IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 8 June 2017; revised 7 October 2017; accepted 4 January 2018.
Date of publication 17 January 2018; date of current version 9 June 2020.

Digital Object Identifier 10.1109/TETC.2018.2794611

Private Machine Learning Classification Based on Fully Homomorphic Encryption

XIAOQIANG SUN, PENG ZHANG, JOSEPH K. LIU, JIANPING YU, AND WEIXIN XIE

X. Sun, P. Zhang, J. Yu, and W. Xie are with the ATR Key Laboratory of National Defense Technology,
College of Information Engineering, Shenzhen University, Shenzhen 518060, China
J.K. Liu is with the Faculty of Information Technology, Monash University, Clayton, VIC 3800, Australia
CORRESPONDING AUTHOR: P. ZHANG (zhangp@szu.edu.cn)

ABSTRACT Machine learning classification is an useful tool for trend prediction by analyzing big data. As supporting homomorphic operations over encrypted data without decryption, fully homomorphic encryption (FHE) contributes to machine learning classification without leaking user privacy, especially in the outsouring scenario. In this paper, we propose an improved FHE scheme based on HElib, which is a FHE library implemented based on Brakerski's FHE scheme. Our improvement focuses on two aspects. On the one hand, we first use the relinearization technique to reduce the ciphertext size, and then the modulus switching technique is used to reduce the modulus and decryption noise. On the other hand, we need no relinearization and modulus switching if there is additive homomorphic or no homomorphic operation in the multiplicative ciphertext's next homomorphic operation. Homomorphic comparison protocol, private hyperplane decision-based classification and private Naïve Bayes classification are implemented by additive homomorphic and multiplicative homomorphic first. In our homomorphic comparison protocol, the number of interactions is reduced from 3 to 1. We choose the proposed FHE scheme to implement private decision tree classification. Simulation results show that the efficiency of our FHE scheme and implementation of private decision tree classification are more efficient than other two schemes.

INDEX TERMS Machine learning classification, privacy preserving, fully homomorphic encryption, hyperplane decision-based, Naïve Bayes, decision tree

I. INTRODUCTION

Machine learning classification has been widely used in many research fields, such as medical computation, genomics prediction, spam detection. As outsourcing computation (e.g., cloud computing) becomes one of the most popular computation paradigms, and some outsourced data (e.g., health data [1], [2], image data [3]–[6]) refers to user privacy, data privacy [7]–[12] should be preserved first. Thus, several private machine learning classification schemes have been constructed recently [13]–[15], which can implement machine learning classification over encrypted data.

Liu *et al.* [13] proposed a privacy preserving patient-centric clinical decision support system, which helps to diagnose the patient's disease in a privacy preserving way. In the proposed system, patient's data are encrypted by Paillier cryptosystem [16], then they are used to train the model of Naïve Bayes classification [17] without decryption. As Paillier cryptosystem only supports additive homomorphic, its multiplicative homomorphic is achieved by the SM protocol [18]. Bost *et al.* [14] constructed three private machine learning classifications without leaking user privacy, which include private hyperplane decision-based classification, private Naïve Bayes classification and private decision tree classification. Private hyperplane decision-based classification and private Naïve Bayes classification are implemented by additive homomorphic cryptosystems (Quadratic Residuosity [19] and Paillier cryptosystems [16]). Because Brakerski's fully homomorphic encryption (FHE) scheme (BGV12) [20] supports the technique of simple instruction multiple data (SIMD) [21], private decision tree classification is achieved by a FHE library, namely HElib [22]–[24], which is implemented based on BGV12. Based on Gentry's FHE scheme [25], an optimized FHE scheme is used to construct Bayesian spam filter, secure multiple keyword search and secure binary decision tree [15]. However, the optimized FHE scheme doesn't support SIMD. To protect the security of data, Jiang *et al.* [26] used Brakerski's FHE scheme [20] to implement perception algorithm and support vector machines.

As mentioned above, homomorphic encryption can be used for privacy preserving machine learning classification, which was first proposed by Rivest *et al.* [27] in 1978. Generally, homomorphic encryption can be classified into partial homomorphic encryption (e.g., Paillier cryptosystem [16]), which only supports additive homomorphic or multiplicative homomorphic, somewhat homomorphic encryption, which supports limited additive homomorphic and multiplicative homomorphic, and fully homomorphic encryption (e.g., BGV12 [20]), which supports arbitrary additive homomorphic and multiplicative homomorphic. In fact, we don't know which computations are operated before outsourcing data, and it is impossible that data are stored and computed by two different encryption schemes, respectively. As FHE allows any computation over encrypted data, it may be more beneficial to practical application.

There are three dominant methods to construct a FHE scheme. The first version is the ideal lattices based FHE scheme constructed by Gentry [28] in 2009. The second version is the learning with errors (LWE) assumption based FHE scheme [29], which has the advantage of high efficiency without squashing decryption circuits. However, the relinearization technique [30] requires an additional evaluation key, which size increases exponentially when homomorphic operations are complicated. The learning with errors over rings (RLWE) assumption [31] is an enhanced form of the LWE assumption. It has advantages such as simpler construction and higher efficiency, which can be used to construct more efficient FHE schemes [20]–[35]. The third version is the approximate eigenvector based FHE scheme proposed by Gentry *et al.* [25] in 2013, where additive homomorphic and multiplicative homomorphic are just matrix addition and matrix multiplication. These schemes focus on the efficiency of FHE, which should be improved further for private machine learning classification.

A. CONTRIBUTION

First, we propose an improved FHE scheme, which is suitable for private machine learning classification due to its efficiency improvement of multiplicative homomorphic. Compared with HElib [22], our scheme has two advantages in efficiency. The first advantage is, unlike the relinearization technique is taken after the using of modulus switching technique in HElib's each implementation of BGV12's multiplicative homomorphic, we first use the relinearization technique to reduce the ciphertext size, then the modulus switching technique is used to reduce the modulus and decryption noise. The second advantage is we need no relinearization and modulus switching if there is additive homomorphic or no homomorphic operation in the multiplicative ciphertext's next homomorphic operation, while relinearization and modulus switching are used in HElib's every implementation of BGV12's multiplicative homomorphic. In addition, SIMD is used to speed up homomorphic operations.

Second, based on the improved FHE scheme, some private machine learning classification schemes, including hyperplane decision-based classification, Naïve Bayes classification, and decision tree classification, are achieved. In detail, homomorphic comparison protocol, private hyperplane decision-based classification and private Naïve Bayes classification are implemented by FHE first, with no limitation on the possible computation operations over the encrypted data. Compared with Bost's homomorphic comparison protocol, the number of interactions is reduced from 3 to 1 in our homomorphic comparison protocol. In our implementation of the private decision tree classification, due to the efficiency improvement of our proposed FHE scheme, it's more efficient than the FHE library HElib and Khedr's.

B. ORGANIZATION

The rest of the paper is organized as follows. Preliminaries are introduced in Section II. Our improved FHE scheme is presented in Section III. Building blocks for private machine learning classification are shown in Section IV. In Section V, we show how to implement private machine learning classification by our FHE scheme. Detailed simulation results and efficiency analysis are shown in Section VI. Section VII concludes the whole paper.

II. PRELIMINARIES

In this section, we begin with basic notations, formal definitions of cryptographic problems. Then, we give descriptions of three machine learning classifications, which include hyperplane decision-based classification, Naïve Bayes classification and decision tree classification [17].

A. BASIC NOTATIONS

For a real number z, let $\lceil z \rceil$, $\lfloor z \rfloor$ or $\lfloor z \rceil$ denote rounding up, rounding down or the nearest integer of z, respectively. Namely, they are integers in half open intervals $[z, z+1)$, $(z-1, z]$ and $(z-1/2, z+1/2]$, respectively.

For a real number z and an integer p, let $[z]_p$ denote the remainder of z with respect to p, namely $[z]_p = z - \lfloor z/p \rceil \cdot p$. It should be noticed that $[z]_p \in (-p/2, p/2]$.

For two n-dimensional vectors $\vec{a} = (a_0, a_1, \ldots, a_{n-1})$ and $\vec{b} = (b_0, b_1, \ldots, b_{n-1})$, the inner product $\langle \vec{a}, \vec{b} \rangle$ is defined as $\vec{a} \cdot \vec{b}$.

For a security parameter λ, parameter $m = m(\lambda)$, let $R = \mathbb{Z}/\langle \Phi_m(x) \rangle$ be the ring modulo $\Phi_m(x)$, where $\Phi_m(x)$ represents the mth cyclotomic polynomial. Let $R_q = \mathbb{Z}_q[x]/\langle \Phi_m(x) \rangle$ be the ring modulo both the prime modulus q and $\Phi_m(x)$, where $q \geq 2$. For any $\beta > 0$, let $D_\beta(x)$ denote the density function of the Gaussian distribution over the real domain, where $D_\beta(x) = (1/\beta) \cdot \exp(-\pi \cdot (x/\beta)^2)$. The distribution $\bar{\psi}_\beta(q)$ on $\mathbb{Z}_q$ is generated by drawing $y \leftarrow D_\beta$, then output $\lfloor q \cdot y + 1/2 \rfloor$. Let χ denote an error distribution whose coefficients are chosen from $\bar{\psi}_\beta(q)$ randomly.

B. LEARNING WITH ERRORS OVER RINGS

The learning with errors over rings assumption was first introduced by Lyubashevsky *et al.* [31], which is shown as follows.

Definition 1 (RLWE). The $\mathrm{RLWE}_{\lambda,q,\chi}$ assumption is to distinguish following two distributions: (1)$(a, a \cdot s + e) \in$

$R_q \times R_q$, where a and s are chosen from R_q uniform randomly, error term e is selected from χ randomly. (2)$(a, c) \in Unif(R_q \times R_q)$, where Unif represents uniform random.

It has been proved that the RLWE assumption is hard over ideal lattices (Theorem 1) in the Ref. [31], and $(a, b = a \cdot s + e)$ is pseudorandom.

Definition 2 (Shortest Vector Problem [36], SVP). *An input (B, d) for $GapSVP_\gamma$, where B is an n-dimensional lattice basis, $d \in \mathbb{R}$. If $\lambda_1(L(B)) \le d$, it will be a Yes instance, otherwise it will be a No instance, where $\lambda_1(L(B))$ is the minimum distance of the lattice $L(B)$, namely*

$$GapSVP_\gamma(B, d) = \begin{cases} Yes, \lambda_1(L(B)) \le d \\ No, \lambda_1(L(B)) > d. \end{cases}$$

Theorem 1 ([31]). For the security parameter λ, the prime modulus q. There is an efficient distribution χ, which could output a ring element $r \in R$ with the maximum length B. Hence, if there exists an efficient algorithm to solve $RLWE_{\lambda,q,\chi}$, there will be a quantum algorithm for the approximate SVP over the ring R on ideal lattices under the worst case.

