Efficient Evaluation of Low Degree Multivariate Polynomials in Ring-LWE Homomorphic Encryption Schemes

Sergiu Carpov and Oana Stan

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

Abstract. Homomorphic encryption schemes allow to perform computations over encrypted data. In schemes based on RLWE assumption the plaintext data is a ring polynomial. In many use cases of homomorphic encryption only the degree-0 coefficient of this polynomial is used to encrypt data. In this context any computation on encrypted data can be performed. It is trickier to perform generic computations when more than one coefficient per ciphertext is used.
In this paper we introduce a method to efficiently evaluate low-degree multivariate polynomials over encrypted data. The main idea is to encode several messages in the coefficients of a plaintext space polynomial. Using ring homomorphism operations and multiplications between ciphertexts, we compute multivariate monomials up to a given degree. Afterwards, using ciphertext additions we evaluate the input multivariate polynomial. We perform extensive experimentations of the proposed evaluation method. As example, evaluating an *arbitrary* multivariate degree-3 polynomial with 100 variables over Boolean space takes under 13 seconds.

1 Introduction

The widespread of cloud storage and computing services has conducted to a massive shift towards data and computation outsourcing both for particular users and businesses. Between the incontestable advantages of using cloud computing, one can cite the cost reduction in the development and in the maintenance of data centers, the scalability and elasticity of cloud resources, improved accessibility (via only an Internet connection) and a guaranteed reliability (with redundancy and back-up mechanisms).

However, one of the main barriers in the large-scale adoption of cloud based services for data processing and storage concerns the data privacy, since the data owners have little or no control on the cloud provider security policies and practices. One immediate and recommended solution is to encrypt the highly sensitive data before sending and migrating it to a cloud environment. In this context one can use Homomorphic Encryption (HE) schemes, which allow to perform computations directly over encrypted data.

Homomorphic encryption schemes have been known for a long time, with the first partial constructions dating back to the seventies [23]. In his seminal work

[13], Gentry proposed the first Fully Homomorphic Encryption (FHE) scheme capable to evaluate an arbitrary number of additions and multiplications over encrypted data (allowing, in theory, to execute any computation). From this major theoretical breakthrough, there was an increased research interest for homomorphic encryption. Many fully and somewhat homomorphic encryption schemes (SHE) have been proposed in literature [5, 12, 26, 20, 10] based on different security assumptions. Compared to FHE schemes, SHE schemes allow to perform only a limited number of homomorphic operations and are used as a basis to construct FHE schemes. In one of the latest survey on homomorphic encryption [1], FHE schemes are classified into four main families:

- Ideal Lattice based [13]
- Schemes over integers [26]
- Schemes based on the Learning With Error (LWE) or the ring version (RLWE) problem [6, 5]
- NTRU-like schemes [20]

Using both addition and multiplication over ciphertexts, one can execute arbitrary arithmetic circuits, evaluate multivariate polynomials, etc. A typical use of FHE is to express the function to be computed on encrypted data as a static control-flow program and execute homomorphically the associated Boolean circuit [8]. Despite their recent and successive improvements, the main issue about FHE schemes is the performance (in terms of execution time and memory requirements) and, consequently, the practical applicability.

Contribution

In this work, we focus on the third category of schemes, the ones based on the RLWE assumption, and we propose a new method for the efficient evaluation of multivariate polynomials in the homomorphic domain. For this type of schemes the plaintext and, respectively, the ciphertext space are polynomial ring elements with coefficients of different size (i.e. being defined over different integer modulus). In many applications using RLWE schemes only the zero degree coefficient is used to encode useful data.

Several researches were conducted in order to improve the evaluation performance of polynomials over encrypted data and take advantage of the plaintext space polynomial structure. The coefficient packing method, introduced in [22], allows to pack several messages into a single ciphertext. In a series of papers [28, 27] the authors describe how to evaluate multivariate linear polynomials over coefficient packed messages. In this work, we further generalize their method to allow evaluation of low-degree multivariate polynomials. The coefficients of the evaluated multivariate polynomial can be either in clear or encrypted forms. The proposed packing and computation methods allow not only to reduce ciphertext expansion ratio[1] but also to perform computations using messages encoded in the same ciphertext. As shown later, our method reduces the complexity of basic

[1] The ratio between ciphertext and plaintext message sizes.

operations performed in homomorphic space, and thus ameliorates the performances for different types of computations manipulating private data. As an example of applications in which using our polynomial evaluation method can be useful are the machine learning algorithms.

Related works

Several research works found in the literature propose solutions on how to ameliorate the efficiency of homomorphic encryption schemes in the context of practical applications.

In [22] it is introduced the coefficient packing technique for homomorphic ciphertexts. Besides decreasing ciphertext size this method allows to accelerate multi-bit additions/multiplications. Roughly speaking, the main idea is to encode a bit-wise representation of integers into plaintext polynomial coefficients. Under certain conditions, adding and multiplying such encrypted plaintexts allows to perform binary addition and respectively binary multiplication of initial integers. As such, this method is appropriate in computations using a small number of multiplications (i.e. computing standard deviation). The authors used this technique to efficiently compute means, variances and inner products (i.e. degree-1 polynomials). The inner product is used in a protocol for making logistic model predictions over homomorphically encrypted data.

In [28] the authors propose an extension of the data packing technique introduced in [22] and use it to homomorphically compute inner products and Hamming distances. Hamming distance computation is equivalent to evaluating a particular degree-2 polynomial or a degree-1 polynomial with encrypted coefficients. As explained earlier, our work is an extension of the approach from [28] for general polynomials of degree larger than one.

In [9] the authors introduce an "unpacking" technique for coefficient packed ciphertexts, thus they describe how to obtain several ciphertexts from a single one in the encrypted domain. No computation methods over packed ciphertext is proposed.

