symmetric encryption scheme for trans-ciphering (refer to [GHS12b,CCK$^+$,CDS15]). Resulting HE ciphertexts will have a multiplicative depth of 40 without any computation performed on data yet. A more recent work [CCF$^+$15] studies symmetric encryption schemes which are "HE friendly", thus which can generate HE ciphertexts with smaller multiplicative depths. The authors propose a family of symmetric encryption schemes based on Trivium with multiplicative depth 12.

In batched HE schemes [GHS12a] one can pack multiple plaintext messages into the slots of a single ciphertext. Using slot permutation/rotation procedure one can recover each message into a separate ciphertext and execute a circuit. After the execution, one can reassemble circuit output ciphertexts into a single ciphertext. This method allows to send almost fresh ciphertexts[1] and support downlink/uplink transmission. Although when compared to trans-ciphering it is less efficient in terms of the expansion factor of the sent data.

A third method, proposed in [NLV11], is to build a ciphertext encrypting a polynomial which coefficients are plaintext messages to be packed. This method can be used only for downlink transmissions because, as the authors stated, there is no method to unpack such a ciphertext. In terms of data traffic this packing method is more efficient than the batched one because the number of slots in batched schemes depends on the factorization of the ring modulus polynomial, which is always inferior to the number of polynomial coefficients (except for specific plaintext spaces).

In this work, we introduce an improvement for the message packing into polynomial coefficients method. We use finite field extension as plaintext spaces and propose a method which allows to unpack messages in the ciphertext domain (using homomorphic operations). The proposed method also works for ring extensions plaintext spaces. Still, in order to lighten the discourse we limit ourselves to the finite field extension case. This packing/unpacking method can also be used to combine execution of boolean circuits and arithmetic circuits over finite fields. As for example to execute in the same ciphertext domain an efficient version of the AES circuit (over finite field $\mathbb{F}_{2^8}$ as in [GHS12b]) and an additional (payload) boolean circuit. We limit our discourse only to ring learning with errors (ring-LWE) based HE schemes, because the proposed packing method needs a polynomial ring plaintext space.

1.2 Homomorphic encryption schemes

Let $\mathbb{A} = \mathbb{Z}[X]/\Phi(X)$ be the ring of integers modulo an irreducible polynomial $\Phi(X)$. Let $\mathbb{A}_q = \mathbb{A}/q\mathbb{A}$ be the set of integer polynomials reduced modulo the

[1] Slot switching increases the noise in the ciphertext, although much less than trans-ciphering.

polynomial $\Phi(X)$ and with coefficients reduced modulo q. In ring-LWE based cryptosystems ciphertexts and secret keys are elements (vector of elements) from the ring $\mathbb{A}_q$.

Leveled HE schemes [BGV12] use a series integer modulus $q_0, q_1, \ldots$ for ciphertexts at different moments of homomorphic evaluation. A modulus switching technique is used (switch ciphertext from modulus q_i to modulus q_{i+1}, $q_i > q_{i+1}$) to deal with the noise increase. In [Bra12] a notion of scale-invariance for leveled HE schemes is introduced. In scale-invariant schemes a single modulus, q, is used for ciphertexts during the whole homomorphic evaluation.

The plaintext space in this HE schemes is the ring of polynomials $\mathbb{A}_t$. In the first schemes, only one binary message was encrypted in a ciphertext (the zero-degree coefficient of a polynomial from $\mathbb{A}_2$). This allowed to evaluate homomorphically arbitrary boolean circuits. Using a larger modulus ($t > 2$) for the plaintext space it is possible to execute operations on integers modulo t homomorphically. One also can encrypt finite field elements $\mathbb{F}_{t^n}$ into ciphertexts when t is prime, polynomial $\Phi(X)$ is irreducible modulo t and n divides the degree of $\Phi(X)$.

In a batched schemes [GHS12a] the plaintext space ring $\mathbb{A}_t$ can be factored into sub-rings (defined by the factorization of the polynomial $\Phi(X)$ modulo t) such that addition and multiplication applies to each sub-ring independently. Batching several messages into ciphertext slots allows to execute homomorphic operations on all messages at once. Using ring homomorphism operations one can permute plaintext messages in a ciphertext.

To summarize, ring-LWE based HE schemes allow to execute homomorphic operations (batched or not) over: (i) finite field $\mathbb{F}_{t^n}$ elements or (ii) polynomial ring $\mathbb{A}_t$ elements (includes the integer modulo ring case).

1.3 Homomorphic operations

A homomorphic encryption scheme is described by a set of operations. We try to give a generic list of operations supported by any ring-LWE leveled HE scheme ignoring implementation details. We limit our discourse to non-batched schemes, but without loss of generality the results presented in this paper are also valid for batched schemes.

$\texttt{KeyGen}\left(1^{\lambda}\right)$ – generate the set of keys: a secret key sk used for encrypting/decrypting messages, a public key pk used for encrypting messages and an additional set of evaluation keys evk (for key-switching in homomorphic multiplication and ring homomorphism operations).

