Fully Homomorphic Encryption for Classification in Machine Learning

Seiko Arita
Graduate School of Information Security
Institute of Information Security, Japan
Email: arita@iisec.ac.jp

Shota Nakasato
Graduate School of Information Security
Institute of Information Security, Japan
Email: snakasato@snakasato.com

***Abstract*—Using fully homomorphic encryption scheme, we construct fully homomorphic encryption scheme $\mathsf{FHE4GT}$ that can homomorphically compute an encryption of the greater-than bit that indicates $x > x'$ or not, given two ciphertexts c and c' of x and x', respectively, without knowing the secret key. Then, we construct homomorphic classifier $\mathsf{homClassify}$ that can homomorphically classify a given encrypted data without decrypting it, using machine learned parameters.**

I. Introduction

Fully Homomorphic Encryption (FHE) scheme enables us to homomorphically compute an encrypted XORed bit $\mathsf{Enc}(b_1 \,\mathsf{XOR}\, b_2)$ and encrypted AND bit $\mathsf{Enc}(b_1 \,\mathsf{AND}\, b_2)$ of given encrypted bits $\mathsf{Enc}(b_1)$ and $\mathsf{Enc}(b_2)$ without knowing the secret key [7]. Since any function can be written using XOR and AND gates, this means that one can homomorphically compute any function of encrypted bits without knowing the secret key.

Practically, computation over bitwise encryptions is not efficient. It is a kind of "1-bit" processor. In schemes such as [3], [4], one can encrypt congruent integers (i.e., $x \bmod n$) and can homomorphically compute addition $\mathsf{Enc}(x_1 + x_2 \bmod n)$ and multiplication $\mathsf{Enc}(x_1 \times x_2 \bmod n)$ of encrypted congruent integers $\mathsf{Enc}(x_1)$ and $\mathsf{Enc}(x_2)$ without knowing the secret key.

Using the FHE schemes, many researchers consider about the privacy-preserving data mining for outsourced data such as encrypted genomic, medical, financial data and so on [10], [14], [5], [15], [13], [12].

In this paper, we propose a *homomorphic classifier* using machine learned parameters. Suppose some machine learned parameters are stored in some cloud servers, and clients want the cloud server to classify their own data using the machine learned parameters stored in the cloud. Here, if clients send their data to the server without encryption, they must worry about their data privacy. However, if clients encrypts their data, then the cloud server cannot do the classifying task against such encrypted data.

Our solution to the dilemma is to encrypt client's data using FHE schemes. Clients encrypt their own data using some FHE scheme and send the encrypted data to the cloud server. The cloud server can perform its classifying task homomorphically (i.e., without decryption of encrypted client's data) thanks to the homomorphic property of FHE, where the server can only know its encrypted classified result. The encrypted result will send back to the client, and only the client can decrypt the classifying result. By using such homomorphic classifier, we can utilize machine learned parameters in the cloud, keeping the privacy of client's data.

However, here is an important technical problem. Classifying algorithms require to compute greater-than relation of given data items. However, conventional FHEs are not able to compute greater-than relation of encrypted data homomorphically. Actually, conventional methods compute greater-than relation after decrypting [10] or by using the greater-than protocol [1].

Therefore, we construct fully homomorphic encryption scheme $\mathsf{FHE4GT}$ that can homomorphically compute an encryption of the greater-than bit that indicates $x > x'$ or not, given two ciphertexts c and c' of x and x', respectively, without knowing the secret key.

A. Our Contribution

1) FHE scheme $\mathsf{FHE4GT}$ *for Greater-Than relation:* At the first, we construct the FHE scheme $\mathsf{FHE4GT}$ that can homomorphically compute an encryption of greater-than bit that indicates $x > x'$ or not, given two ciphertexts c and c' of x and x', respectively, without knowing the secret key.