One of the first approaches aiming to efficiently encode the messages for HE schemes is the packing method proposed by Smart and Vercauteren in [24, 14]. By using CRT (Chinese Remainder Theorem) on polynomials, one can perform SIMD (Single Instructions Multiple Data) operations on encrypted data. Roughly speaking, the plaintext space is split into several independent "slots" if the cyclotomic polynomial defining the polynomial ring can be factored. The multivariate polynomial evaluation method we introduce is complementary to the batching and can be applied on top of it (i.e. do a polynomial evaluation in each slot).

For a view on the different applications of these optimization methods for RLWE-based schemes, we refer to paper [15] where the authors discuss which machine learning algorithms can be expressed in polynomial form. As said earlier only polynomials can be evaluated using homomorphic encryption. Several homomorphic implementations of classification algorithms are proposed in [3].

In particular, the authors describe how to perform hyperplane decision (linear classifier), Naive Bayes and decision trees classification algorithms on HE encrypted data. In a series of papers [29, 27] the authors discuss different applications (pattern matching and biometric authentication) of secure inner-product computation. Different encoding methods for representing fixed-point numbers, designed for Ring-based SHE schemes, were presented in [2] and [11] along with their applications to a homomorphic forecasting algorithm and respectively to an image processing algorithm.

This paper is organized as follows. After a brief introduction of generic operations supported by a RLWE based homomorphic scheme, we describe the proposed evaluation method for multivariate polynomials in Section 3. Later on we provide some experimental results in Section 4 followed by an example of a practical application of our method. Finally, Section 5 concludes the paper and provides some perspectives on the future work.

2 Homomorphic encryption

2.1 Preliminiaries

Let us first give the basic notation and introduce the RLWE problem. Let $\mathbb{A} = \mathbb{Z}[X]/f(x)$ be the ring of polynomials modulo a monic irreducible polynomial $f(x)$. Usually, one would typically restrict $f(x)$ to be the cyclotomic polynomial $\Phi_m(X)$, i.e. the minimal polynomial of the primitive m-th root of unity. Let $\mathbb{A}_q = \mathbb{A} \mod q$ be the set of polynomials modulo $\Phi_m(X)$ with coefficients modulo q. Thus an element in $\mathbb{A}$ is a polynomial of degree d over $\mathbb{Z}_q$, with $d = \varphi(m)$ (Euler's totient function) in the case of a cyclotomic polynomial modulus.

Using these above notations, we recall a simple definition of the RLWE problem, first introduced by Lyubashevsky, Peikert and Regev [21].

RLWE problem. For security parameter λ, $f(x)$ is the cyclotomic polynomial depending on λ, the ring $\mathbb{A}$ is defined as before and q is an integer. Let $\chi(\lambda)$ be an error distribution over $\mathbb{A}$. The decision-RLWE problem is to distinguish between two distributions: a first distribution obtained by sampling (a_i, b_i) uniformly from $\mathbb{A}_q^2$ and a second distribution made of a polynomial number of samples of the form $(a_i, b_i = a_i * s + e_i) \in \mathbb{A}_q^2$, where a_i is uniformly random in $\mathbb{A}_q$, $e_i \leftarrow \chi$ and $s \leftarrow \mathbb{A}_q$ is a uniformly random element. The $\mathrm{RLWE}_{d,q,\lambda}$ assumption is that the RLWE problem is infeasible.

This problem can be reduced using a quantum algorithm to the shortest vector problem over ideal lattices and its harness is independent of q (usually either prime or power of 2). The above RLWE problem easily leads to several homomorphic encryption schemes, such as BGV [5] or FV [12].

In RLWE based homomorphic encryption scheme ciphertexts and secret keys are elements from the ring $\mathbb{A}_q$. The plaintext space is the ring of polynomials $\mathbb{A}_t$ $(t \ll q)$.

Leveled SHE schemes [5] use a series of integer modulus $q_0, q_1, \ldots$ for ciphertexts at different moments of an 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 [4] a notion of scale-invariance for leveled SHE schemes is introduced. In scale-invariant schemes a single modulus, q, is used for ciphertexts during the whole homomorphic evaluation.

Usually, the plaintext space is chosen for $t = 2$ and a single binary message is encrypted per ciphertext, in the zero-degree coefficient of the polynomial from $\mathbb{A}_2$, allowing to homomorphically evaluate arbitrary Boolean circuits. By using a larger modulus ($t > 2$) for the plaintext space it is possible to execute operations on integers modulo t homomorphically or even on elements from the polynomial ring $\mathbb{A}_t$ (see next section).

In the so-called batched schemes [14], the plaintext space ring $\mathbb{A}_t$ can be factored into sub-rings (defined by the factorization of the polynomial $\Phi_m(X)$ modulo t) such that homomorphic operations apply to each sub-ring independently. Batching several messages into ciphertext slots allows to execute homomorphic operations on all messages in parallel at the same time.

To summarize, RLWE based HE schemes allow to execute homomorphic operations (batched or not) over polynomial ring $\mathbb{A}_t$ elements (includes the integer modulo ring and finite field cases). In the next section, we will present more formally the basic operations a SHE can execute.

2.2 Homomorphic operations

Beside the typical key generation, encryption and decryption, a homomorphic encryption scheme is also defined by a set of plaintext operations which can execute in the encrypted domain. Below, we give a generic list of operations supported by any public-key RLWE SHE scheme ignoring implementation details. We limit our description to the non-batched schemes, the results presented in this paper being also valid for the batched ones.

$\mathtt{KeyGen}\left(1^\lambda\right)$ – 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 multiplications and ring homomorphism operations).