$\texttt{Enc}_{pk}(m)$ – encrypts a plaintext message $m \in \mathbb{A}_t$ using the public key pk.

$\texttt{Dec}_{sk}(\texttt{ct})$ – decrypts a ciphertext $\texttt{ct}$ using the secret key sk.

$\texttt{Add}(\texttt{ct}_1, \texttt{ct}_2)$ – outputs a ciphertext which represents the addition of plaintext messages encrypted by $\texttt{ct}_1$ and $\texttt{ct}_2$:
$\texttt{Dec}_{sk}(\texttt{Add}(\texttt{ct}_1, \texttt{ct}_2)) \equiv \texttt{Dec}_{sk}(\texttt{ct}_1) + \texttt{Dec}_{sk}(\texttt{ct}_2)$

$\texttt{Mult}(\texttt{ct}_1, \texttt{ct}_2, evk)$ – outputs a ciphertext which represents the multiplication of plaintext messages encrypted by $\texttt{ct}_1$ and $\texttt{ct}_2$:
$\texttt{Dec}_{sk}(\texttt{Mult}(\texttt{ct}_1, \texttt{ct}_2, evk)) \equiv \texttt{Dec}_{sk}(\texttt{ct}_1) \cdot \texttt{Dec}_{sk}(\texttt{ct}_2)$

$\texttt{Frob}(\texttt{ct}, i, evk)$ – outputs a ciphertext which represents the application of the Frobenius endomorphism ϕ^i $(X \mapsto X^{t^i})$, $1 < i < n$, over the plaintext message encrypted by $\texttt{ct}$:
$\texttt{Dec}_{sk}(\texttt{Frob}(\texttt{ct}, i, evk)) \equiv \phi^i(\texttt{Dec}_{sk}(\texttt{ct}))$
The Frobenius endomorphism operation (or Frobenius powering) can be performed with small noise increase (much smaller than simple ciphertext squaring) for some HE scheme instantiations.

To lighten the discourse we use addition and multiplication operators for homomorphic addition and multiplication of ciphertexts: $\texttt{ct}_1 + \texttt{ct}_2 = \text{Add}(\texttt{ct}_1, \texttt{ct}_2)$ and respectively $\texttt{ct}_1 \cdot \texttt{ct}_2 = \texttt{Mult}(\texttt{ct}_1, \texttt{ct}_2, evk)$. The same applies for Frobenius endomorphism: $\phi^i(\texttt{ct}) = \texttt{Frob}(\texttt{ct}, i, evk)$. Evaluation key evk use is implicit in operator notation. We recall that all the arithmetic operations are performed over the plaintext space ring $\mathbb{A}_t$.

Homomorphic addition and multiplication can be applied to a plaintext and a ciphertext, i.e. $\texttt{ct}_1 + m_2$ means an addition between the ciphertext $\texttt{ct}_1$ and the plaintext message m_2. Such an homomorphic operation shall be denoted as a plaintext-ciphertext homomorphic operation. The noise increase of a plaintext-ciphertext operation is lower when compared to the noise increase of this operation applied onto ciphertexts.

2 Ciphertext packing

Let $m_i \in \mathbb{Z}_t$, $0 \leq i < n$, be n plaintext messages. The messages are packed as coefficients of a polynomial $m(X)$:

$$m(X) = m_0 + m_1 \cdot X + \ldots + m_{n-1} \cdot X^{n-1} \tag{1}$$

We suppose that n is inferior to the degree of the plaintext space ring $\mathbb{A}_t$, so that $m(X) \in \mathbb{A}_t$. Afterwards polynomial $m(X)$ is encrypted and a ciphertext which packs n plaintext messages is obtained: $\texttt{ct}_{pack} = \texttt{Enc}_{pk}(m(X))$. We shall note that this packing method was used in [NLV11] to optimize the homomorphic addition or multiplication of several numbers.

The same packing methodology can be used to pack ciphertexts. Suppose n ciphertexts $\texttt{ct}_i$, $0 \leq i < n$, are given. The ciphertext $\texttt{ct}_{pack}$ which packs ciphertexts $\texttt{ct}_i$, $0 \leq i < n$, is computed using expression:

$$\texttt{ct}_{pack} = \texttt{ct}_0 + \texttt{ct}_1 \cdot X + \ldots + \texttt{ct}_{n-1} \cdot X^{n-1}$$

The packing of ciphertexts can be done using n plaintext-ciphertext homomorphic multiplications and $n-1$ homomorphic additions. The noise increase is mainly due to the homomorphic sums and lesser to plaintext-ciphertext multiplications. If we suppose that the ciphertext $\texttt{ct}_i$ encodes the plaintext message $m_i (\in \mathbb{Z}_t)$ then $\texttt{Dec}_{sk}(\texttt{ct}_{pack}) \equiv m_0 + m_1 \cdot X + \ldots + m_{n-1} \cdot X^{n-1}$.