2) Homomorphic Classifier $\mathsf{homClassify}$*:* By using the $\mathsf{FHE4GT}$, we construct homomorphic classifier $\mathsf{homClassify}$ that can homomorphically classify a given encrypted data without decrypting it, using machine learned parameters.

Implementation and benchmark: We implemented our scheme using homomorphic encryption library HElib by Halevi and Shoup [11], which is based on the BGV scheme [3].

Organization: Section II recalls FHE scheme and its basic properties. Then, we treat the problem of homomorphic computation for greater-than relation in Section II-B. We construct the $\mathsf{FHE4GT}$ scheme of greater-than relation in Section III. And then, we construct the homomorphic classifier $\mathsf{homClassify}$ for binary classification (i.e. a machine learning algorithm like SVM[16]) in Section IV. Finally, we show our benchmark results in Section V.

II. Preliminaries

A. Homomorphic Encryption

A homomorphic encryption scheme is a quadruple $\mathsf{HE} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ of probabilistic polynomial time algorithms. Keygen generates a public key pk, a secret key sk and an evaluation key evk: $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) \leftarrow \mathsf{Keygen}(1^\lambda)$. Enc encrypts a plaintext $x \in \mathbb{Z}_n$ to a ciphertext c under a public key pk: $c \leftarrow \mathsf{Enc}(\mathsf{pk}, x)$. Dec decrypts a ciphertext c to a plaintext x by the secret key sk: $x \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$. Eval applies a function $f : \mathbb{Z}_n^l \to \mathbb{Z}_n$ to given ciphertexts $c_1, \ldots, c_l$ and outputs a ciphertext c_f using the evaluation key evk : $c_f \leftarrow \mathsf{Eval}(\mathsf{evk}, f, c_1, \ldots, c_l)$.

A homomorphic encryption scheme HE is called L-*homomorphic* for $L = L(n)$ if for any function $f : \mathbb{Z}_n^l \to \mathbb{Z}_n$ given as a circuit of depth L and for any l bits $x_1, \ldots, x_l$, it holds that $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Eval}_{\mathsf{evk}}(f, c_1, \ldots, c_l)) = f(x_1, \ldots, x_l)$ for $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(x_i)$ $(i = 1, \ldots, l)$ except with a negligible probability. A homomorphic encryption scheme is called *fully homomorphic encryption* (FHE) scheme if it is L-homomorphic for any polynomial function $L = poly(n)$.

B. Greater-than relation

We consider the problem of comparison of magnitude of two encrypted numbers. Suppose we have two encrypted numbers $\mathsf{Enc}(x)$ and $\mathsf{Enc}(x')$. Define a bit b to be 1 if $x > x'$ and 0 otherwise. We want to compute an encryption $\mathsf{Enc}(b)$ of the bit b given only ciphertexts $\mathsf{Enc}(x)$ and $\mathsf{Enc}(x')$ without knowing the secret key. In the literature [13], [12] such problem is tackled by Greater-Than protocol based on (such as) the one given by Golle [9]. Their protocol is based on the fact that if $x > x'$, there exists a positive integer i such that $x = x' + i$.

III. The Proposed Scheme FHE4GT

In this section we construct an FHE scheme that can homomorphically compute the greater-than bit, using FHE schemes based on RLWE problem such as [2], [3], [6] as building blocks.

Parameters: The FHE4GT scheme is parameterized by three parameters q, r and t. The parameter q denotes ciphertext modulus. The parameter 2^r denotes plaintext modulus of FHE and the parameter t denotes bit-length of plaintext. Let $r = t + 1$. For $z \in \mathbb{Z}$, $[z]_q$ denotes the unique integer in $[-q/2, q/2)$ with $[z]_q \equiv z \bmod q$.