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

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

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

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

$\mathtt{Hom}(\mathtt{ct}, k, evk)$ – outputs a ciphertext which represents the application of the ring homomorphism $X \mapsto X^k$ over the plaintext message polynomial encrypted by $\mathtt{ct}$:
$\mathtt{Dec}_{sk}(\mathtt{Hom}(\mathtt{ct}, k, evk)) \equiv p\left(X \mapsto X^k\right)$ where $p(X) = \mathtt{Dec}_{sk}(\mathtt{ct})$
For some homomorphic encryption scheme instantiations, this homomorphism operation can be performed with only a small noise increase.

For simplicity sake, we use addition and multiplication operators for homomorphic addition and multiplication of ciphertexts: $\mathtt{ct}_1 + \mathtt{ct}_2 = \mathtt{Add}\left(\mathtt{ct}_1, \mathtt{ct}_2\right)$ and respectively $\mathtt{ct}_1 \cdot \mathtt{ct}_2 = \mathtt{Mult}\left(\mathtt{ct}_1, \mathtt{ct}_2, evk\right)$. The ring homomorphism operation $\mathtt{Hom}\left(\mathtt{ct}, k, evk\right)$ is denoted using $\phi^k\left(\mathtt{ct}\right)$. 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, e.g. $\mathtt{ct}_1 + m_2$ means an addition between the ciphertext $\mathtt{ct}_1$ 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.

3 Homomorphic evaluation of multivariate polynomials

Let $\mathcal{P}^n$ be the space of all the polynomials with n variables, $x_0, \ldots, x_{n-1}$. Without loss of generality we suppose that the constant term is zero. The subspace of polynomials of maximal degree d, $1 \leq d \leq n$, is denoted by $\mathcal{P}_d^n$ and composed by polynomials defined as:

$$P\left(x_0, \ldots, x_{n-1}\right) = \sum_{1 \leq k \leq d} \sum_{0 \leq e_1 \leq \ldots \leq e_k < n} c_{e_1,\ldots,e_k} \cdot x_{e_1} \cdot \ldots \cdot x_{e_k} \tag{1}$$

In this formulation the monomial terms $x_{e_1} \cdot \ldots \cdot x_{e_k}$ are grouped by their degree k. The inner sum adds up all the combinations with repetition of k variables. Variables $x_0, \ldots, x_{n-1}$ and coefficients $c_{e_1,\ldots,e_k}$ belong to a ring. In this work the plaintext space for homomorphic encryption is used, namely the ring of integers modulo t.

In what follows we describe some naive methods for multivariate polynomial evaluation over homomorphically encrypted data and afterwards we introduce an optimized method for multivariate polynomial evaluation.

Let $P\left(x_0, \ldots, x_{n-1}\right)$ be a polynomial from $\mathcal{P}_d^n$ which has to be evaluated at an encrypted point $a_0, \ldots, a_{n-1}$. The polynomial is evaluated over the integer ring $\mathbb{Z}_t$, with $t \geq 2$. In this work we focus on efficient evaluation of polynomials over homomorphic domain using the lowest possible parameters for the configuration of the HE scheme.

3.1 Naive methods

A straightforward method is to encrypt each value $a_0, \ldots, a_{n-1}$ into separate homomorphic ciphertexts. Using homomorphic multiplications/additions one can compute polynomial representation (1). As mentioned in Section 2, the parameters of HE schemes depend mainly on the degree of monomials and less on their number (noise increase due to homomorphic additions is much smaller than for homomorphic multiplications). The lowest HE scheme parameters are obtained when a tree-like structure is used to compute the monomials of P. Homomorphic polynomial evaluation time mainly depends on the number of homomorphic multiplications. Note that $P\left(x_0, \ldots, x_{n-1}\right)$ can have as many as $\frac{(n+d)!}{n! \cdot d!}$ monomials[2]. So, in the general case, the number of homomorphic multiplications is not polynomial, but potentially factorial.

As different monomials share common parts we can minimize the number of homomorphic multiplications by evaluating them once and reusing them when needed. Finding the optimal way to do so is a difficult optimization problem and has been studied from multiple standpoints: common subexpression elimination, arithmetic circuit optimization, etc. [7, 18, 19]. Larger HE scheme parameters should be used for the aforementioned methods as the multiplicative depth of computation increases when compared to direct homomorphic computation of polynomial terms. In return, the number of homomorphic operations needed for polynomial evaluation is lower.

Let us now present our method to be applied when the degree of the polynomials to be evaluated homomorphically respects certain condition with regard to the ciphertext space.

3.2 Optimized method for $n^d \leq \deg\left(\Phi\left(X\right)\right)$

In the naive methods presented earlier only a single coefficient (the degree zero one) of the polynomial ($\in \mathbb{A}_t$) to be encrypted in a homomorphic ciphertext is used. Other coefficients are set to zero and are not used. A better solution will be to use more than one coefficient of the polynomial. In this section we introduce an optimized method for polynomial evaluation in which the values $a_0, \ldots, a_{n-1}$ are packed in the coefficients of a homomorphic plaintext polynomial (2). The polynomial is further encrypted into the ciphertext $\texttt{ct}$. This packing technique was introduced by the authors of [22]. The proposed evaluation method is restricted to cases where relation $n^d \leq \deg\left(\Phi\left(X\right)\right)$ is verified (we explain later why).

$$Q\left(X\right) = \sum_{0 \leq i < n} a_i \cdot X^i \qquad (2)$$
$$\texttt{ct} = \texttt{Enc}_{pk}\left(Q\left(X\right)\right)$$

First, we describe the proposed polynomial evaluation method applied on plain data. Afterwards we explain how to perform this evaluation over homomorphic encrypted data, i.e. having ciphertext $\texttt{ct}$ as input.

[2] The number of combinations with repetitions for k degree monomials is $\frac{(n+k-1)!}{(n-1)! \cdot k!}$. Polynomial P has monomials of degree up to d (inclusive). By summing up the number of degree-k monomials one can obtain the expression $\frac{(n+d)!}{n! \cdot d!}$.