3 Ciphertext unpacking

Let $\mathtt{ct}_{pack}$ be a ciphertext in which are packed n messages as in relation (1). We call unpacking the procedure of extracting ciphertexts $\mathtt{ct}_i$, $0 \leq i < n$, such that $\mathtt{Dec}_{sk}(\mathtt{ct}_i) = m_i$. Unpacking such a ciphertext is a bit trickier than packing. Unpacking is possible for plaintext rings $\mathbb{A}_t$ with a prime characteristic, thus for finite field plaintext spaces[2]. In what follows we suppose that $\mathbb{A}_t$ is an extension finite field $\mathbb{F}_{t^n}$. We introduce a method for extracting messages from a packed message in the plaintext domain. This method can be applied in the ciphertext space also using homomorphic operations.

Let $\mathtt{Extr}_i : \mathbb{F}_{t^n} \to \mathbb{F}_t$ be a function which extracts the i-th coefficient of the polynomial in argument. For example when function $\mathtt{Extr}_i$ is applied to a polynomial which packs messages as in (1) message m_i is obtained:

$$\mathtt{Extr}_i\left(m(X) = m_0 + m_1 \cdot X + \ldots + m_{n-1} \cdot X^{n-1}\right) = m_i$$

Function $\mathtt{Extr}_i$ is an affine transformation over the standard power basis of $\mathbb{F}_{t^n}$. We use the fact that an affine transformation over the standard power basis can be computed as a $\mathbb{F}_{t^n}$ affine transformation over the conjugates (i.e. Frobenius endomorphisms ϕ^j, $0 \leq j < n$). The extraction function can be expressed as $\mathtt{Extr}_i(m) = \sum_{j=0}^{n-1} a_{i,j} \cdot \phi^j(m)$, $a_{i,j} \in \mathbb{F}_{t^n}$. There is no constant term because $\mathtt{Extr}_i(0) = 0$.

Suppose for the moment that coefficients $a_{i,j}$ are given. As said earlier because of scheme homomorphism functions $\mathtt{Extr}_i$ can be evaluated directly on a packed ciphertext in order to obtain n unpacked ciphertexts. In particular, given a ciphertext $\mathtt{ct}_{pack}$ the encryption of the i-th message can be found using expression:

$$\mathtt{ct}_i = \mathtt{Extr}_i(\mathtt{ct}_{pack}) = a_{i,0} \cdot \mathtt{ct}_{pack} + a_{i,1} \cdot \phi(\mathtt{ct}_{pack}) + \ldots + a_{i,n-1} \cdot \phi^{n-1}(\mathtt{ct}_{pack}) \quad (2)$$

In terms of homomorphic operations the evaluation of an extraction function require $n-1$ Frobenius power operations, n ciphertext-plaintext multiplications and $n-1$ homomorphic sums. We note that the conjugates of $\mathtt{ct}_{pack}$ are computed only once for all extractions. When the homomorphic encryption scheme supports Frobenius powering the noise increase is low, when this is not the case then we can perform the power operations using homomorphic multiplications. In the latter case, the minimal multiplicative depth for computing conjugates $\phi^j(\mathtt{ct}_{pack})\left(= \mathtt{ct}_{pack}^{t^j}\right)$, $0 \leq j < n$, is $\lceil \log_2 n \rceil + \lceil \log_2 t \rceil$. Although there is no need to compute all the extraction functions (in the case the number of packed messages is inferior to n), however all the conjugates must be computed. In the case of extension fields of large degree computing the conjugates by homomorphic multiplications will have a huge impact on performance.

[2] As said earlier unpacking is also possible for ring extensions $\mathbb{A}_{p^r}$ where p is prime.

In what follows we describe how the extraction functions coefficients $a_{i,j}$ are computed and afterwards we give some practical issues on using extracting functions.

3.1 Computing extraction functions coefficients

Let us examine the application of all extraction functions in matrix form. The evaluation off all extractions functions (i.e. unpacking of all messages) can be expressed as a matrix multiplication $M = A \cdot C$ where $M = (m_i)_{0 \le i < n}$ is the vector of messages packed in $m \left(= m_0 + m_1 \cdot X + \ldots + m_{n-1} \cdot X^{n-1}\right)$, A is the matrix (unpacking matrix) with coefficients $a_{i,j}$ such that in the row i are the coefficients of the extraction function $\texttt{Extr}_i$ and $C = \left(\phi^j(m)\right)_{0 \le j < n}$ is the vector of conjugates of the packed message m.

In order to find the coefficients of matrix A we shall study the inverse transformation, i.e. the packing of messages. The packing can be performed by multiplying the inversed matrix A to the values to pack: $C = A^{-1} \cdot M$. Only the first element of C represents the packed message, other elements are conjugates of the packed message. Let us expand the j-th conjugate of m:

$$\phi^j(m) = \phi^j\left(\sum_{0 \le i < n} m_i X^i\right) = \sum_{0 \le i < n} \phi^j(m_i)\,\phi^j\left(X^i\right) = \sum_{0 \le i < n} m_i \phi^j\left(X^i\right) \quad (3)$$