C. MACHINE LEARNING CLASSIFICATIONS

Supposed the user has an n-dimensional vector $x = (x_i)_{i=0, 1,\ldots,n-1}$, where $x_i \in \mathbb{R}$. To classify x, the classification algorithm $C_w(x) : \mathbb{R}^n \to c_{k^*}$ is executed based on the model w, where $k^* \in [0, k)$, where k is the number of classes. In the following, we give formal definitions of hyperplane decision-based classification, Naïve Bayes classification and decision tree classification.

1) HYPERPLANE DECISION-BASED CLASSIFICATION

In the hyperplane decision-based classification, the model w includes k weights $(w_i)_{i=0,1,\ldots,k-1}$, where $w_i \in \mathbb{R}^n$. The classification result is

$$k^* = argmax_{i\in[0,k)} \langle w_i, x\rangle,$$

where *argmax* outputs the index i that makes $\langle w_i, x\rangle$ as large as possible.

2) NAïVE BAYES CLASSIFICATION

In the Naïve Bayes classification, each class c_i corresponds to the probability $p(C = c_i)$, where $i = 0, 1, \ldots, k-1$. Supposed the jth element x_j of x is a, it happens with a probability $p(x_j = a|C = c_i)$ under the class c_i, where $j = 0, 1, \ldots, n-1, i = 0, 1, \ldots, k-1$, $a \in D_j$, D_j belongs to x's domain. This classification chooses the class by maximum posteriori probability, which is shown as follows:

$$\begin{aligned} k^* &= argmax_{i\in[0,k)} p(C = c_i|X = x) \\ &= argmax_{i\in[0,k)} \frac{p(C = c_i, X = x)}{p(X = x)} \\ &= argmax_{i\in[0,k)} p(C = c_i, X = x) \\ &= argmax_{i\in[0,k)} p(C = c_i, X_0 = x_0, X_1 = x_1, \ldots, \\ &\quad X_{n-1} = x_{n-1}). \end{aligned}$$

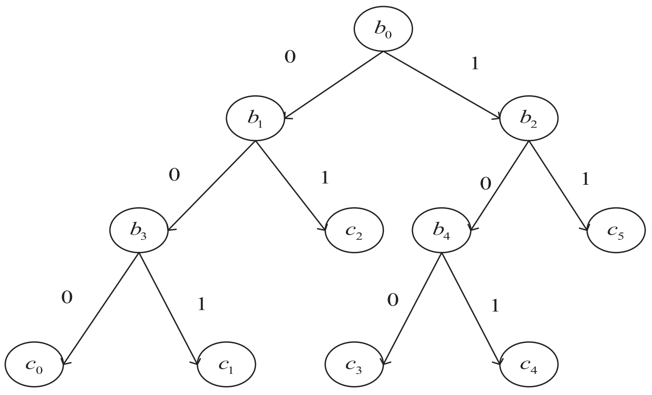


FIGURE I. **Decision tree classification.**

For the fixed x, the normalizing factor $p(X = x)$ can be omitted according to Bayes's rule. Then, $p(C = c_i, X = x)$ is factorized as follows:

$$\begin{aligned} &p(C = c_i, X_0 = x_0, X_1 = x_1, \ldots, X_{n-1} = x_{n-1}) \\ &\quad = p(C = c_i) \cdot \prod\nolimits_{j=0}^{n-1} p(X_j = x_j|C = c_i), \end{aligned}$$

where n features are independent of each other under the given class c_i. For convenience, x_j in our Naïve Bayes classification is discrete and finite.

3) DECISION TREE CLASSIFICATION

Decision tree is a nonparametric classification. The model of decision tree classification includes weights $(w_i)_{i=0,1,\ldots,n-1}$ and l'-bit classes $(c_j)_{j=0,1,\ldots,k-1}$. The interior node represents the division rule, while the leaf node represents the class. Then, the cloud server calculates the comparison result between x and w_i, namely b_i. If $b_i = 1$, it means $x \ge w_i$; otherwise $x < w_i$, where $i = 0, 1, \ldots, n-1$. x is classified by traversing the tree beginning from the root node to the leaf node. The division rule at each node is used for selecting the next subfield until the class is obtained, if $b_i = 1$, we choose the left node; otherwise, we choose the right node, where $i = 0, 1, \ldots, n-1$. Similar to the Ref. [14], we represent the decision tree classification as a polynomial $P = (P_i)_{i=0, 1,\ldots,l'-1}$. For example, in Figure 1, $P_i = b_0(b_2 \cdot c_{5,i} + (1-b_2)(1-b_4)c_{3,i} + (1-b_2)b_4 \cdot c_{4,i}) + (1-b_0)(b_1 \cdot c_{2,i} + (1-b_1)\, b_3 \cdot c_{1,i} + (1-b_1)(1-b_3)c_{0,i})$, $c_{0,i}, c_{1,i}, \ldots, c_{5,i}$ represent ith elements of $c_0, c_1, \ldots, c_5$, respectively, where $i = 0, 1, \ldots, l'-1$. Finally, the classification result $k^* = \sum_{i=0}^{l'-1} P_i \cdot 2^i$.

III. OUR IMPROVED FULLY HOMOMORPHIC ENCRYPTION SCHEME

A. DESCRIPTION OF THE PROPOSED SCHEME

In this section, we describe an improved fully homomorphic encryption scheme $FHE = (Setup, KeyGen, Encrypt, Add, Multiply, Decrypt)$ as follows:

- $FHE.Setup(1^\lambda, L)$: Input the security parameter λ and level L. Then, generate a sufficient large prime modulus $p_i = p_i(\lambda, L) = 1(mod\ m)$, $\quad i = 0, 1, \ldots, L-1$. Let $q_j = \prod_{i=0}^{j} p_i$, where $j = 0, 1, \ldots, L-1$. It satisfied that

$q_0 < q_1 < \cdots < q_{L-1}$. Let χ denote the discrete Gaussian distribution. Let $params = (q_{j=0,1,\ldots,L-1}, \chi)$.

- *FHE.KeyGen*$(params)$: Input $params$, choose $s \in R_2$ randomly whose coefficients are selected from $\{0, \pm 1\}$. Generate $b = [-(a \cdot s + t \cdot e)]_{q_{L-1}}$, where a is sampled from $R_{q_{L-1}}$ uniform randomly, the error term e is chosen from χ uniform randomly, t is the plaintext modulus. The key switching matrix $w_j = (w_{j,0}, w_{j,1})$, where $w_{j,0} = [-w_{j,1} \cdot s + t \cdot e_j + T \cdot s^2]_{q_{L-1}}$, $w_{j,1} \in R_{q_{L-1}}$, $e_j \in \chi$, where $j = 0, 1, \ldots, \lceil \log_T q_{L-1} \rceil - 1$, T is a power of 2. Output the public key $pk = (b, a, (w_j)_{j=0,1,\ldots,\lceil \log_T q_{L-1} \rceil - 1})$ and the secret key $sk = s$.
- *FHE.Encrypt*(pk, m') : The plaintext space is R_t. Sample u and e_i from χ uniform randomly, where $i = 0, 1$. To encrypt the message $m' \in R_t$, generate the ciphertext c as follows:

$$c = ([m' + b \cdot u + t \cdot e_1]_{q_{L-1}}, [a \cdot u + t \cdot e_2]_{q_{L-1}}).$$

- *FHE.Add*(c', c''): Suppose two ciphertexts $c' = ([c'_0]_{q_j}, [c'_1]_{q_j}, \ldots, [c'_r]_{q_j})$ and $c'' = ([c''_0]_{q_j}, [c''_1]_{q_j}, \ldots, [c''_k]_{q_j})$ are under the same secret key, where $r, k \in \{1, 2\}$, let $r \le k$. We compute the additive ciphertext c_{fresh} as follows.
  1) If $r = 1, k = 1$, then $c_{fresh} = ([c'_0 + c''_0]_{q_j}, [c'_1 + c''_1]_{q_j})$.
  2) If $r = 1$, $k = 2$, then $c_{fresh} = ([c'_0 + c''_0]_{q_j}, [c'_1 + c''_1]_{q_j}, [c''_2]_{q_j})$.
  3) If $r = 2$, $k = 2$, then $c_{fresh} = ([c'_0 + c''_0]_{q_j}, [c'_1 + c''_1]_{q_j}, [c'_2 + c''_2]_{q_j})$.

For convenience, let $+_h (-_h)$ denote additive homomorphic.

- *FHE.Multiply*(c', c''): Suppose two ciphertexts $c' = ([c'_0]_{q_j}, [c'_1]_{q_j})$ and $c'' = ([c''_0]_{q_j}, [c''_1]_{q_j})$ are under the same secret key, compute the multiplicative ciphertext $c = (c_0, c_1, c_2)$ as follows:

$$c_0 = [c'_0 \cdot c''_0]_{q_j},$$
$$c_1 = [c'_0 \cdot c''_1 + c'_1 \cdot c''_0]_{q_j},$$
$$c_2 = [c'_1 \cdot c''_1]_{q_j}.$$

  1) If there is additive homomorphic or no homomorphic operation in c's next homomorphic operation, output c.
  2) If there is multiplicative homomorphic in c's next homomorphic operation, the ciphertext size of c is reduced from three ring elements to two ring elements, which is called as the relinearization technique. The new ciphertext $c^* = (c^*_0, c^*_1)$ is calculated as follows:

$$c^*_0 = [c_0 + \sum\nolimits_{i=0}^{\lceil \log_T q_j \rceil - 1} c_{2,i} \cdot w_{i,0}]_{q_j},$$
$$c^*_1 = [c_1 + \sum\nolimits_{i=0}^{\lceil \log_T q_j \rceil - 1} c_{2,i} \cdot w_{i,1}]_{q_j},$$

  where $c_{2,i}$ is the ith element of c_2, $c_2 = \sum_{i=0}^{\lceil \log_T q_j \rceil - 1} c_{2,i} \cdot T^i$. We can also use Gentry's idea of raising the modulus [30] to reduce c's ciphertext size. The decryption noise of c^* can be reduced by the modulus switching technique. Then, the modulus of c^* will be decreased from q_j to q_k, where $k \in [0, L-2]$. For convenience, let $\times_h$ denote multiplicative homomorphic. For example, suppose c''' is a ciphertext under the secret key of c. If we need to compute $c +_h c'''$, we implement the additive homomorphic between c and c''' directly. If we need to compute $c \times_h c'''$, we first convert c to c_{fresh}, then we implement the multiplicative homomorphic between c_{fresh} and c'''.
- *FHE.Decrypt*(c, sk) : Given the ciphertext $c = ([c_0]_{q_{L-1}}, [c_1]_{q_{L-1}})$ without homomorphic operations, output the decryption result

$$m' = [[c_0 + c_1 \cdot s]_{q_{L-1}}]_t.$$

  After homomorphic operations on c, suppose the final ciphertext $c_{fresh} = ([c_{fresh,0}]_{q_j}, [c_{fresh,1}]_{q_j}, \ldots, [c_{fresh,k}]_{q_j})$, then the decryption of c_{fresh} is shown as follows:

$$m' = [[c_{fresh,0} + c_{fresh,1} \cdot s + \cdots + c_{fresh,k} \cdot s^k]_{q_j}]_t,$$

  where $k \in \{1, 2\}$.