Polynomial $P\left(x_0, \ldots, x_{n-1}\right)$ evaluation. Let $R^{(k)}\left(X\right)$ be a polynomial defined as follows:

$$R^{(k)}\left(X\right) = Q\left(X\right) \cdot \ldots \cdot Q\left(X^{n^{k-1}}\right),\ k \geq 1 \tag{3}$$

Polynomial $R^{(1)}\left(X\right)$ has n non-zero coefficients and $R^{(1)}\left(X\right) \equiv Q\left(X\right)$. Polynomial $R^{(2)}\left(X\right)$ has n^2 non-zero coefficients and is given by expression:

$$\begin{aligned} R^{(2)}\left(X\right) &= Q\left(X\right) \cdot Q\left(X^n\right) \\ &= \left(\sum_{0 \leq i < n} a_i \cdot X^i\right) \cdot \left(\sum_{0 \leq j < n} a_j \cdot X^{n \cdot j}\right) \\ &= \sum_{0 \leq i,j < n} a_i \cdot a_j \cdot X^{i + n \cdot j} \end{aligned}$$

Namely, $R^{(2)}\left(X\right)$ coefficients are evaluations of degree-2 monomials $x_i \cdot x_j$ with $0 \leq i, j < n$, at point $a_0, \ldots, a_{n-1}$. Equivalently we can see that polynomial $R^{(k)}\left(X\right)$ has n^k non-zero coefficients which are evaluations of degree-k monomials:

$$R^{(k)}\left(X\right) = \sum_{0 \leq e_1, \ldots, e_k < n} a_{e_1} \cdot \ldots \cdot a_{e_k} \cdot X^{\sum_{1 \leq i \leq k} e_i \cdot n^{k-i}} \tag{4}$$

The l-th degree coefficient of $R^{(k)}\left(X\right)$ is $a_{e_1} \cdot \ldots \cdot a_{e_k}$ where $\left(e_1, \ldots, e_k\right)$ is the base-n decomposition of l. The polynomial $R^{(k)}\left(X\right)$ contains the products of all k-element permutations with repetition from the set $\{a_0, \ldots, a_{n-1}\}$. As the multiplication is a commutative operation, the same monomial is found several times in different coefficients of $R^{(k)}\left(X\right)$. The number of "useless" coefficients (coefficients representing the same monomial) is equal to $(n^k - \frac{(n+k-1)!}{k!(n-1)!})$, i.e. the difference between the number of $R^{(k)}\left(X\right)$ coefficients (permutations with repetitions) and the number of possible monomials of degree k (combinations with repetitions).

When $R^{(k)}\left(X\right)$ computation is performed in the ring $\mathbb{A}$ relation (5) must be verified, otherwise monomials will mix (due to modular reduction by $\Phi\left(X\right)$). So, instead of a single monomial per $R^{(k)}\left(X\right)$ coefficient we will obtain a sum of monomials.

$$n^d \leq \deg\left(\Phi\left(X\right)\right) \tag{5}$$

Having a way to compute monomial values up to degree d, lets now focus on how to multiply them by the corresponding coefficients of polynomial $P\left(x_0, \ldots, x_{n-1}\right)$ and compute the inner sum from relation (1).

Let $C^{(k)}\left(X\right)$ be a polynomial which packs degree-k monomial coefficients $c_{e_1, \ldots, e_k}$ of polynomial $P\left(x_0, \ldots, x_{n-1}\right)$:

$$C^{(k)}\left(X\right) = \sum_{0 \leq e_1 \leq \ldots \leq e_k < n} c_{e_1, \ldots, e_k} \cdot X^{N - \sum_{1 \leq i \leq k} e_i \cdot n^{k-i}} \tag{6}$$

where $N = \deg(\Phi(X))$. When polynomials $C^{(k)}(X)$ and $R^{(k)}(X)$ are multiplied together, the N-th degree coefficient of the resulting polynomial[3] is exactly the inner sum of equation (1). The product $C^{(k)}(X) \cdot R^{(k)}(X)$ is thus equal to:

$$\left(X^N \cdot \sum_{0 \le e_1 \le \ldots \le e_k < n} c_{e_1,\ldots,e_k} \cdot a_{e_1} \cdot \ldots \cdot a_{e_k} \right) + \ldots \qquad (7)$$

Other coefficients are also sum of monomial evaluations except that they are multiplied by "wrong" coefficients of multivariate polynomial $P(x_0, \ldots, x_{n-1})$.

Summing up these polynomial products for $k = 1, \ldots, d$ we obtain, in the highest degree coefficient, the evaluation of polynomial $P(x_0, \ldots, x_{n-1})$ at point $a_0, \ldots, a_{n-1}$:

$$\sum_{1 \le k \le d} C^{(k)}(X) \cdot R^{(k)}(X) =$$

$$\left(X^N \cdot \underbrace{\sum_{1 \le k \le d} \sum_{0 \le e_1 \le \ldots \le e_k < n} c_{e_1,\ldots,e_k} \cdot a_{e_1} \cdot \ldots \cdot a_{e_k}}_{P(a_0,\ldots,a_{n-1})} \right) + \ldots \qquad (8)$$