Relations $\phi^j(a+b) = \phi^j(a) + \phi^j(b)$, $a, b \in \mathbb{F}_{t^n}$, and $\phi^j(a) = a$, $a \in \mathbb{F}_t$, are used in the development of the conjugate expression. Arranging equations (3) in matrix form we obtain the following expression for matrix A^{-1}:

$$A^{-1} = \begin{pmatrix} 1 & X & X^2 & & X^{(n-1)} \\ 1 & X^t & X^{2t} & & X^{(n-1)t} \\ 1 & X^{t^2} & X^{2t^2} & & X^{(n-1)t^2} \\ & & & \ddots & \\ 1 & X^{t^{n-1}} & X^{2t^{n-1}} & & X^{(n-1)t^{n-1}} \end{pmatrix}$$

We observe that A^{-1} is the Vandermonde matrix over the conjugates of X: $A^{-1} = V\left(X, \phi(X), \phi^2(X), \ldots, \phi^{n-1}(X)\right)$. Vandermonde matrix inversion complexity is $O\left(n^2\right)$ in terms of finite field operations [EF06].

Using the explicit formula for Vandermonde matrix inversion we prove relation (4). A proof is given in appendix A.

$$\phi(a_{i,j}) = a_{i,j+1 \mod n},\ 0 \le i, j < n \quad (4)$$

Formulation $\sum_{j=0}^{n-1} \phi^j\left(a_{i,0} \cdot \texttt{ct}_{pack}\right)$ can also be used for the extraction function $\texttt{Extr}_i\left(\texttt{ct}_{pack}\right)$ in order to compute and store only the first row of matrix A. If k is the number of extraction functions to be evaluated then in this case the Frobenius endomorphism will have to be done $k \cdot (n-1)$ times instead of $(n-1)$, whilst the number of plaintext-ciphertext multiplications lowers from $k \cdot n$ to k.

To summarize, the computation of the unpacking matrix A is done in two steps: in the first step matrix A^{-1} is built[3] and afterwards this matrix is inverted (quadratic complexity). We shall note that if matrix A is to be integrated in the public key-chain we have the flexibility to store only the first column of A and afterwards compute when needed other columns using relation (4).

4 Experimentation

In this section we describe several experiments we have performed in order to obtain numeric estimates of Vandermonde matrix inversion time, coefficient extraction speed and packing performance. In our experiments we have used the HElib library [HS14,HS15]. Cyclotomic polynomials are used in HElib as irreducible modulus for plaintext/ciphertext rings. Let $\Phi_m(X)$ be the m-th cyclotomic polynomial. The degree of $\Phi_m(X)$ is given by Euler's totient function $\varphi(m)$. If polynomial $\Phi_m(X)$ can be factored modulo t (plaintext coefficient modulo), then all the factors will have the same degree n. The ciphertext coefficient size is chosen automatically as a function of the number multiplication levels L to support. The security of the obtained homomorphic encryption scheme (security of the ring-LWE instance) depends on the cyclotomic polynomial degree and on the ciphertext coefficient size. Many other parameters allow to fine tune HElib. In our experiments we limit ourselves to the following parameters: plaintext modulo – t, number of multiplication levels – L and cyclotomic polynomial order – m.

We shall note that HElib allows to evaluate linear functions over the standard power basis. These methods can be used to perform a coefficient extraction procedure, however a non-optimized one. We have implemented in HElib the new matrix inversion procedure and an optimized extraction procedure. In the case when the cyclotomic polynomial factors over the plaintext space the extraction procedure is applied onto each plaintext slot separately. So actually a single extraction procedure extracts many messages, one message per slot. Furthermore one can use existing plaintext slot rotation methods in order to move all the packed messages into the same slot.

4.1 Matrix inverse

In HElib the Vandermonde matrix is inverted using Gauss elimination over field extension (NTL library function). Afterwards Hansel's lifting is used to compute the inverse over ring extension when the plaintext space is a ring extension. This method has a cubic complexity in terms of extension field operations. In our implementation we initially compute the first column of the inversed Vandermonde matrix using the explicit formula directly over the field/ring extension. Other columns are computed using relation (4) on demand. The complexity for this method is quadratic. In what follows we compare HElib implementation with our

[3] It can be built on the fly and integrated in the matrix inversion step.

implementation for matrix inverting. Firstly we study the influence of field/ring extension size (which is the size the of Vandermonde matrix) and afterwards the influence of coefficient size (plaintext modulo size) on the inversion time. We have used cyclotomic polynomials $\Phi_m(X)$ of small[4] size prime order m and which are irreducible modulo t. Obtained matrix inversion times can be generalized to cyclotomic polynomials $\Phi_m(X)$ whose factors modulo t have equivalent degrees.

In figure 1 are plotted execution times of matrix inversion as a function of matrix size (plaintext extension size). Just as it was expected our method, which has a smaller theoretical complexity, is faster in practice too. For large matrices our method is up to 240 times faster. With the increase in matrix size this gap grows exponentially.