Compared with HElib [22], our FHE scheme's improvement focuses on two aspects. First, in HElib's every implementation of BGV12's multiplicative homomorphic, the relinearization technique is taken after the using of modulus switching technique. Namely, the modulus switching technique is applied on three ring elements in each multiplicative homomorphic. In order to improve the efficiency of multiplicative homomorphic, we first use the relinearization technique for reducing the ciphertext size from three ring elements to two ring elements, then the modulus switching technique is applied to reduce the modulus and decryption noise. Second, techniques of relinearization and modulus switching are used in HElib's each implementation of BGV12's multiplicative homomorphic. For the multiplicative homomorphic's efficiency, we use relinearization and modulus switching only if there is multiplicative homomorphic in the multiplicative ciphertext's next homomorphic operation. If there is additive homomorphic or no homomorphic operation in the multiplicative ciphertext's next homomorphic operation, relinearization and modulus switching are eliminated.

B. CORRECTNESS WITHOUT HOMOMORPHIC OPERATIONS

Claim 1. The proposed FHE scheme is correct without homomorphic operations.

Proof. Given the ciphertext $c = (c_0, c_1) = ([m' + b \cdot u + t \cdot e_1]_{q_{L-1}}, [a \cdot u + t \cdot e_2]_{q_{L-1}})$ of plaintext m without homomorphic operations, the secret key of c is s. Decryption of c is shown as follows:

$$\begin{aligned}[[c_0 + c_1 \cdot s]_{q_{L-1}}]_t &= [[m' + b\cdot u + t\cdot e_1 + (a\cdot u + t\cdot e_2)\cdot s]_{q_{L-1}}]_t \\ &= [[m' - (a\cdot s + t\cdot e)\cdot u + t\cdot e_1 + (a\cdot u + t\cdot e_2)\cdot \\ &\quad s]_{q_{L-1}}]_t \\ &= [m' - t\cdot e\cdot u + t\cdot e_1 + t\cdot e_2\cdot s]_t \\ &= m',\end{aligned}$$

where $|m' - t\cdot e\cdot u + t\cdot e_1 + t\cdot e_2\cdot s| < q_{L-1}/2$. Hence, our FHE scheme is correct without homomorphic operations. □

C. CORRECTNESS OF HOMOMORPHIC OPERATIONS

Claim 2. The proposed FHE scheme satisfies both multiplicative homomorphic and additive homomorphic.

Proof (1). The analysis of multiplicative homomorphic is shown as follows. Suppose two ciphertexts c' and c'' are under the same secret key s, plaintexts of c' and c'' are m' and m'', respectively, where $c' = (c'_0, c'_1) = ([m' + b\cdot u' + t\cdot e'_1]_{q_{L-1}}, [a\cdot u' + t\cdot e'_2]_{q_{L-1}})$, $c'' = (c''_0, c''_1) = ([m'' + b\cdot u'' + t\cdot e''_1]_{q_{L-1}}, [a\cdot u'' + t\cdot e''_2]_{q_{L-1}})$, $u', e'_1, e'_2, u'', e''_1, e''_2$ are randomly sampled from χ. The multiplicative homomorphic between c' and c'' is $c_{mul} = (c_{mul,0}, c_{mul,1}, c_{mul,2})$, where $c_{mul,0} = [c'_0\cdot c''_0]_{q_{L-1}}$, $c_{mul,1} = [c'_0\cdot c''_1 + c'_1\cdot c''_0]_{q_{L-1}}$, $c_{mul,2} = [c'_1\cdot c''_1]_{q_{L-1}}$, then

$$\begin{aligned}&[[c_{mul,0} + c_{mul,1}\cdot s + c_{mul,2}\cdot s^2]_{q_{L-1}}]_t \\ &= [[c'_0\cdot c''_0 + (c'_0\cdot c''_1 + c'_1\cdot c''_0)\cdot s + c'_1\cdot c''_1\cdot s^2]_{q_{L-1}}]_t \\ &= [[(m' + b\cdot u' + t\cdot e'_1)(m'' + b\cdot u'' + t\cdot e''_1) \\ &\quad + ((m' + b\cdot u' + t\cdot e'_1)(a\cdot u'' + t\cdot e''_2) + (m'' + b\cdot u'' \\ &\quad + t\cdot e''_1)(a\cdot u' + t\cdot e'_2))\cdot s + (a\cdot u' + t\cdot e'_2)(a\cdot u'' \\ &\quad + t\cdot e''_2)\cdot s^2]_{q_{L-1}}]_t \\ &= [[m'\cdot m'' + b\cdot u'\cdot m'' + t\cdot e'_1\cdot m'' + m'\cdot b\cdot u'' \\ &\quad + b^2\cdot u'\cdot u'' + t\cdot e'_1\cdot b\cdot u'' + (m' + b\cdot u')t\cdot e''_1 \\ &\quad + ((m' + b\cdot u')a\cdot u'' + t\cdot e'_1\cdot a\cdot u'' + (m' + b\cdot u') \\ &\quad \cdot t\cdot e''_2 + t^2\cdot e'_1\cdot e''_2 + (m'' + b\cdot u'')a\cdot u' + t^2\cdot e''_1 \\ &\quad \cdot e'_2 + (m'' + b\cdot u'')t\cdot e'_2 + t\cdot e''_1\cdot a\cdot u')\cdot s + (a^2\cdot \\ &\quad u'\cdot u'' + t^2\cdot e'_2\cdot e''_2 + t\cdot e'_2\cdot a\cdot u'' + a\cdot u'\cdot t\cdot e''_2) \\ &\quad \cdot s^2]_{q_{L-1}}]_t \\ &= [m'\cdot m'' + (-t\cdot e\cdot u'\cdot m'') + t\cdot e'_1\cdot m'' + m' \\ &\quad \cdot u''(-t\cdot e) + b\cdot u'\cdot u''(-t\cdot e) + t\cdot e'_1\cdot u''(-t\cdot e) \\ &\quad + m'\cdot t\cdot e''_1 + t\cdot e''_1\cdot u'(-t\cdot e) + (m'\cdot t\cdot e''_2 - u'\cdot t \\ &\quad \cdot e''_2\cdot t\cdot e + t^2\cdot e'_1\cdot e''_2 - a\cdot t\cdot e\cdot u'\cdot u'' + t^2\cdot e''_1\cdot \\ &\quad e'_2 + m''\cdot t\cdot e'_2 - t\cdot u''\cdot e'_2\cdot t\cdot e)\cdot s + (t^2\cdot e'_1\cdot e''_2) \\ &\quad \cdot s^2]_t \\ &= [m'\cdot m'' + (-t\cdot e\cdot u'\cdot m'') + t\cdot e'_1\cdot m'' + m'\cdot \\ &\quad u''(-t\cdot e) + (t\cdot e)u'\cdot u''(t\cdot e) + t\cdot e'_1\cdot u''(-t\cdot e) \\ &\quad + m'\cdot t\cdot e''_1 + t\cdot e''_1\cdot u'(-t\cdot e) + (m'\cdot t\cdot e''_2 - u'\cdot t \\ &\quad \cdot e''_2\cdot t\cdot e + t^2\cdot e'_1\cdot e''_2 + t^2\cdot e''_1\cdot e'_2 + m''\cdot t\cdot e'_2 \\ &\quad - t\cdot u''\cdot e'_2\cdot t\cdot e)\cdot s + (t^2\cdot e'_1\cdot e''_2)\cdot s^2]_t \\ &= m'\cdot m'',\end{aligned}$$

where $|m'\cdot m'' + (-t\cdot e\cdot u'\cdot m'') + t\cdot e'_1\cdot m'' + m'\cdot u''(-t\cdot e) + (t\cdot e)u'\cdot u''(t\cdot e) + t\cdot e'_1\cdot u''(-t\cdot e) + m'\cdot t\cdot e''_1 + t\cdot e''_1\cdot u'(-t\cdot e) + (m'\cdot t\cdot e''_2 - u'\cdot t\cdot e''_2\cdot t\cdot e + t^2\cdot e'_1\cdot e''_2 + t^2\cdot e''_1\cdot e'_2 + m''\cdot t\cdot e'_2 - t\cdot u''\cdot e'_2\cdot t\cdot e)\cdot s + (t^2\cdot e'_1\cdot e''_2)\cdot s^2| < q_{L-1}/2$. The technique of relinearization or modulus reduction will not affect our scheme's multiplicative homomorphic.