Polynomial $P(x_0, \ldots, x_{n-1})$ evaluation over binary plaintext space. Formulation of $\mathcal{P}_d^n$ polynomials is simpler when evaluated over the binary ring $\mathbb{Z}_2$ because, for any $a \in \mathbb{Z}_2$ and $p \ge 1$, we have $a^p \equiv a$. Any monomial $x_{e_1}^{l_0} \cdot \ldots \cdot x_{e_k}^{l_k}$ of degree $(l_0 + \ldots + l_k)$ is equivalent to monomial $x_{e_1} \cdot \ldots \cdot x_{e_k}$ of degree k. It is easy to see in this case, that polynomial $R^{(k)}(X)$ contains all the monomials of degree up to k (not only monomials of degree exactly k as previously). Employing relation (8) is no necessary for binary plaintext space. Polynomial evaluation can be performed using only $R^{(d)}(X)$ as it contains all the needed monomial evaluations. On the other hand, a new coefficient packing polynomial should be used:

$$C(X) = \sum_{1 \le k \le d} \sum_{0 \le e_1 < \ldots < e_k < n} c_{e_1,\ldots,e_k} \cdot X^{p_k} \qquad (9)$$

where

$$p_k = N - \sum_{1 \le i \le k} e_i \cdot n^{k-i} - e_1 \cdot \sum_{k \le i \le d-1} n^i.$$

The final evaluation is performed by multiplying the newly introduced coefficient packing polynomial with $R^{(d)}(X)$. As previously, the N-th degree coefficient of the result is the evaluation $P(a_0, \ldots, a_{n-1})$.

[3] Observe that the term $\sum_{1 \le i \le k} e_i \cdot n^{k-i}$ from the X-th power cancels out when $e_1, \ldots, e_k$ are equal in (4) and (6).

Polynomial $P(x_0,\dots,x_{n-1})$ homomorphic evaluation. Ciphertext `ct` encrypts the polynomial $Q(X)$ in which the values $a_0,\dots,a_{n-1}$ are coefficient packed. Polynomial $R^{(k)}(X)$ can be homomorphically computed using multiplication and ring homomorphism operations applied on the packed ciphertext `ct`:

$$\texttt{Enc}_{pk}\left(R^{(k)}(X)\right) \equiv \texttt{ct}\cdot\ldots\cdot\phi^{n^{k-1}}(\texttt{ct}),\, 1\le k\le d$$

With homomorphic encryption schemes defined in Section 2 the best way to compute this expression is to use a tree-shaped structure to perform the multiplications. The homomorphic cryptosystem should support a logarithmic (in the degree d of the polynomial) multiplicative depth. As $R^{(k)}(X)$ computations for different k share common parts we can further decrease the number of employed homomorphic multiplications by a logarithmic factor.

Homomorphic multiplication with a plaintext input is used to compute monomials multiplied by respective polynomial P coefficients (i.e. terms $C^{(k)}(X)\cdot R^{(k)}(X)$) and homomorphic additions for the final sum. The decryption of the obtained ciphertext gives (in the highest degree coefficient) the polynomial $P(x_0,\dots,x_{n-1})$ evaluated at point $a_0,\dots,a_{n-1}$.

For binary plaintext space evaluating polynomial is simpler as only a single $R^{(d)}(X)$ must be computed. This saves several homomorphic operations. The multiplicative depth of the HE scheme remains the same.

In Table 1 are shown the complexity values in terms of homomorphic operations of polynomial evaluations for different kind of operations.

Operations\plaintext space	$\mathbb{Z}_t$	$\mathbb{Z}_2$
`Hom`	$d-1$	$d-1$
`Add`	$d-1$	0
`Mult`	$\frac{d\log_2 d}{2}$	$d-1$
`Mult` with plaintext	d	1

Table 1. Polynomial evaluation complexity in case of binary ($\mathbb{Z}_2$) and general ($\mathbb{Z}_t$) plaintext spaces.

Polynomial evaluation example. Suppose we want to evaluate a polynomial $P(x_0,x_1,x_2)$ of degree $d=2$ with $n=3$ variables at a point a_0,a_1,a_2. The generic formulation of this polynomial is:

$$\begin{aligned}P(x_0,x_1,x_2) = {} & c_0x_0 + c_1x_1 + c_2x_2 + \\ & c_{0,0}x_0^2 + c_{0,1}x_0x_1 + c_{0,2}x_0x_2 + \\ & c_{1,1}x_1^2 + c_{1,2}x_1x_2 + c_{2,2}x_2^2\end{aligned}$$

Let $Q(X) = a_0 + a_1 \cdot X + a_2 \cdot X^2$ be the polynomial packing values a_0, a_1, a_2. Polynomials $R^{(k)}(X)$, $1 \leq k \leq 2$, computed using relation (3) are:

$$\begin{aligned} R^{(1)}(X) &= a_0 && + a_1 X && + a_2 X^2 \\ R^{(2)}(X) &= a_0^2 && + a_0 a_1 X && + a_0 a_2 X^2 + \\ & a_0 a_1 X^3 && + a_1^2 X^4 && + a_1 a_2 X^5 + \\ & a_0 a_2 X^6 && + a_1 a_2 X^7 && + a_2^2 X^8 \end{aligned}$$

The coefficients of $R^{(1)}(X)$ and $R^{(2)}(X)$ are evaluations at point a_0, a_1, a_2 of degree-1 and respectively degree-2 monomials. Polynomials $C^{(1)}(X)$ and $C^{(2)}(X)$ (relation (6)) pack the coefficients of polynomial $P(x_0, x_1, x_2)$:

$$\begin{aligned} C^{(1)}(X) &= c_0 X^N && + c_1 X^{N-1} && + c_2 X^{N-2} \\ C^{(2)}(X) &= c_{0,0} X^N && + c_{0,1} X^{N-1} && + c_{0,2} X^{N-2} \\ & && + c_{1,1} X^{N-4} && + c_{1,2} X^{N-5} \\ & && && + c_{2,2} X^{N-8} \end{aligned}$$

Multiplying together $R^{(k)}(X)$ and $C^{(k)}(X)$, $1 \leq k \leq 2$, and summing up the results we obtain in the degree-N coefficient the polynomial P evaluated at point a_0, a_1, a_2. We note that only 6 out of 9 coefficients of $R^{(2)}(X)$ participate in the final computation. The other 3 (X^3, X^6 and X^7 coefficients) are the "useless" coefficients we talked about earlier.