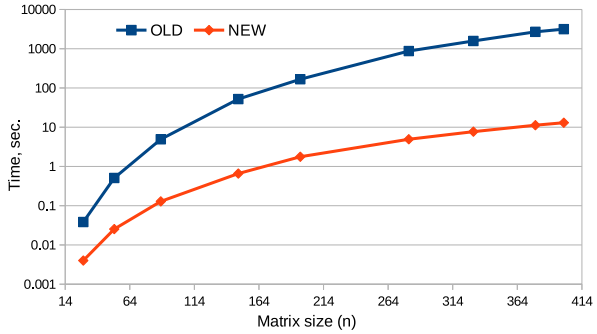

Fig. 1. Inversion time as a function of the extension field size. The same coefficient modulus is used. Field extensions are $\mathbb{Z}_{1033}/\Phi_n(X)$, $n \in [29, 53, 89, 149, 197, 281, 331, 379, 401]$. The execution times for HElib implementation (OLD label) or ours (NEW label) is on the vertical axis. Extension field size (matrix size) is on the horizontal axis.

In figure 2 is represented the execution time of HElib inversion method and our inversion as a function of the field/ring extension coefficient bit-size. We can observe that for both methods the inversion over ring extension is faster than the inversion over field extension for equivalent coefficient bit-sizes. We suppose this is due to a more optimized implementation of ring extension operations in NTL. Our inversion method is at least 40 times faster for any coefficient sizes.

4.2 Extraction performance

There are two possible formulations for the extraction function, either $\sum_{j=0}^{n-1} \phi^j (a_{i,0} \cdot \mathtt{ct}_{pack})$ or $\sum_{j=0}^{n-1} a_{i,j}\phi^j (\mathtt{ct}_{pack})$. We have implemented and have tested both expressions using HElib. Unsurprisingly the second formulations is faster in all the test cases

[4] HE scheme security was irrelevant in this experiment.

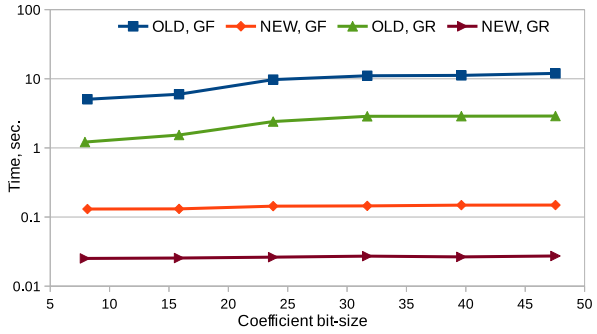

Fig. 2. Inversion time as a function of the field/ring extension coefficient bit-size. The used extensions are $\mathbb{Z}_t/\Phi_{89}(X)$, with either t prime (field extension– GF label) or a power of prime $t = 3^k$, $k \in [5, 10, 15, 20, 25, 30]$, (ring extension – GR label). $\Phi_{89}(X)$ is the 89-th cyclotomic polynomial (matrix size is 88×88). On horizontal axis the bit-size ($\log_2 t$) of plaintext modulo (t) is given. On vertical axis the execution time of the inversion method: either the HElib implementation (OLD label) or ours (NEW label).

we have performed. This is due to the fact that Frobenius endomorphisms need a key-switching operation and is approximatively 10 times slower than plaintext-ciphertext multiplications. In what follows the second formulation is used.

Extraction methods were implemented in HElib. The used plaintext coefficient modulo t is either 2 or 1024. The order m of cyclotomic polynomials $\Phi_m(X)$ is chosen in such a way that m is a prime inferior to 10^5, $\Phi_m(X)$ factors modulo t into factors of degree $n < 200$ and the obtained HE scheme has at least 80 bits of security. The number of multiplicative levels L to support by the HE scheme is set to 5. In total 12 instances of HE schemes are obtained.

Suppose $\mathtt{ct}_{pack}$ is a ciphertext which packs $\varphi(m)$ messages $m_{i,j}$, $0 \leq i < \varphi(m)/n$ and $0 \leq j < n$. Let $\sum_{j=0}^{n-1} m_{i,j} X^j$ be the polynomial packed into the i-th plaintext slot of ciphertext $\mathtt{ct}_{pack}$. The extraction procedure is performed in two steps. Initially n ciphertexts $\mathtt{ct}_j$ are obtained using the extraction function (2). Ciphertext $\mathtt{ct}_j$ packs messages $m_{i,j}$ in its plaintext slots. Afterwards plaintext slot rotations were performed on each ciphertext $\mathtt{ct}_j$ in order to bring all messages $m_{i,j}$ into the same slot. In total $\varphi(m) - n$ rotations are done, n messages being already in the good slot. As a result $\varphi(m)$ ciphertexts $\mathtt{ct}_{i,j}$ are obtained each containing in the same slot a message $m_{i,j}$ from the initial packed ciphertext $\mathtt{ct}_{pack}$.