(2) The analysis of additive homomorphic is divided into following three cases.

a) Suppose two ciphertexts c', c'' are under the same secret key s, plaintexts of c' and c'' are m' and m'', respectively, where $c' = (c'_0, c'_1) = ([m' + b\cdot u' + t\cdot e'_1]_{q_{L-1}}, [a\cdot u' + t\cdot e'_2]_{q_{L-1}})$, $c'' = (c''_0, c''_1) = ([m'' + b\cdot u'' + t\cdot e''_1]_{q_{L-1}}, [a\cdot u'' + t\cdot e''_2]_{q_{L-1}})$, $u', e'_1, e'_2, u'', e''_1, e''_2$ are randomly sampled from χ. The additive homomorphic between c' and c'' is $c_{add} = (c_{add,0}, c_{add,1}) = ([c'_0 + c''_0]_{q_{L-1}}, [c'_1 + c''_1]_{q_{L-1}})$, then

$$\begin{aligned}[[c_{add,0} + c_{add,1}\cdot s]_{q_{L-1}}]_t &= [[c'_0 + c''_0 + (c'_1 + c''_1)\cdot s]_{q_{L-1}}]_t \\ &= [[(c'_0 + c'_1\cdot s) + (c''_0 + c''_1\cdot s)]_{q_{L-1}}]_t \\ &= [m' - t\cdot e\cdot u' + t\cdot e'_1 + t\cdot e'_2\cdot s + m'' \\ &\quad - t\cdot e\cdot u' + t\cdot e'_1 + t\cdot e'_2\cdot s]_t \\ &= m' + m'',\end{aligned}$$

where $|m' + m'' - t\cdot e\cdot u' + t\cdot e'_1 + t\cdot e'_2\cdot s - t\cdot e\cdot u' + t\cdot e'_1 + t\cdot e'_2\cdot s| < q_{L-1}/2$.

b) Suppose two ciphertexts $c' = ([c'_0]_{q_{L-1}}, [c'_1]_{q_{L-1}})$, $c'' = ([c''_0]_{q_{L-1}}, [c''_1]_{q_{L-1}}, [c''_2]_{q_{L-1}})$ are under the same secret key s, plaintexts of c' and c'' are m' and m'', respectively. The additive homomorphic between c' and c'' is $c_{add} = (c_{add,0}, c_{add,1}, c_{add,2}) = ([c'_0 + c''_0]_{q_{L-1}}, [c'_1 + c''_1]_{q_{L-1}}, [c''_2]_{q_{L-1}})$, then

$$\begin{aligned}[[c_{add,0} + c_{add,1}\cdot s + c_{add,2}\cdot s^2]_{q_{L-1}}]_t &= [[c'_0 + c''_0 + (c'_1 + c''_1)\cdot s \\ &\quad + c''_2\cdot s^2]_{q_{L-1}}]_t \\ &= [[(c'_0 + c'_1\cdot s) + (c''_0 \\ &\quad + c''_1\cdot s + c''_2\cdot s^2)]_{q_{L-1}}]_t \\ &= [m' + e' + m'' + e'']_t \\ &= m' + m'',\end{aligned}$$

where $|m' + m'' + e' + e''| < q_{L-1}/2$, e' and e'' are decryption noises of c' and c'', respectively, $[e']_t = 0$, $[e'']_t = 0$.

c) Suppose two ciphertexts $c' = ([c'_0]_{q_{L-1}}, [c'_1]_{q_{L-1}}, [c'_2]_{q_{L-1}})$, $c'' = ([c''_0]_{q_{L-1}}, [c''_1]_{q_{L-1}}, [c''_2]_{q_{L-1}})$ are under the same secret key s, plaintexts of c' and c'' are m' and m'', respectively. The additive homomorphic between c' and c'' is $c_{add} = (c_{add,0}, c_{add,1}, c_{add,2}) = ([c'_0 + c''_0]_{q_{L-1}}, [c'_1 + c''_1]_{q_{L-1}}, [c''_1 + c''_2]_{q_{L-1}})$, then

$$\begin{aligned}[[c_{add,0} + c_{add,1}\cdot s + c_{add,2}\cdot s^2]_{q_{L-1}}]_t &= [[c'_0 + c''_0 + (c'_1 + c''_1)\cdot s \\ &\quad + (c''_1 + c''_2)\cdot s^2]_{q_{L-1}}]_t \\ &= [[(c'_0 + c'_1\cdot s + c'_2\cdot s^2) \\ &\quad + (c''_0 + c''_1\cdot s + c''_2\cdot s^2)]_{q_{L-1}}]_t \\ &= [m' + e' + m'' + e'']_t \\ &= m' + m'',\end{aligned}$$

where $|m' + m'' + e' + e''| < q_{L-1}/2$, e' and e'' are decryption noises of c' and c'', respectively, $[e']_t = 0$, $[e'']_t = 0$.

Consequently, our improved FHE scheme supports both multiplicative homomorphic and additive homomorphic. □

D. SECURITY ANALYSIS

Claim 3. For any parameters λ, L, q_{L-1} and χ, which satisfying conditions of the proposed FHE scheme, if $RLWE_{\lambda,q_{L-1},\chi}$ assumption is hard, then our FHE scheme could resist chosen plaintext attack (CPA).

Proof. Let $\mathcal{A}$ be a probabilistic polynomial adversary, which has the advantage ϵ to distinguish the challenge ciphertext in the CPA model. Detailed steps of proof are shown as follows.

- *Setup*: Input the security parameter λ and level L, the challenger runs $FHE.Setup(1^\lambda, L)$ to get $params$.
- *Queries* 1: The challenger runs $FHE.KeyGen(params)$ to get secret key sk_i and public key pk_i, where $i = 1, \ldots, t$, t is the polynomial number of query. Then, $(sk_i, pk_i)_{i=1,\ldots,t}$ is returned to $\mathcal{A}$.
- *Challenge*: After polynomial queries, $\mathcal{A}$ outputs two different plaintexts $m_0, m_1 \in R_t$. The challenger chooses a random bit $k \in \{0, 1\}$, generates the challenge public key $pk^* = ([a^*]_{q_{L-1}}, [b^*]_{q_{L-1}})$, which has not been queried before. Then, $\mathcal{A}$ runs $FHE.Encrypt$ (pk^*, m_k), outputs the challenge ciphertext
$$c^* = ([m_k + b^* \cdot u^* + t \cdot e_1^*]_{q_{L-1}}, [a^* \cdot u^* + t \cdot e_2^*]_{q_{L-1}}) \leftarrow FHE.Enc(pk^*, m_k),$$
where u^*, e_1^*, e_2^* are randomly sampled from χ.
- *Queries* 2: It is the same as *Queries* 1, the challenger queries $FHE.KeyGen(params)$, and obtains (sk_j, pk_j), but pk^* can't be queried before, where $j = t+1, \ldots, r$, r is the max number of query.
- *Output*: $\mathcal{A}$ outputs a guess $k' \in \{0, 1\}$. If $\mathcal{A}$ guesses right, outputs 1, else outputs 0. □

In order to prove the security of our scheme, we need to build a distinguisher D, namely
$$(a^*, b^*) \text{ and } Unif(R_{q_{L-1}} \times R_{q_{L-1}}).$$

D takes (a^*, b^*) and $c \in Unif(R_{q_{L-1}} \times R_{q_{L-1}})$ as inputs. When $\mathcal{A}$ gets plaintexts $m_0, m_1 \in R_t$, D chooses a random bit $k \in \{0, 1\}$, then generates the challenge ciphertext $c_k = ([m_k + b^* \cdot u^* + t \cdot e_1^*]_{q_{L-1}}, [a^* \cdot u^* + t \cdot e_2^*]_{q_{L-1}})$.

Suppose D has the advantage ϵ to distinguish ciphertexts c_0 and c_1, which plaintexts are m_1 and m_0, respectively. Then, D has the advantage ϵ to distinguish (a^*, b^*) and c, namely, D could solve the $RLWE_{\lambda,q_{L-1},\chi}$ assumption successfully. Because $\mathcal{A}$'s probability of distinguishing the challenge ciphertext is negligible, the proposed FHE scheme could resist chosen plaintext attack.

E. PARAMETER SETTING

Given a ciphertext c with modulus q_j. Let e_c be the decryption noise of c. Let f be the function of homomorphic operations. After the operation of f on c, the decryption noise will become e_{c_f}. For the correctness of c's decryption, we need to set $|e_{c_f}| < q_j/2$.

The security of our proposed FHE scheme relies on the RLWE assumption. The difficulty of the RLWE assumption is decided by the security parameter λ, parameter m, prime modulus q_j. In order to guarantee our scheme's λ, we need to set $phi(m) > \log_2(q_j) \cdot (\lambda + 110)/7.2$[37], where $phi(m)$ represents the dimension of $\Phi_m(x)$. For a fixed λ, q_j will increase with the growing of m. For example, $\lambda = 80$, if $phi(m) = 1176$, then $\log_2(q_j) = 44$, namely $m = 1247$, $q_j = 2^{44}$; if $phi(m) = 2880$, then $\log_2(q_j) = 109$, namely $m = 3133$, $q_j = 2^{109}$.

IV. BUILDING BLOCKS

A. SIMD

Most of FHE schemes' plaintext space is 2. If the plaintext space is large, time of encoding, encryption and decryption will become too long. In order to improve the efficiency of homomorphic operations, we need to encrypt multiple small plaintexts into a ciphertext by SIMD [21].

SIMD can be realized by Chinese Remainder Theorem. For example, suppose the dimension of polynomial ring is n, the plaintext modulus is t. Assume $x^n + 1 = [\prod_{i=1}^{k} Q_i(x)]_t$, where $Q_1(x), Q_2(x), \ldots, Q_k(x)$ are coprime. The isomorphism $\frac{\mathbb{Z}_t[x]}{x^n+1} \cong \prod_{i=1}^{k} \frac{\mathbb{Z}_t[x]}{Q_i(x)}$ is constructed by Chinese Remainder Theorem. Then, k plaintexts in $\frac{\mathbb{Z}_t[x]}{Q_1(x)}, \frac{\mathbb{Z}_t[x]}{Q_2(x)}, \ldots, \frac{\mathbb{Z}_t[x]}{Q_k(x)}$ could be packed into a message in $\frac{\mathbb{Z}_t[x]}{x^n+1}$. Then, the message can be encrypted into a ciphertext. In most cases, the selection of k polynomials $Q_1(x), Q_2(x), \ldots, Q_k(x)$ is possible, and it satisfies $x^n + 1 = [\prod_{i=1}^{k} Q_i(x)]_t$, where $Q_i(x)$ is linear polynomial, $\frac{\mathbb{Z}_t[x]}{Q_i(x)} \cong \mathbb{Z}_t$, $\frac{\mathbb{Z}_t[x]}{x^n+1} \cong \mathbb{Z}_t^n$. More details about SIMD can be found in the Ref. [21].

B. ENCODING RATIONAL NUMBER

In the real machine learning classification, original training data and testing data are usually rational numbers. Presently, most of FHE schemes are unable to encode or decode rational numbers. Hence, we need to create a bridge between integer and rational number. The simplest way is to convert the rational number to an integer by multiplying a fixed power of B $(B \geq 2)$[38], for example B^2, which will not affect the rational number's precision. Then, rational number can be encrypted and operated homomorphically. Finally, the decryption result should be divided by an appropriate amount, for example B^2. However, this method has the disadvantage that each plaintext needs to be scaled down after every multiplicative homomorphic. In order to avoid this

```
$comp(c_0, c_1)$ :
Input: Ciphertexts $c_0$ and $c_1$.
Output: $c_b$.
1. Compute $c_b = c_t +_h c_0 -_h c_1$.
2. Return $c_b$ to the user.
```

FIGURE 2. **Homomorphic comparison protocol.**

disadvantage, we take the technique of fractional encoder [39], which is described as follows.