In case of binary plaintext space polynomial $P(x_0, x_1, x_2)$ formulation is simpler because the square terms disappear:

$$\begin{aligned} P(x_0, x_1, x_2) &= c_0 x_0 && + c_1 x_1 && + c_2 x_2 && + \\ & c_{0,1} x_0 x_1 && + c_{0,2} x_0 x_2 && + c_{1,2} x_1 x_2 \end{aligned}$$

$R^{(2)}(X)$ and $C(X)$ polynomials found using relations (4) and (9) are:

$$\begin{aligned} R^{(2)}(X) &= a_0 && + a_0 a_1 X && + a_0 a_2 X^2 + \\ & a_0 a_1 X^3 && + a_1 X^4 && + a_1 a_2 X^5 + \\ & a_0 a_2 X^6 && + a_1 a_2 X^7 && + a_2 X^8 \end{aligned}$$

$$\begin{aligned} C(X) &= c_0 X^N && + c_{0,1} X^{N-1} && + c_{0,2} X^{N-2} \\ & && + c_1 X^{N-4} && + c_{1,2} X^{N-5} \\ & && && + c_2 X^{N-8} \end{aligned}$$

$R^{(2)}(X)$ contains all the degree-1 monomial evaluations in addition to degree-2 ones. The coefficient packing polynomial $C(X)$ has all polynomial $P(x_0, x_1, x_2)$ coefficients in the right place. Multiplying these polynomials gives in the N-th degree coefficient the evaluation $P(a_0, a_1, a_2)$.

4 Experimentations

We have implemented the optimized polynomial evaluation method using the HELib library [16, 17]. HELib is an open-source library implementing BGV scheme introduced in [5] together with some utility functions. Cyclotomic polynomials are used in HELib as the irreducible modulus of the plaintext and respectively the ciphertext rings. A workstation with an Intel Xeon E3-1240 (3.50GHz) processor and 16GB of RAM was used to execute test applications.

Let $\Phi_m(X)$ be the m-th cyclotomic polynomial. The degree of $\Phi_m(X)$ is given by Euler's totient function $\varphi(m)$. HELib implements a BGV variant in which the polynomial rings are of the form $\mathbb{A} = \mathbb{Z}[X]/\Phi_m(X)$. The native plaintext space is defined by elements over $\mathbb{A}_2$ but other plaintext spaces in the form $\mathbb{A}_{p^r}$ with p an arbitrary, small prime (not dividing m and r) are also possible. The ciphertext space consists of polynomials over $\mathbb{A}_q$ where q is an odd modulus evolving with the homomorphic evaluation. More specifically, there are L modulus $q_1 < q_2 < \cdots < q_L$ where freshly encrypted ciphertexts defined over q_L.

As such, the maximal ciphertext coefficient size is chosen automatically as a function of the multiplication levels L to support. The security of the obtained homomorphic encryption scheme (security of the RLWE instance) depends on the cyclotomic polynomial degree and on the ciphertext coefficient size. Many other parameters allow to fine tune HELib execution performance. In our experiments we limit ourselves to the selection of the following parameters: plaintext modulo t, number of multiplication levels L and cyclotomic polynomial order m.

Under certain conditions, polynomial $\Phi_m(X)$ can be factored modulo t (the modulo for the plaintext coefficients), i.e.

$$\Phi_m(X) = F_1(X) \cdot F_2(X) \cdot \ldots \cdot F_w(X) \mod t.$$

Each factor $F_i(X)$ have the same degree $\frac{\varphi(m)}{w}$, where w is the number of factors. The polynomial evaluation method we propose can be implemented either using the full polynomial or the polynomials of each slot independently (i.e. in batching mode). In the later case our method is able to evaluate w different multivariate polynomials in parallel. The points over which polynomials are evaluated are also different. The drawback of using batching is that the polynomials which can be evaluated are smaller, due to relation (5) which should stay valid. In our experiments we test only the first case, thus the largest possible polynomials are evaluated.

Table 2 shows the results of our evaluation methods when varying the plaintext modulo t and the cyclotomic polynomial order m for multivariate polynomials with degree d , $1 \leq d \leq 4$, in the form defined by relation (1). All the experiments have been performed with at least 128 bits of security. Column "m" gives the cyclotomic polynomial order defining the ciphertext space, "deg" is the degree d of the multivariate polynomial $P(x_0, \ldots, x_{n-1})$, "#vars" is the number of variables n, "L" is the number of multiplication levels the HELib is configured with, "ct. size" is the ciphertext size in MB and "time" is the evaluation time of multivariate polynomial in seconds (i.e. computing expression $\sum_{1 \leq k \leq d} C^{(k)}(X) \cdot R^{(k)}(X)$). Obtained evaluation time is an average over 10 executions. Ciphertext ring homomorphism $(X \to X^t)$ and ciphertext multiplication are the predominant part of the computation. As expected, the execution

			plaintext space size								
			2			2^8			2^{16}		
m	deg	#vars	L	ct. size	time	L	ct. size	time	L	ct. size	time
10007	1	10006	1	0.025	0.002	3	0.077	0.006	3	0.077	0.006
	2	100	3	0.077	0.034	5	0.129	0.069	5	0.129	0.069
	3	21	3	0.077	0.079	5	0.129	0.143	7	0.181	0.223
	4	10	4	0.077	0.158	5	0.129	0.260	7	0.181	0.403
100003	1	100002	1	0.262	0.019	3	0.783	0.056	3	0.783	0.056
	2	316	3	0.783	0.440	5	1.303	0.695	5	1.303	0.699
	3	46	3	0.783	1.054	5	1.303	1.444	7	1.822	1.883
	4	17	4	0.783	2.097	5	1.303	2.627	7	1.822	3.323
1000003	1	1000002	1	2.984	0.197	3	8.188	0.787	3	8.188	0.784
	2	1000	3	8.188	5.157	5	13.354	11.024	5	13.354	10.412
	3	100	3	8.188	12.439	5	13.354	22.449	7	18.515	29.281
	4	31	4	8.188	24.965	5	13.354	40.858	7	18.515	51.578

Table 2. Results of polynomial evaluation method using HELib.

times for the polynomial evaluation increases with its degree but also with the parameter m defining the ciphertext space and parameter t, the plaintext modulus.