The application of extraction function increases the noise level of input ciphertext. This must be considered when the dimensioning of the HE scheme is done. Ciphertext noise increase (in bits) incurred by the extraction procedure is represented in figure 3. It includes the noise due to the extraction func-

tion (2) and to plaintext slot rotation[5]. For both plaintext spaces unexplained jumps in the noise increase can be observed for the same cyclotomic polynomials ($m = 5419, 6529,$ etc.). Without entering into much details we found that this is caused by the cyclotomic ring representation in HElib. Plaintext slot rotation increases ciphertext noise only for these cyclotomic rings and no noise at all is added for the others. Extraction procedure noise increase for plaintext modulo 2 is around $6-7$ bits when slot rotation does not add any noise and around $11-12$ bits otherwise. Noise increase is a bit larger for plaintext modulo 1024. Reported to the HE instances we use ($L = 5$) this corresponds up to 1 multiplicative level for $t = 2$ and up to 2 multiplicative levels for $t = 1024$.

In order to see if the noise increase depends on the number of multiplicative levels the same tests were performed for multiplicative level $L = 10$. The difference between noise increase in both cases ($L = 5$ and $L = 10$) was under 0.03% for $t = 2$ and even smaller for $t = 10$. We assume that the noise increase does not vary much as a function of multiplicative depth.

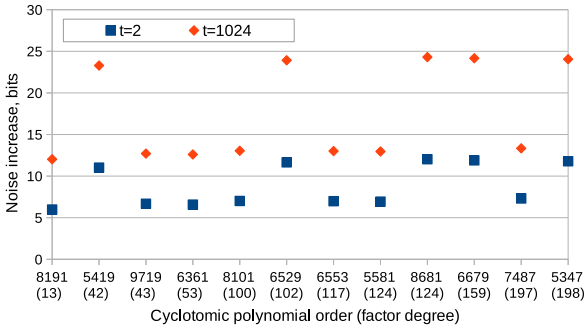

Fig. 3. Increase of ciphertext noise in bits (vertical axis) after unpacking procedure related to cyclotomic polynomial order m (horizontal axis). Horizontal axis values are sorted in increasing order of the degree of cyclotomic polynomial factors (given in brackets).

Execution time of the unpacking procedure is represented in figure 4. Total execution time of unpacking procedure is plotted with continuous line, whilst dotted line is used for the fraction of the total execution time took by the extraction function. Extraction function execution time (dotted line) depends greatly on the cyclotomic polynomial factor degree and less on the cyclotomic polynomial degree. The unusual jumps of the total execution time are explained as previously by the properties of the employed cyclotomic ring. For some cyclo-

[5] Plaintext slot rotations in HElib are performed using one or more ciphertext homomorphisms followed by a key-switchings. We suppose that any rotation can be performed in a single step so that the noise due to slot rotation remains low and bounded.

tomic rings plaintext slot rotation is faster. Execution time is always larger for plaintext modulo $t = 1024$ when compared to $t = 2$.

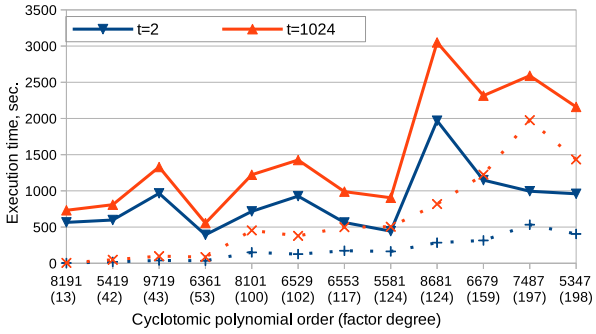

Fig. 4. Execution time (vertical axis) of the unpacking procedure related to cyclotomic polynomial order m (horizontal axis). Horizontal axis values are sorted in increasing order of the degree of cyclotomic polynomial factors (in brackets).

4.3 Comparison to literature methods

In work [CCF+15] the authors propose a method in which messages are initially encrypted using a stream cipher (low-depth ones: Trivium and Kreyvium) and then a trans-ciphering procedure is used to switch from stream cipher encryption to HE scheme encryption. The authors used HElib and an in-house implementation of HE scheme for their experimentations. Only the results obtained using HElib are reported here. After trans-ciphering the HE parameters allow to execute 7 more multiplicative levels of homomorphic operations. The best latency and throughput results they obtain are provided in table 1. The latency is the homomorphic execution time of a stream cipher and the throughput is the number of messages which are obtained during an execution. The throughput results are little bit "inflated" because the messages from the plaintext slots of a ciphertext are also counted (no ciphertext rotation is performed).

Stream cipher	latency (sec.)	throughput (bit/min.)
Trivium-13	11380	516.3
Kreyvium-13	12450	286.8

Table 1. Best results in terms of latency and throughput from [CCF+15].