In Bos's technique of fractional encoder [39], first the rational number should be decomposed into $b_{n_1} \dots b_2 b_1 b_0 . b_{-1} b_{-2} \dots b_{-n_2}$ with an integer base $k \geq 2$, $b_{n_1} \dots b_2 b_1 b_0$ are decomposed bits of its integral part, n_1 represents the most significant decomposed bit of its integral part, $b_{-1} b_{-2} \dots b_{-n_2}$ are decomposed bits of its fractional part, n_2 represents the most significant decomposed bit of its fractional part. The rational number's encoding formula is $\sum_{0 \leq i \leq n_1} x^i \cdot b_i - \sum_{0 < i \leq n_2} x^{n-i} b_{-i}$, where n represents the dimension of polynomial ring. For example, $3.5 = (11.1)_2$ is encoded as $x + 1 - x^{n-1}$, $1.25 = (1.01)_2$ is encoded as $1 - x^{n-2}$, where the integer base is 2. The multiplication between $x + 1 - x^{n-1}$ and $1 - x^{n-2}$ is operated as follows:

$$\begin{aligned}
&[(x + 1 - x^{n-1}) \times (1 - x^{n-2})]_{x^n+1} \\
&\quad = [x + 1 - x^{n-1} - x^{n-1} - x^{n-2} + x^{2n-3}]_{x^n+1} \\
&\quad = [-2x^{n-1} - x^{n-2} - x^{n-3} + x + 1]_{x^n+1} \\
&\quad = [1 - x^{n-2} - x^{n-3} + x + 1]_{x^n+1} \\
&\quad = [-x^{n-2} - x^{n-3} + x + x]_{x^n+1} \\
&\quad = -x^{n-2} - x^{n-3} + x^2.
\end{aligned}$$

The decoding of $-x^{n-2} - x^{n-3} + x^2$ is $(100.011)_2$, which equals to $(11.1)_2 \times (1.01)_2 = (100.011)_2$.

C. HOMOMORPHIC COMPARISON PROTOCOL

In this section, we begin to describe the homomorphic comparison protocol for private machine learning classification. Given c_0, c_1 and c_t, which are ciphertexts of plaintext m_0 ,m_1 and t encrypted by our FHE scheme, respectively, where t is the plaintext modulus. Then, c_0 and c_1 are stored on the cloud server. Only the user has the secret key. Bost's comparison protocol [14] is realized by Quadratic Residuosity cryptosystem [19] and Paillier cryptosystem [16]. In our homomorphic comparison protocol (Figure 2), the cloud server first computes the comparison ciphertext $c_b = c_t +_h c_0 -_h c_1$ without decryption. Then, c_b is returned to the user. The user gets the decryption result b by his secret key. The comparison result is b's lth bit b_l, where $l = \log_2 t + 1$. If $b_l = 1$, it means $m_0 \geq m_1$; otherwise $m_0 < m_1$. For example, let $t = 2$, $m_0 = 0$, $m_1 = 1$, then $l = 2$, $c_b = c_2 +_h c_0 -_h c_1$. After the decryption of c_b, the lth bit of b is 0, hence $m_0 < m_1$.

In Bost's homomorphic comparison protocol [14], adopted cryptosystems only support additive homomorphic. Hence, its multiplicative homomorphic needs to be realized by several interactions between the user and the cloud server. For the scheme's security, Bost used additional random number r to blind b. Because FHE supports both additive homomorphic and multiplicative homomorphic, we reduce the number of interactions from 3 to 1.

```
$argmax(c_0, c_1, \cdots, c_{k-1})$:
Input: Ciphertexts $c_0, c_1, \cdots, c_{k-1}$.
Output: $c_{max}$.
1. Set $max = 0$.
2. For $i = 1$ to $k - 1$ do
3. $c_b = comp(c_i, c_{max})$.
4. The user decrypts $c_b$ to get $b$.
5. If $b_l = 1$
$c_{max} = c_i$.
End for
```

FIGURE 3. **Homomorphic maximum protocol.**

D. HOMOMORPHIC MAXIMUM PROTOCOL

In this section, we begin to describe the homomorphic maximum protocol for private machine learning classification. Supposed the user owns k ciphertexts $(c_i)_{i=0,1,\dots,k-1}$ of plaintexts $(m_i)_{i=0,1,\dots,k-1}$, respectively, which are stored on the cloud server. The user owns the secret key. In this situation, the user could know the index of the largest plaintext without decryption, which will not leak user privacy. Our homomorphic maximum protocol is described as follows. Let max denote the index of the largest plaintext, initial $max = 0$. Then, the cloud server compares c_{max} with c_i by our homomorphic comparison protocol, where $i = 1, 2, \dots, k - 1$. If $b_l = 1$, it means $c_{max} \leq c_i$, set $c_{max} = c_i$; otherwise the cloud server needs to compare c_{max} and c_{i+1} until $i + 1 > k - 1$. After all these iterations, the user will get c_{max} without leaking user privacy. Detailed procedures are shown in Figure 3.

V. PRIVATE MACHINE LEARNING CLASSIFICATION

In this section, we begin to describe how to implement private machine learning classifications, which include private hyperplane decision-based classification, private Naïve Bayes classification and private decision tree classification.

A. PRIVATE HYPERPLANE DECISION-BASED CLASSIFICATION

Supposed the user has weights $(w_i)_{i=0,1,\dots,k-1}$ and input data x, where $w_i \in \mathbb{R}^n$. Based on our FHE scheme, $(w_i)_{i=0,1,\dots,k-1}$ and x are encrypted as $(c_{w_i})_{i=0,1,\dots,k-1}$ and c_x, respectively, where $x = (x_0, x_1, \dots, x_{n-1}) \in \mathbb{Z}^n$. Then, $(c_{w_i})_{i=0,1,\dots,k-1}$ and c_x are stored on the cloud server. The user owns the secret key. In the private hyperplane decision-based classification, the cloud server calculates the classification result without decryption, namely

$$c_{k^*} = argmax_{i \in [0,k)} \langle c_{w_i}, c_x \rangle,$$

which will not leak user privacy. The private hyperplane decision-based classification works as follows. First, the cloud server calculates $c_{v_i} = \langle c_{w_i}, c_x \rangle$, where $i = 0, 1, \dots, k - 1$.

```
Input: Ciphertexts $(c_{w_i})_{i=0,1,\cdots,k-1}$, $c_x$.
Output: The encrypted classification result $c_{k^*}$.
1. For $i = 0$ to $k-1$ do
    Execute $c_{v_i} = \langle c_{w_i}, c_x \rangle$.
   End for
2. Execute $c_{k^*} = argmax(c_{v_0}, c_{v_1}, \cdots, c_{v_{k-1}})$.
```

FIGURE 4. Private hyperplane decision classification.

Then, the cloud server computes $c_{k^*} = argmax_{i\in[0,k)} c_{v_i}$. Finally, the cloud server sends c_{k^*} to the user. The user obtains k^* by his secret key. Detailed procedures are shown in Figure 4.

In order to speed up homomorphic operations, we use SIMD. Let c_p denote the ciphertext of k small plaintexts $(w_i)_{i=0,1,\ldots,k-1}$. Then, the cloud server calculates $c_v = c_p \times_h c_x$. The user gets the decryption result $(v_i)_{i=0,1,\ldots,k-1}$ of c_v. The classification result k^* can be easily obtained by comparison.

Bost's private hyperplane decision-based classification [14] is implemented by Quadratic Residuosity cryptosystem [19] and Paillier cryptosystem [16]. However, it needs too many interactions and ciphertext transformation. In our private hyperplane decision-based classification, the homomorphic maximum protocol is implemented by FHE, so it will eliminate excessive interactions and ciphertext transformation. Then, SIMD is used to speed up homomorphic operations.

B. PRIVATE NAÏVE BAYES CLASSIFICATION

Suppose the user has probabilities $(p(x_j = a|C = c_i))_{i=0,1,\ldots,k-1}$, conditional probabilities $(p(x_j = a|C = c_i))_{i=0,1,\ldots,k-1}$, where c_i is the class, x_j is the jth component of input data x, $a \in D_j$, D_j belongs to x's domain, where $j = 0, 1, \ldots, n-1$. Based on our FHE scheme, $(p(x_j = a|C = c_i))_{i=0,1,\ldots,k-1}$ and $(p(x_j = a|C = c_i))_{i=0,1,\ldots,k-1}$ are encrypted as $cp_{p(C=c_i)}$ and $cp_{p(x_j=a|C=c_i)}$, respectively. Then, $(p(x_j = a|C = c_i))_{i=0,1,\ldots,k-1}$ and $(p(x_j = a|C = c_i))_{i=0,1,\ldots,k-1}$ are stored on the cloud server. The user owns the secret key. In the private Naïve Bayes classification, the cloud server's goal is to calculate c_{k^*} in a privacy preserving way, namely

$$c_{k^*} = argmax_{i\in[0,k)} cp_{p(C=c_i)} \times_h \prod_{j=0}^{n-1} cp_{p(x_j=a|C=c_i)}.$$

Based on above building blocks, the private Naïve Bayes classification is implemented as follows. First, the cloud server calculates $cp_{r_i} = cp_{p(C=c_i)} \times_h \prod_{j=0}^{n-1} cp_{p(x_j=a|C=c_i)}$, where $i = 0, 1, \ldots, k-1$, $j = 0, 1, \ldots, n-1$. Then the cloud server computes $k^* = argmax_{i\in[0,k)} cp_{r_i}$ by the homomorphic maximum protocol. Finally, the cloud server sends c_{k^*} to the user. The user obtains k^* by his secret key. Detailed procedures are shown in Figure 5.

We use SIMD to speed up homomorphic operations, which is described as follows. First, every k plaintexts $(p(x_j = a|C = c_i))_{i=0,1,\ldots,k-1}$ are encrypted as a ciphertext cp_{p_j}, where $j = 0, 1, \ldots, n-1$. Let cp_k denote the ciphertext of k plaintexts $(p(C = c_i))_{i=0,1,\ldots,k-1}$. Then, the cloud server

```
Input: Ciphertexts $(p(x_j = a|C = c_i))_{i=0,1,\cdots,k-1}$,
    $(p(x_j = a|C = c_i))_{i=0,1,\cdots,k-1}$, $j = 0, 1, \cdots, n-1$.
Output: The encrypted classification result $c_{k^*}$.
1. For $i = 0$ to $k-1$ do
    Execute $cp_{r_i} = cp_{p(C=c_i)} \times_h \prod_{j=0}^{n-1} cp_{p(x_j=a|C=c_i)}$.
   End for
2. Execute $c_{k^*} = argmax(cp_{r_0}, cp_{r_1}, \cdots, cp_{r_{k-1}})$.
```

FIGURE 5. Private Naïve Bayes classification.

computes $cp_r = cp_k \times_h \prod_{j=0}^{n-1} cp_{p_j}$. The user obtains the decryption result $(r_i)_{i=0,1,\ldots,k-1}$ from cp_r by his secret key. Finally, the classification result k^* can be easily obtained by comparison. SIMD helps us to reduce the number of multiplicative homomorphic from kn to n.