Quadratic classifier.

Let us now present a possible application of the polynomial evaluation method we propose here and compare the results with those obtained in [25].

We investigate the performances of our method in the context of a classification algorithm used by a remote service to label the residentials buildings in a small district based on their energy consumption. A basic Gaussian classifier can be adapted such that the prediction step is executed on homomorphically encrypted data. As such, given an encrypted attribute vector x, the purpose is to predict its class label based on the learning model acquired during the training step. We focus here only on the labeling step using private data and we suppose that the model building was realized previously in the clear domain.

In the case of a Gaussian Classifier, each class C_j from the m classes defined during the training phase is assumed characterized by a Gaussian distribution with a mean μ_j and a covariance matrix Σ_j. The mean of a class C_j is the vector $\mu_j \in \mathbb{R}^n$: $\mu_j^\intercal = \{\mu_{j_i}\}$ with μ_{j_i} the mean for the components i of the examples vectors x belonging to class C_j (i.e. $\mu_{j_i} = \frac{\sum_i^n x(i)}{n}$). For vectors with n features, the covariance matrix of a class C_j is a positive semi-definite matrix of size $n \times n$ computed as: $\Sigma_j = \{c(a, b)\}$ with $a, b \in \{1, \ldots, n\}$ and $c(a, b)$ the covariance between the features a and b, measuring their tendency to vary together.

A feature vector x from the training set T is thus classified by measuring a Mahalanobis distance from x to each of the classes and by selecting the minimal norm. The main steps of the prediction phase of the Gaussian classification algorithm are Steps 4-6 from Algorithm 1. The training phase realized on T_0, the set of training vectors x_0, has been realized before, resulting in a model with m classes. After computing the mean and the covariance of each class C_j (Steps 1-3), a class label is predicted for each testing vector $x \in T$.

Algorithm 1 Gaussian classifier - prediction step

```
Require: $T_0 = \{x_0 \in \mathbb{R}^n\}$; $T = \{x \in \mathbb{R}^n\}$ ; $m$ classes $C_j$
 1: for $\forall C_j$, $j \in \{1, \ldots, m\}$ do
 2:     compute $\mu_j$ and $\Sigma_j$ using $x_0$
 3: end for
 4: for $x \in T$ do
 5:     compute $d_M(x, C_j)$, $\forall j \in \{1, \ldots, m\}$
 6:     $C(x) \leftarrow argmin(d_M(x, C_j))$
 7: end for
Ensure: $C(x)$, $\forall x \in T$
```

It seems then that the most important step to be performed on homomorphic encrypted data is the computation of distances between the attribute vector x and the classes. The Mahanalobis distance from an encrypted vector x to a class c_j is defined as:

$$d_M^2(x, C_j) = (x - \mu_j)^\intercal \Sigma_j^{-1} (x - \mu_j).$$

Note that in the particular case where the features are uncorrelated or of a unidimensional feature vector the Mahalanobis distance is equivalent to the Euclidean distance.

For their experiments, the authors from [25] consider an additive Paillier cryptosystem as well as BGV cryptosystem as implemented in HELib library to classify 40 residential profiles using a feature vector size of 6. For the HELib-based prototype, they use the batching technique in two different ways. In a first solution, for a given attribute vector x with n elements, each of the attributes x_i, $i = 1, \ldots, n$, is embedded in a different plaintext slot in the form of an integer modulo 2^8. This allows to encrypt all the attributes of x in the same ciphertext. The references, i.e. the means of the classes, are represented as m vectors of dimension n. As such, for one instance to label, they obtain m ciphertexts corresponding to the encrypted distances to each class. When such a ciphertext is decrypted, the sum on the slots for the obtained plaintext gives the clear distance to the associated class.

In the second solution,they exploit the free plaintexts slots by remarking that usually the number of slots is much larger than the number of attributes and, for a single instance x of dimension n to label with regards to m classes, they replicate it m times and embedded into the slots of a plaintext, by padding with 0 the remaining space. In this configuration, the means are expressed as a single array of dimension $m \times n$ and all the distances are computed in the same time using a single ciphertext. Once received and decrypted, one can obtain the clear distances by making the sum on sub-sets of successive slots. The necessary condition for the second approach is that the number of slots has to be higher or equal to $m \times n$.

Even if the number of operations to be executed on homomorphic domain is reduced through batching (more specifically, for the second approach, one ciphertext-ciphertext multiplication, two multiplications between a ciphertext and a plaintext, a sum between two ciphertext and a sum between a ciphertext and a plaintext), at the end, they have to perform a quite costly operation i.e. a running sum to recuperate the actual distances by bunches of m slots.

With our approach using multivariate polynomials combined with the batching, it is possible to ameliorate even more the evaluation times of the distances. All we need is to evaluate a degree-2 multivariate polynomial with $n = 6$ variables (i.e. the degree of the slot defining plaintext space polynomial should be at least n^2), embedded in a number of slots equal to the number of vectors we want to classify in parallel (in the example, 40 residential energy consumption therefore at least 40 slots).