We have executed the same tests as in previous section except that the plaintext modulo was binary and the multiplicative level was set to 8. After the

extraction procedure the HE scheme supports 7 more multiplicative levels (approximatively) as in [CCF+15]. We keep only HE instances which provide more than $\lambda = 80$ bits of security. Two use cases are envisaged: (i) only the extraction function equation (2) is applied (n ciphertext are obtained, one message per plaintext slot) and (ii) additionally plaintext slot rotations are performed ($\varphi(m)$ ciphertexts are obtained, one message per ciphertext). In table 2 are shown the obtained latencies (execution times) and throughputs for different cyclotomic polynomials. First and second use cases results are given in columns "extr." and respectively in columns "extr.+rot.". Column "#slots" gives the available number of plaintext slots ($= \varphi(m)/n$) and column "|slot|" represents the cyclotomic polynomial factor degree n.

m	#slots	|slot|	λ	latency (sec.)		throughput (bit/min.)		expansion
				extr.	extr.+rot.	extr.	extr.+rot.	factor
6529	64	102	> 80	167.6	1460.1	2336.3	268.3	339.4
6553	56	117		225.3	847.9	1744.7	463.6	341.7
6679	42	159		434.6	1861.0	922.0	215.3	339.8
7487	38	197		749.2	1564.0	599.6	287.2	340.7
8101	81	100	> 128	202.4	1132.5	2400.8	429.1	337.3
8191	630	13		4.9	991.5	99366.8	495.6	339.6
8681	70	124		393.0	3169.6	1325.1	164.3	335.1
9719	226	43		53.8	1582.2	10833.9	368.5	339.7

Table 2. Latency and throughput results obtained using our extraction procedure.

The expansion factor (ratio between the size of sent data and the number of sent messages) obtained in [CCF+15] is asymptotically close to 1 for a large number of messages. The expansion factor in our case (given in the last column) is approximatively 340 ciphertext bits for a message bit. It does not depend on the number of messages but depends on the HE scheme parameters.

Trans-ciphering [CCF+15] is performed on each plaintext slot independently. Resulting ciphertexts contain many input messages (first use case described in previous section). In order to be fair we compare their results to the results we obtain using the first use case only ("extr." columns). The latency in our case is at least 15 times lower. Obtained throughput is at least as good as in their work and for some HE instances ($m = 8191$) much larger. Sure that this comparison is only approximate as we have ignored the machines on which the test were performed, the fact that "unpacked" ciphertext are of different sizes: 8 levels in our case compared to 21 in theirs[6], and also the two methods have different expansion factors.

[6] Subsequent homomorphic evaluations on "unpacked" ciphertext will be faster in our case.

5 Conclusions

In this work we have proposed a packing procedure for homomorphic encryption schemes. The messages are packed into coefficients of plaintext polynomials. A method for homomorphically unpacking messages packed in this way from a ciphertext is proposed. Performed experimentations showed that lower unpacking times are obtained when compared to existing work based on trans-ciphering.

The packing ratio (number of bits to send for a payload bit) is larger than for trans-ciphering methods (which is worse). In these methods (trans-ciphering) only the uplink traffic can be packed. Whereas the method proposed in this paper can be used for packing uplink as well as for downlink traffic. In order to increase the packing ratio of this packing method the digit-extraction (from plaintext spaces modulo p^r) procedure described in [HS15] can also be used.

References

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully Homomorphic Encryption Without Bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, 2012.

[Bra12] Zvika Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In *Advances in Cryptology - Crypto 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012.

[CCF+15] Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. How to Compress Homomorphic Ciphertexts. Cryptology ePrint Archive, Report 2015/113, 2015.

[CCK+] JungHee Cheon, Jean-Sébastien Coron, Jinsu Kim, MoonSung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch Fully Homomorphic Encryption over the Integers. In *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 315–335.

[CDS15] Sergiu Carpov, Paul Dubrulle, and Renaud Sirdey. Armadillo: A Compilation Chain for Privacy Preserving Applications. In *SCC@ASIACCS*, pages 13–19, 2015.

[EF06] A. Eisinberg and G. Fedele. On the inversion of the Vandermonde matrix. *Applied Mathematics and Computation*, 174(2):1384–1397, 2006.

[Gen09] Craig Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, 2009.

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully Homomorphic Encryption with Polylog Overhead. In *Proceedings of the 31st Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'12, pages 465–482, Berlin, Heidelberg, 2012. Springer-Verlag.

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In *In CRYPTO*, 2012.

[GLN12] Thore Graepel, Kristin Lauter, and Michael Naehrig. ML Confidential: Machine Learning on Encrypted Data. In *International Conference on Information Security and Cryptology – ICISC 2012, Lecture Notes in Computer Science, to appear*, December 2012.

[HS14] Shai Halevi and Victor Shoup. Algorithms in HElib. In *CRYPTO*, volume 8616 of *Lecture Notes in Computer Science*, pages 554–571, 2014.

[HS15] Shai Halevi and Victor Shoup. Bootstrapping for HElib. In *EUROCRYPT*, volume 9056 of *Lecture Notes in Computer Science*, pages 641–670. Springer, 2015.

[LLAN] Kristin Lauter, Adriana López-Alt, and Michael Naehrig. Private Computation on Encrypted Genomic Data. In *Progress in Cryptology - LATINCRYPT 2014*, volume 8895 of *Lecture Notes in Computer Science*, pages 3–27.

[NLV11] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can Homomorphic Encryption Be Practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, CCSW '11, pages 113–124, 2011.