In our private Naïve Bayes classification, the homomorphic maximum protocol is implemented by the proposed FHE scheme. Compared with Bost's private Naïve Bayes classification [14], we eliminate excessive interactions and ciphertext transformation.

C. PRIVATE DECISION TREE CLASSIFICATION

Suppose the user has input data $x = (x_i)_{i=0,1,\ldots,n-1}$ and weights $(w_i)_{i=0,1,\ldots,k-1}$, they are encrypted as c_x and $(c_{w_i})_{i=0,1,\ldots,k-1}$, respectively. Let l'-bit $c_j = (c_{j,i})_{i=0,1,\ldots,l'-1}$ denote the class. Let $(C_{c_{j,i}})_{i=0,1,\ldots,l'-1}$ represent ciphertexts of $(c_{j,i})_{i=0,1,\ldots,l'-1}$, where $c_{j,i}$ represents the ith element of c_j, $j = 0, 1, \ldots, k-1$. Then, c_x, $(c_{w_i})_{i=0,1,\ldots,k-1}$ and $(C_{c_{j,i}})_{i=0,1,\ldots,l'-1}$ are stored on the cloud server. The user owns the secret key.

In the private decision tree classification, the goal for the cloud server is to calculate the classification result $k^* \in [0, k)$ privately. This classification can be implemented securely by above building blocks, which works as follows. First, the cloud server compares c_x with c_{w_i} by the homomorphic comparison protocol, where $i = 0, 1, \ldots, k-1$. Let b_i denote the comparison result. Let c_{b_i} denote the ciphertext of b_i. The user returns c_{b_i} to the cloud server. Then, the cloud server calculates $(c_{P_i})_{i=0,1,\ldots,l'-1}$ homomorphically. For example, in the Figure 1, $c_{P_i} = c_{b_0} \times_h (c_{b_2} \times_h C_{c_{5,i}} +_h (1 -_h c_{b_2}) \times_h (1 -_h c_{b_4}) \times_h C_{c_{3,i}} +_h (1 -_h c_{b_2}) \times_h c_{b_4} \times_h C_{c_{4_i}}) +_h (1 -_h c_{b_0}) \times_h (c_{b_1} \times_h C_{c_{2,i}} +_h (1 -_h c_{b_1}) \times_h c_{b_3} \times_h C_{c_{1,i}} +_h (1 - c_{b_1}) \times_h (1 -_h c_{b_3}) \times_h C_{c_{0,i}})$, where $(c_{b_j})_{j=0,1,\ldots,4}$ represent ciphertexts of $(b_j)_{j=0,1,\ldots,4}$ respectively, $i = 0, 1, \ldots, l'-1$. Finally, $c_{k^*} = \sum_{i=0}^{l'-1} c_{P_i} \times_h c_{2^i}$, where c_{2^i} is the ciphertext of 2^i. The user obtains k^* by his secret key. Detailed procedures are shown in Figure 6.

In order to speed up homomorphic operations, we use SIMD. l' plaintexts $(c_{j,i})_{i=0,1,\ldots,l'-1}$ are encrypted as a ciphertext C_{c_j}, where $j = 0, 1, \ldots, k-1$. Then, the cloud server calculates c_P homomorphically. For example, in the Figure 1, the cloud server computes $c_P = c_{b_0} \times_h (c_{b_2} \times_h C_{c_5} +_h (1 -_h c_{b_2}) \times_h (1 -_h c_{b_4}) \times_h C_{c_3} +_h (1 -_h c_{b_2}) \times_h c_{b_4} \times_h C_{c_4}) +_h (1 -_h c_{b_0})(c_{b_1} \times_h C_{c_2} +_h (1 -_h c_{b_1}) \times_h c_{b_3} \times_h C_{c_1} +_h (1 -_h c_{b_1}) \times_h (1 -_h c_{b_3}) \times_h C_{c_0})$, where $(C_{c_j})_{j=0,1,\ldots,5}$ are

```
Input: Ciphertexts $c_x$, $(c_{w_i})_{i=0,1,\cdots,k-1}$, $(C_{c_j})_{j=0,1,\cdots,k-1}$.
Output: The encrypted classification result $c_{k^*}$.
1. For $i = 0$ to $k-1$ do
   Execute $c_{b_i} = comp(c_x, c_{w_i})$.
   End for
2. Calculate $(c_{P_i})_{i=0,1,\cdots,l'}$ by $(c_{b_i})_{i=0,1,\cdots,k-1}$ and $(C_{c_j})_{j=0,1,\cdots,k-1}$.
3. Compute $c_{k^*} = \sum_{i=0}^{l'} c_{P_i} \times_h c_{2^i}$.
```

FIGURE 6. Private decision tree classification.

TABLE 1. Implementation time of 10 homomorphic operations with multiplicative depth 1 among our FHE scheme with SIMD, HElib with SIMD and Khedr's FHE scheme (unit: microsecond).

Security parameter λ	5	50	53	55
Our FHE scheme with SIMD	0.54	0.77	0.78	0.84
HElib with SIMD	31.33	40.28	89.89	98.77
Khedr's FHE scheme	72.60	236.46	594.86	790.26

ciphertexts of $(c_{j,i})_{i=0,1,\ldots,l'-1, j=0,1,\ldots,5}$, respectively. The user obtains $(P_i)_{i=0,1,\ldots,l'-1}$ by his secret key. At last, $k^* = \sum_{i=0}^{l'-1} P_i \cdot 2^i$.

In Bost's private decision tree classification[14], $(c_{b_i})_{i=0,1,\ldots,k-1}$ are implemented by additive homomorphic encryption schemes [16], [19], $(c_{P_i})_{i=0,1,\ldots,l'-1}$ are implemented by the FHE library HElib [22]. Based on Gentry's FHE scheme [25], Khedr *et al.* [15] first proposed an improved FHE scheme, then it is used to implement private decision tree classification. However, Khedr's improved FHE scheme doesn't support SIMD. We calculate $(c_{b_i})_{i=0,1,\ldots,k-1}$ and $(c_{P_i})_{i=0,1,\ldots,l'-1}$ by our FHE scheme with SIMD, which could reduce the number of interactions in the computation of $(c_{b_i})_{i=0,1,\ldots,k-1}$, and improve the efficiency in the computation of $(c_{P_i})_{i=0,1,\ldots,l'-1}$.

VI. EXPERIMENT RESULTS

In this section, we first compare the efficiency of multiplicative homomorphic among our FHE scheme with SIMD, the FHE library HElib [22] with SIMD and Khedr's FHE scheme [15]. Then, we analyze the implementation time of our homomorphic comparison protocol, private hyperplane decision-based classification and private Naïve Bayes classification. Finally, we compare the implementation time of private decision tree classification among our FHE scheme with SIMD, the FHE library HElib [22] with SIMD and Khedr's FHE scheme [15]. Experiments are carried out on the same personal computer's virtual machine with no GPU hardware platform, the experimental environment is set as follows: the operating system is microsoft windows 7, featuring two Intel (R) Core (TM) i5-3470 CPU processors, running at 3.20 GHz, with 8.00 GB RAM, and the virtual machine's operation system is Ubuntu 12.04, featuring single Intel (R) Core (TM) i5-3470 CPU processor, with 3.70 GB RAM. Our

TABLE 2. Implementation time of 10 homomorphic operations with multiplicative depth 2 among our FHE scheme with SIMD, HElib with SIMD and Khedr's FHE scheme (Unit: microsecond).

Security parameter λ	5	50	53	55
Our FHE scheme with SIMD	7.72	10.73	21.63	23.41
HElib with SIMD	77.75	84.20	170.56	214.66
Khedr's FHE scheme	174.95	493.07	923.44	2310.10

TABLE 3. Implementation time of 10 homomorphic operations with multiplicative depth 3 among our FHE scheme with SIMD, HElib with SIMD and khedr's FHE scheme (unit: microsecond).

Security parameter λ	5	50	53	55
Our FHE scheme with SIMD	48.30	55.58	116.84	132.62
HElib with SIMD	111.88	123.59	253.54	316.24
Khedr's FHE scheme	181.39	592.97	1208.11	3107.92

TABLE 4. Implementation time of our homomorphic comparison protocol (unit: microsecond).

Security parameter λ	5	50	53	55
Our FHE scheme	0.08	0.10	0.11	0.13

implementation uses NTL large number library for high level numeric algorithms. The experimental code is compiled on GCC platform by the C++ language. Private machine learning classifications' models are trained by machine learning module scikit-learn [40]. We adopt Wisconsin Breast Cancer (Original) data set from the UCI machine learning repository for experiments. For convenience, we set λ ranging from 5 to 55, $k = 4$, $n = 3$. The number and depth of multiplicative homomorphic are key factors of private machine learning classification's implementation time. Depths of multiplicative homomorphic in priavte hyperplane decision-based classification, private Naïve Bayes classification and private decision tree classification are 1, 3 and 3, respectively.

The implementation time of 10 homomorphic operations with multiplicative depths 1, 2 and 3 among our FHE scheme with SIMD, HElib [22] with SIMD and Khedr's FHE scheme [15] is shown in Tables 1, 2 and 3, respectively. Because Khedr's FHE scheme [15] can't support SIMD, we implement Khedr's FHE scheme's 10 homomorphic operations successively. The implementation time of our homomorphic comparison protocol is shown in Table 4. The implementation time of private hyperplane decision-based classification and private Naïve Bayes classification between our FHE scheme and our FHE scheme with SIMD is shown in Tables 5 and 6, respectively. Bost's [14] homomorphic comparison protocol, private hyperplane decision classification and private Naïve Bayes classification are implemented by Quadratic Residuosity cryptosystem [19] and Paillier cryptosystem [16], which are additive homomorphic encryption schemes. Our homomorphic comparison protocol, private hyperplane decision-based classification and private Naïve Bayes classification are first implemented by multiplicative homomorphic and additive

TABLE 5. Implementation time of private hyperplane decision-based classification between our FHE scheme and our FHE scheme with SIMD (unit: microsecond).

Security parameter λ	5	50	53	55
Our FHE scheme	5.62	7.91	8.17	9.22
Our FHE scheme with SIMD	1.32	1.98	2.07	2.41

TABLE 6. Implementation time of private Naïve Bayes classification between our FHE scheme and our FHE scheme with SIMD (unit: microsecond).