Table 3 resumes the results we obtained for two configurations of parameters in HELib with a security level (column λ) similar to the one determined by the tests in [25] (and at least 80 bits). The plaintext module 2^8 is used in the experiments. As before, m stands for the cyclotomic polynomial order and L is the number multiplication levels. The overall polynomial which will be encrypted as a single ciphertext can be dived into "#fact" polynomials, each one of degree "deg_fact". The total execution time (column "time") is expressed in seconds and the ciphertext size in MB (column "ct. size"). The execution times and ciphertext sizes are a lot smaller than the ones obtained by the authors of [25]. This is partly due to a smaller number of homomorphic operations (no need to add slots together) and to a smaller multiplications level.

m	deg_fact	#fact	L	λ	time	ct. size
2113	44	48	2	173	0.004	0.340
3191	55	58	2	315	0.005	0.514

Table 3. Results for quadratic classifier with HELib.

5 Conclusion

This paper presents a new method to efficiently evaluate low-degree multivariate polynomial over homomorphic encrypted data. With this new technique applicable to RLWE based homomorphic schemes, one can perform all types of computation, with the condition to express it using polynomial (of relatively low-degree) operations. Since all the coefficients of the plaintext space polynomial are used to encode the messages, this method is more efficient than the usual case in which only the lowest degree of the polynomial is used. Moreover, as shown by the experiments we conducted, our method is compatible with the batching technique allowing to perform operations in a SIMD manner.

We have implemented and executed the proposed polynomial evaluation method using the HELib library. Besides measuring the performance of the evaluation method within diverse settings, we have tested its performance for a machine learning application (namely a quadratic classification algorithm).

In future works, we plan to investigate the case when $n^d > \deg(\Phi(X))$, which will permit to evaluate higher degree multivariate polynomials. Another research line, more applicative, is to use the evaluation method on homomorphically encrypted data for more complex classes of machine learning algorithms.

References

1. Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *arXiv preprint arXiv:1704.03578*, 2017.
2. Charlotte Bonte, Carl Bootland, Joppe W Bos, Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Faster homomorphic function evaluation using non-integral base encoding. *IACR Cryptology ePrint Archive*, 2017:333, 2017.
3. Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine Learning Classification over Encrypted Data. In *NDSS*. The Internet Society, 2015.
4. 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.
5. 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.
6. Zvika Brakerski and Vinod Vaikuntanathan. Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.
7. Melvin A. Breuer. Generation of optimal code for expressions via factorization. *Commun. ACM*, 12(6):333–340, 1969.
8. Sergiu Carpov, Paul Dubrulle, and Renaud Sirdey. Armadillo: A Compilation Chain for Privacy Preserving Applications. In *SCC@ASIACCS*, pages 13–19. ACM, 2015.
9. Sergiu Carpov and Renaud Sirdey. Another Compression Method for Homomorphic Ciphertexts. In *SCC@AsiaCCS*, pages 44–50. ACM, 2016.
10. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In *Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22*, pages 3–33. Springer, 2016.
11. Anamaria Costache, Nigel P Smart, Srinivas Vivek, and Adrian Waller. Fixed point arithmetic in she scheme. *IACR Cryptology ePrint Archive*, 2016:250, 2016.
12. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.
13. Craig Gentry et al. Fully homomorphic encryption using ideal lattices. In *STOC*, volume 9, pages 169–178, 2009.

14. 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. Springer-Verlag, 2012.
15. Thore Graepel, Kristin E. Lauter, and Michael Naehrig. ML Confidential: Machine Learning on Encrypted Data. In *ICISC*, volume 7839 of *Lecture Notes in Computer Science*, pages 1–21. Springer, 2012.
16. Shai Halevi and Victor Shoup. Algorithms in HElib. In *CRYPTO*, volume 8616 of *Lecture Notes in Computer Science*, pages 554–571, 2014.
17. Shai Halevi and Victor Shoup. Bootstrapping for HElib. In *EUROCRYPT*, volume 9056 of *Lecture Notes in Computer Science*, pages 641–670. Springer, 2015.
18. Anup Hosangadi, Farzan Fallah, and Ryan Kastner. Optimizing Polynomial Expressions by Algebraic Factorization and Common Subexpression Elimination. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 25:2012–2022, 2006.
19. Charles E. Leiserson, Liyun Li, Marc Moreno Maza, and Yuzhen Xie. Efficient Evaluation of Large Polynomials. In *ICMS*, volume 6327 of *Lecture Notes in Computer Science*, pages 342–353. Springer, 2010.
20. Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1219–1234. ACM, 2012.
21. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On Ideal Lattices and Learning with Errors over Rings. In *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.
22. 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.
23. Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.
24. Nigel P Smart and Frederik Vercauteren. Fully homomorphic simd operations. *Designs, codes and cryptography*, pages 1–25, 2014.
25. Oana Stan, Mohamed-Haykel Zayani, Renaud Sirdey, Amira Ben Hamida, Alessandro Ferreira Leite, and Malek Mziou-Sallami. A new crypto-classifier service for energy efficiency in smart cities. *IACR Cryptology ePrint Archive*, 2017:1212, 2017.
26. Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 24–43. Springer, 2010.
27. Masaya Yasuda, Takeshi Shimoyama, Jun Kogure, Kazuhiro Yokoyama, and Takeshi Koshiba. Packed Homomorphic Encryption Based on Ideal Lattices and Its Application to Biometrics. In *CD-ARES Workshops*, volume 8128 of *Lecture Notes in Computer Science*, pages 55–74. Springer, 2013.
28. Masaya Yasuda, Takeshi Shimoyama, Jun Kogure, Kazuhiro Yokoyama, and Takeshi Koshiba. Practical Packing Method in Somewhat Homomorphic Encryption. In *DPM/SETOP*, volume 8247 of *Lecture Notes in Computer Science*, pages 34–50, 2013.
29. Masaya Yasuda, Takeshi Shimoyama, Jun Kogure, Kazuhiro Yokoyama, and Takeshi Koshiba. Secure pattern matching using somewhat homomorphic encryption. In *CCSW*, pages 65–76. ACM, 2013.