A Vandermonde matrix over $\mathbb{F}_{t^n}$ inverse

In what follows we give a proof for relation $\phi(a_{i,j}) = a_{i,j+1 \mod n}$ verified by the inverse of the Vandermonde matrix $V\left(e, \phi(e), \ldots, \phi^{n-1}(e)\right)$ over finite field $\mathbb{F}_{t^n}$ where ϕ is the Frobenius endomorphism $X \mapsto X^t$.

Lemma 1. *Let $V\left(e, \phi(e), \ldots, \phi^{n-1}(e)\right)$ be a Vandermonde matrix with e an element from an extension field $\mathbb{F}_{t^n}$ and $A = (a_{i,j})_{n\times n}$ be the inverse of this matrix. The elements $a_{i,j}$ verify relation $a_{i,j+1 \mod n} = \phi(a_{i,j})$.*

Proof. The explicit formula for Vandermonde matrix $V(x_0, x_1, \ldots, x_{n-1})$ inversion is:

$$a_{ij} = {}^{\alpha_{ij}}/_{\beta_j},\ 0 \leq i, j < n$$

where

$$\alpha_{ij} = (-1)^{n-i-1} \sum_{\sigma \in C_{S_j}^{n-i-1}} \prod_{k\in\sigma} x_k,$$

$$\beta_j = \prod_{m\in S_j} (x_j - x_m).$$

Here S_j is the set of integers from 0 to $n-1$ excluding j, i.e. $S_j = \{0, \ldots, n-1\} \setminus \{j\}$, and $C_{S_j}^k$ is the set of combinations of k elements from the set S_j.

If each element k in the set S_j is replaced by the element $k+1 \mod n$ then the set $S_{j+1 \mod n}$ is obtained:

$$S_j \overset{k\to k+1 \mod n}{\Longrightarrow} S_{j+1 \mod n} \tag{5}$$

If each element k in every combination σ of $C_{S_j}^p$ is replaced by $k+1 \mod n$ then the set of combinations $C_{S_{j+1 \mod n}}^p$ is obtained:

$$C_{S_j}^p \overset{k\to k+1 \mod n}{\Longrightarrow} C_{S_{j+1 \mod n}}^p \tag{6}$$

For example let $C_{S_1}^2 = \{\{0,2\},\{0,3\},\{2,3\}\}$ be 2-element combinations from the set $S_1 = \{0,2,3\}$. When in $C_{S_1}^2$ every element k is replaced by $k+1 \bmod n$ we obtain $\{\{1,3\},\{1,0\},\{3,0\}\} = C_{S_2}^2$.

In our case we have $x_k = \phi^k(e)$, $0 \leq k < n$. The Frobenius endomorphism of x_k is $\phi(x_k) = \phi\left(\phi^k(e)\right) = \phi^{k+1}(e) = x_{(k+1) \bmod n}$, the " $\bmod n$" comes from the fact that $\phi^n(e) = e$. Lets compute the Frobenius endomorphism of α_{ij} :

$$\phi(\alpha_{ij}) = \phi\left((-1)^{n-i-1} \sum_{\sigma \in C_{S_j}^{n-i-1}} \prod_{k \in \sigma} x_k\right) =$$

$$(-1)^{n-i-1} \sum_{\sigma \in C_{S_j}^{n-i-1}} \prod_{k \in \sigma} \phi(x_k) =$$

$$(-1)^{n-i-1} \sum_{\sigma \in C_{S_j}^{n-i-1}} \prod_{k \in \sigma} x_{(k+1) \bmod n} \overset{(6)}{=}$$

$$(-1)^{n-i-1} \sum_{\sigma \in C_{S_{j+1} \bmod n}^{n-i-1}} \prod_{k' \in \sigma} x_{k'} = \alpha_{i,j+1 \bmod n}$$

Equivalently for β_j we obtain:

$$\phi(\beta_j) = \phi\left(\prod_{k \in S_j} (x_j - x_k)\right) = \prod_{k \in S_j} \phi(x_j - x_k) = \prod_{k \in S_j} (\phi(x_j) - \phi(x_k)) =$$

$$\prod_{k \in S_j} (x_{j+1 \bmod n} - x_{k+1 \bmod n}) \overset{(5)}{=} \prod_{k \in S_{j+1 \bmod n}} (x_{j+1 \bmod n} - x_k) = \beta_{j+1 \bmod n}$$

We conclude that $\phi(a_{i,j}) = \phi\left(\alpha_{i,j}/\beta_j\right) = {}^{\alpha_{i,j+1 \bmod n}}/_{\beta_{j+1 \bmod n}} = a_{i,j+1 \bmod n}$ (β_j is always invertible because $\beta_j \neq 0$).

Corollary 1. *Let $V\left(e, \phi(e), \ldots, \phi^{n-1}(e)\right)$ be a Vandermonde matrix with e an element from an extension ring $GR(t^n, m)$ and $A = (a_{i,j})_{n \times n}$ be the inverse of this matrix. The elements $a_{i,j}$ verify relation $a_{i,j+1 \bmod n} = \phi(a_{i,j})$ where ϕ is the Frobenius endomorphism $X \mapsto X^t$.*

Proof. Equivalently as lemma 1.