Security parameter λ	5	50	53	55
Our FHE scheme	192.78	225.54	468.42	568.07
Our FHE scheme with SIMD	48.79	55.41	119.40	141.95

TABLE 7. Implementation time of private decision tree classification among our FHE scheme with SIMD, HElib with SIMD and Khedr's FHE scheme (unit: microsecond).

Security parameter λ	5	50	53	55
Our FHE scheme with SIMD	61.39	74.85	161.26	177.06
HElib with SIMD	209.36	244.53	517.88	1788.60
Khedr's FHE scheme	237.65	271.49	782.97	2329.26

homomorphic. So, we don't compare our homomorphic comparison protocol, private hyperplane decision classification and private Naïve Bayes classification with other schemes. The implementation time of private decision tree classification among our FHE scheme with SIMD, HElib [22] with SIMD and Khedr's FHE scheme [15] is shown in Table 7. Each test has three iterations and data shown in the tables are averages of them. As seen from Tables 1, 2, and 3, the implementation time of 10 homomorphic operations by our FHE scheme with SIMD is significantly lower than that of HElib [22] with SIMD and Khedr's FHE scheme [15]. Observed from Tables 5 and 6, the implementation time of private hyperplane decision-based classification and private Naïve Bayes classification by our FHE scheme with SIMD is reduced several magnitudes compared with our FHE scheme. The efficiency of private decision tree classification by our FHE scheme with SIMD is better than that of HElib [22] with SIMD and Khedr's FHE scheme [15]. The detailed analysis is described as follows.

Figures 7, 8 and 9 show the efficiency of 10 homomorphic operations among our FHE scheme with SIMD, HElib [22] with SIMD and Khedr's FHE scheme [15] with multiplicative depths 1, 2 and 3, respectively. Figures 10 and 11 show the efficiency of private hyperplane decision-based classification and private Naïve Bayes classification between our scheme and our FHE scheme with SIMD, respectively. Figure 12 shows the efficiency of private decision tree classification among our FHE scheme with SIMD, HElib [22] with SIMD and Khedr's FHE scheme [15]. Simultaneously, above six figures also indicate changing trends of implementation time with the increasing of λ. As seen from Figures 7, 8, and 9, the efficiency of 10 homomorphic operations by

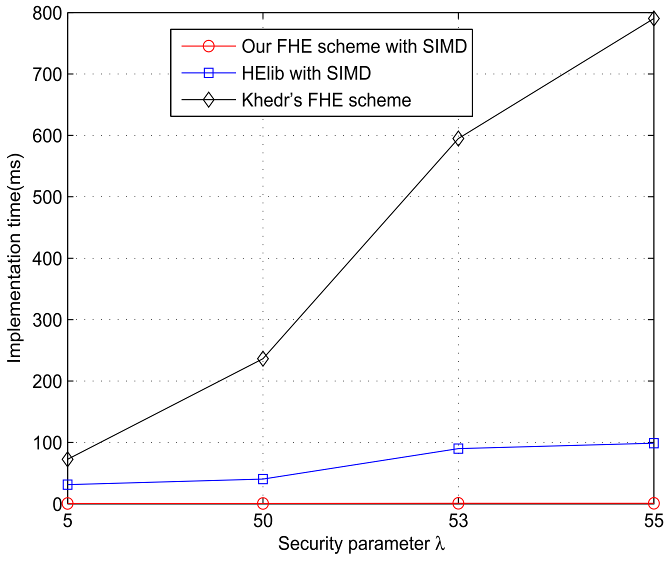


FIGURE 7. Efficiency comparison of 10 homomorphic operations with multiplicative depth 1 among our FHE scheme with SIMD, HElib with SIMD and Khedr's FHE scheme.

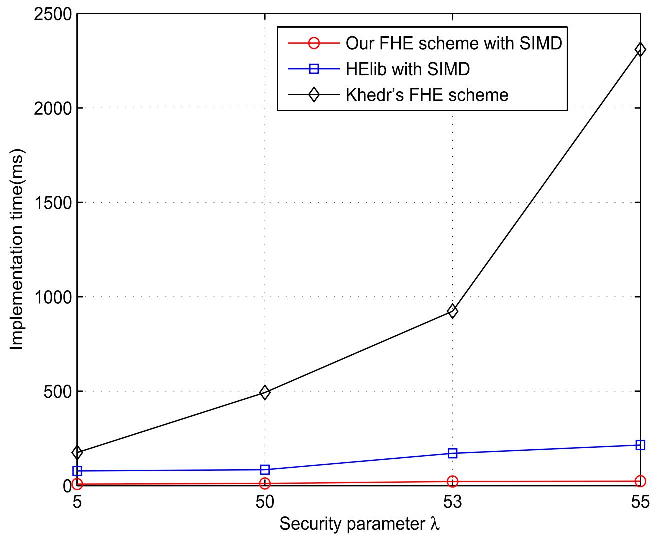


FIGURE 8. Efficiency comparison of 10 homomorphic operations with multiplicative depth 2 among our FHE scheme with SIMD, HElib with SIMD and Khedr's FHE scheme.

our FHE scheme with SIMD is more efficient than that of HElib [22] with SIMD and Khedr's FHE scheme [15] with the increasing of λ. From Figures 10 and 11, it can be easily known that the efficiency of the private hyperplane decision-based classification and the private Naïve Bayes classification by our FHE scheme with SIMD is incomparable to our scheme. And Figure 12 shows that the increasing tendency of the private decision tree classification time by our FHE scheme with SIMD is slower than that of HElib [22] with SIMD and Khedr's FHE scheme [15].

VII. CONCLUSION

In this paper, we propose an improved FHE scheme. For the efficiency of multiplicative homomorphic, we first use the

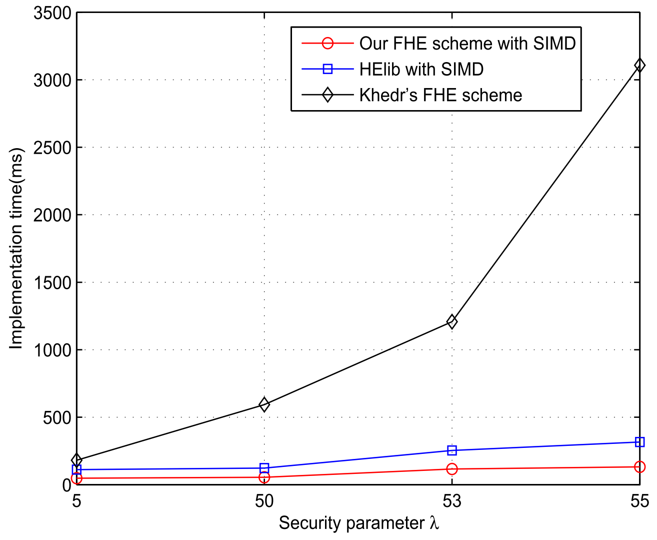


FIGURE 9. Efficiency comparison of 10 homomorphic operations with multiplicative depth 3 among our FHE scheme with SIMD, HElib with SIMD and Khedr's FHE scheme.

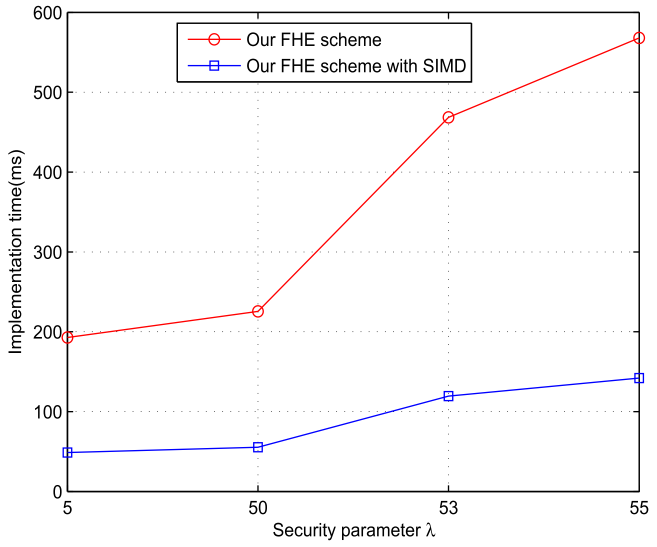


FIGURE 11. Efficiency comparison of private Naïve Bayes classification between our FHE scheme and our FHE scheme with SIMD.

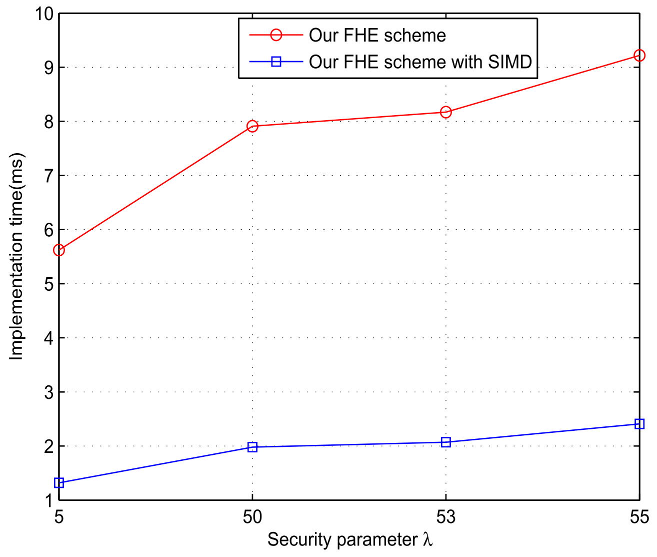


FIGURE 10. Efficiency comparison of private hyperplane decision-based classification between our FHE scheme and our FHE scheme with SIMD.

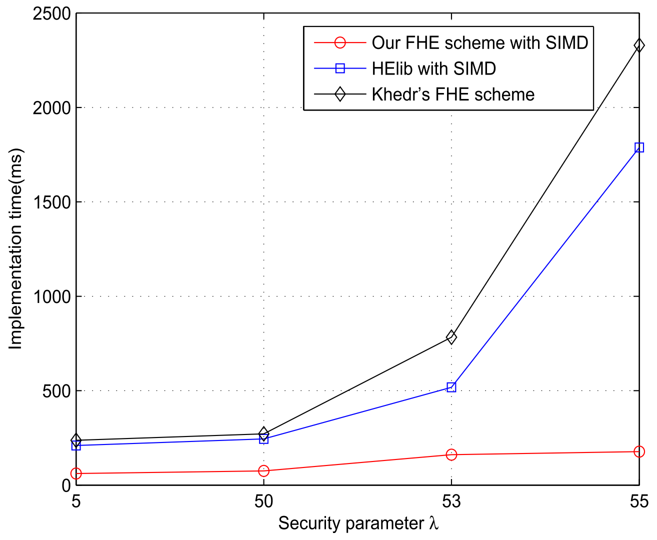


FIGURE 12. Efficiency comparison of private decision tree classification among our FHE scheme with SIMD, HElib with SIMD and Khedr's FHE scheme.

relinearization technique to reduce the ciphertext size, then the modulus switching technique is used for decreasing the modulus and decryption noise. If there is additive homomorphic or no homomorphic operation in the multiplicative ciphertext's next homomorphic operation, we need no relinearization and modulus switching. Simulation results show that the efficiency of our FHE scheme with SIMD is more efficient than the FHE library HElib with SIMD and Khedr's FHE scheme. In our scheme, private hyperplane decision-based classification, private Naïve Bayes classification and private decision tree's comparison are implemented by FHE first, which could eliminate excessive interactions between the user and the cloud server. The speed of homomorphic operations is optimized by SIMD. To implement private decision tree classification efficiently, we utilize the proposed FHE scheme. And our implementation of private decision tree classification is obviously better than that of the FHE library HElib with SIMD and Khedr's FHE scheme.

ACKNOWLEDGMENTS

We would like to thank Amin Sakzad for his valuable comments. This work was supported by the National Natural Science Foundation of China (61702342, 61702341, 61602316), the Science and Technology Innovation Projects of Shenzhen (JCYJ20160307150216309, JCYJ20170302151321095) and Tencent Rhinoceros Birds-Scientific Research Foundation for Young Teachers of Shenzhen University.

REFERENCES

[1] X. Sun, P. Zhang, M. Sookhak, J. Yu, and W. Xie, "Utilizing fully homomorphic encryption to implement secure medical computation in smart cities," *Pers. Ubiquitous Comput.*, vol. 21, no. 5, pp. 831–839, 2017.

[2] A. Castiglione, C. DAmbrosio, A. De Santis, A. Castiglione, and F. Palmieri, "On secure data management in health-care environment," in *Proc. 7th Int. Conf. Innovative Mobile Internet Serv. Ubiquitous Comput.*, 2013, pp. 666–671.

[3] Q. Zou, J. Wang, J. Ye, J. Shen, and X. Chen, "Efficient and secure encrypted image search in mobile cloud computing," *Soft Comput.*, vol. 21, no. 11, pp. 2959–2969, 2017.

[4] C. Xiang, C. Tang, Y. Cai, and Q. Xu, "Privacy-preserving face recognition with outsourced computation," *Soft Comput.*, vol. 20, no. 9, pp. 3735–3744, 2016.

[5] P. Li, T. Li, Z.-A. Yao, C.-M. Tang, and J. Li, "Privacy-preserving outsourcing of image feature extraction in cloud computing," *Soft Comput.*, vol. 21, no. 15, pp. 4349–4359, 2017.

[6] C. Wang, B. Zhang, K. Ren, and J. M. Roveda, "Privacy-assured outsourcing of image reconstruction service in cloud," *IEEE Trans. Emerging Topics Comput.*, vol. 1, no. 1, pp. 166–177, Jun. 2013.

[7] X. Sun, T. Wang, Z. Sun, P. Wang, J. Yu, and W. Xie, "An efficient quantum somewhat homomorphic symmetric searchable encryption," *Int. J. Theoretical Physics*, vol. 56, no. 4, pp. 1335–1345, 2017.

[8] J. Mao, Y. Zhang, P. Li, T. Li, Q. Wu, and J. Liu, "A position-aware Merkle tree for dynamic cloud data integrity verification," *Soft Comput.*, vol. 21, no. 8, pp. 2151–2164, 2017.

[9] A. Castiglione, A. De Santis , B. Masucci, F. Palmieri, A. Castiglione, and X. Huang, "Cryptographic hierarchical access control for dynamic structures," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 10, pp. 2349–2364, Oct. 2016.

[10] A. Castiglione, *et al.*, "Hierarchical and shared access control," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 4, pp. 850–865, Apr. 2016.

[11] H. Li, D. Liu, Y. Dai, T. H. Luan, and X. S. Shen, "Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage," *IEEE Trans. Emerging Topics Comput.*, vol. 3, no. 1, pp. 127–138, Mar. 2015.

[12] D. Thilakanathan, S. Chen, S. Nepal, and R. Calvo, "SafeProtect: Controlled data sharing with user-defined policies in cloud-based collaborative environment," *IEEE Trans. Emerging Topics Comput.*, vol. 4, no. 2, pp. 301–315, Apr.–Jun. 2016.

[13] X. Liu, R. Lu, J. Ma, L. Chen, and B. Qin, "Privacy-preserving patient-centric clinical decision support system on Naive Bayesian classification," *IEEE J. Biomed. Health Informat.*, vol. 20, no. 2, pp. 655–668, Mar. 2016.

[14] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," *Cryptology ePrint Archive*, Report 2014/331, 2014, http://eprint.iacr.org/

[15] A. Khedr, G. Gulak, and V. Vaikuntanathan, "SHIELD: Scalable homomorphic implementation of encrypted data-classifiers," *IEEE Trans. Comput.*, vol. 65, no. 9, pp. 2848–2858, Sep. 2016.

[16] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1999, pp. 223–238.

[17] C. Bishop, "Pattern recognition and machine learning," *J. Electron. Imag.*, vol. 16, no. 4, pp. 140–155, 2006.

[18] B. K. Samanthula, Y. Elmehdwi, and W. Jiang, "k-Nearest neighbor classification over semantically secure encrypted relational data," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1261–1273, May 2015.

[19] S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial information," in *Proc. 14th Annu. ACM Symp. Theory Comput.*, 1982, pp. 365–377.

[20] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," in *Proc. 3rd Innovations Theoretical Comput. Sci. Conf.*, 2012, pp. 309–325.

[21] N. P. Smart and F. Vercauteren, "Fully homomorphic SIMD operations," *Designs Codes Cryptography*, vol. 71, no. 1, pp. 57–81, 2014.

[22] S. Halevi and V. Shoup, "Design and implementation of a homomorphic-encryption library," 2013. [Online]. Available: https://github.com/shaih/HElib

[23] S. Halevi and V. Shoup, "HElib-an implementation of homomorphic encryption," 2013. [Online]. Available: https://github.com/shaih/HElib

[24] S. Halevi and V. Shoup, "Algorithms in HElib," in *Proc. Int. Cryptology Conf.*, 2014, pp. 554–571.

[25] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Proc. Advances Cryptology–CRYPTO*, 2013, pp. 75–92.

[26] L. Jiang, C. Xu, X. Wang, and C. Lin, "Statistical learning based fully homomorphic encryption on encrypted data," *Soft Comput.*, vol. 21, pp. 7473–7483, 2017.

[27] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Found. Secure Comput.*, vol. 4, no. 11, pp. 169–180, 1978.

[28] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2009, pp. 169–169.

[29] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *Proc. IEEE 52nd Annu. Symp. Found. Comput. Sci.*, 2011, pp. 97–106.

[30] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the AES circuit," in *Proc. Advances Cryptology–CRYPTO*, 2012, pp. 850–867.

[31] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2010, pp. 1–23.

[32] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-LWE and security for key dependent messages," in *Proc. Annu. Cryptology Conf.*, 2011, pp. 505–524.

[33] L. Ducas and D. Micciancio, "FHEW: Bootstrapping homomorphic encryption in less than a second," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2015, pp. 617–640.

[34] J. Alperin-Sheriff and C. Peikert, "Faster bootstrapping with polynomial error," in *Proc. Int. Cryptology Conf.*, 2014, pp. 297–314.

[35] X. Sun, J. Yu, T. Wang, Z. Sun, and P. Zhang, "Efficient identity-based leveled fully homomorphic encryption from RLWE," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 5155–5165, 2016.

[36] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proc. 40th Annu. ACM Symp. Theory Comput.*, 2008, pp. 197–206.

[37] R. Lindner and C. Peikert, "Better key sizes (and attacks) for LWE-based encryption," in *Proc. 11th Int. Conf. Topics Cryptology: CT-RSA*, 2011, pp. 319–339.

[38] A. Jäschke and F. Armknecht, "Accelerating homomorphic computations on rational numbers," in *Proc. Int. Conf. Appl. Cryptography Netw. Secur.*, 2016, pp. 405–423.

[39] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig, "Improved security for a ring-based fully homomorphic encryption scheme," in *Proc. IMA Int. Conf. Cryptography Coding*, 2013, pp. 45–64.

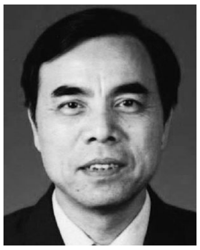
[40] F. Pedregosa, *et al.*, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

XIAOQIANG SUN received the BS degree in mathematics and applied mathematics from Fuyang Normal University, China. He is currently working toward the PhD degree in the ATR Key Laboratory of National Defense Technology, College of Information Engineering, Shenzhen University, China. His research interests include cryptography and information security.

PENG ZHANG received the MS and PhD degrees in signal and information processing from Shenzhen University, in 2008 and 2011, respectively. Now she is a lecturer of the College of Information Engineering, Shenzhen University. Her current research interests include cryptography, information, and network security. She has published more than 20 academic research papers.

JOSEPH K. LIU received the PhD degree in information engineering from the Chinese University of Hong Kong, China, in 2004. He is a senior lecturer in the Faculty of Information Technology, Monash University, Australia. His current technical focus is particularly cyber security in the cloud computing paradigm, smart city, lightweight security, and privacy enhanced technology. He has published more than 90 referred journal and conference papers and received the Best Paper Award from ESORICS 2014 and ESORICS 2015. He has served as the program chair of ProvSec 2007, 2014, ACISP 2016 and as the program committee of more than 35 international conferences.

JIANPING YU received the MS degree in communication and electronic system from Northwestern Polytechnical University, China, in 1992 and the PhD degree in communication and electronics from Xidian University, China, in 1995. He is a professor in the College of Information Engineering, Shenzhen University, China. His current research interests include cryptography, information and communication security, network security, internet of things, and cloud computing. He has published more than 100 academic research papers.



WEIXIN XIE is currently a professor and doctoral tutor in the College of Information Engineering, Shenzhen University, China. He is also the commissioner of the Chinese Institute of Electronics Signal Processing Branch. His main research interests include signal processing, intelligent information processing, radar target recognition, and THz-TDS signal and image analysis. He has authored more than 200 academic research papers.