Scheme Description: Let $\mathsf{FHE} = (\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add}, \mathsf{Mult})$ be conventional FHE such as [2]. The FHE4GT scheme consists of the algorithms in Figure 2. Note that key generation/encryption/decryption/addition/multiplication algorithms are the same as corresponding algorithms of FHE scheme. Additional points are homomorphic bit decomposition and homomorphic computation of greater-than bit. First, we describe procedure $\mathsf{homBitDecomp}(c, r)$, that computes an encryption of the bit decomposition of $x \in R_t$, given an FHE ciphertext c of some plaintext x. It homomorphically extracts the bit decomposition of $(b_0, b_1, ..., b_{r-1})$ of x using Digits-Extraction algorithm in [8]. The Digits-Extraction algorithm in the base-2 case for extracting the ciphertext of rth digit of the integer $x = \sum_i x_i 2^i$ is shown in the Fig. 1. Note that, the homomorphic division by 2 is possible by multiplying the ciphertext by the constant $2^{-1} \bmod q$.

```
Digits-Extraction $(c, r)$ :
  $w_{0,0} \leftarrow c$
  For $i \in [0..r-1]$:
    $y \leftarrow c$
    For $j \in [0..i]$:
      $w_{j,i+1} \leftarrow {w_{j,i}}^2$
      $y \leftarrow (y - w_{j,i+1})/2$
    $w_{i+1,i+1} \leftarrow y$
  return $w_{r,r}$
```

Fig. 1. The Digits-Extraction algorithm

```
Keygen () : return $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{FHE.Keygen}()$
Enc $(\mathsf{pk}, x \in R_{2^t})$ : return $c \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, x)$
Dec $(\mathsf{sk}, c)$ : return $[\mathsf{FHE.Dec}(\mathsf{sk}, c)]_{2^t}$
Add $(c, c')$ : return $\mathsf{FHE.Add}(c, c')$
Mult $(c, c')$ : return $\mathsf{FHE.Mult}(c, c')$
homBitDecomp $(c, r)$ :
  $w_0 \leftarrow c$
  For $i \in [0..r-1]$:
    $y \leftarrow c$
    For $j \in [0..i-1]$:
      $w_j \leftarrow \mathsf{Mult}(w_j, w_j)$
      $y \leftarrow \mathsf{Add}(y, -w_j)/2$
    $w_i \leftarrow y$
  return $(w_0, w_1, ..., w_{r-1})$
GT $(c, c')$ :
  $(\gamma_0, ..., \gamma_{r-1}) \leftarrow \mathsf{homBitDecomp}(\mathsf{Add}(c', -c), r)$
  return $\gamma_{r-1}$
```

Fig. 2. The FHE4GT scheme

Based on Digits-Extraction algorithm, for ciphertext c of x of r bits, we construct homBitDecomp algorithm. Outputs $(w_0, w_1, ..., w_{r-1})$ of homBitDecomp satisfies that

$$x = (\mathsf{Dec}(w_{r-1}) \parallel \mathsf{Dec}(w_{r-2}) \parallel \cdots \parallel \mathsf{Dec}(w_0))_2.$$

Using the homBitDecomp algorithm, only given two ciphertexts c and c' of x and x', we can compute an encryption of greater-than bit homomorphically. In the GT algorithm, it simply computes $\mathsf{homBitDecomp}(c' - c)$, then outputs encryption of the most significant bit because the sign bit of the value $c' - c$ becomes a bit representing the greater-than relation.

Using such GT algorithm, we can compute the greater-than bit encryption $\mathsf{Enc}(b)$ given only $\mathsf{Enc}(x)$ and $\mathsf{Enc}(x')$, neither knowing the secret key nor interaction with secret key holder.

The FHE4GT scheme is semantically secure and fully homomorphic with suitable choice of parameters that makes the underlying FHE scheme so.

Efficiency: We estimate efficiency of homomorphic operations of $\mathsf{FHE4GT}$ scheme of parameter q, r, t. Addition $\mathsf{Add}(c, c')$ and multiplication $\mathsf{Mult}(c, c')$ are done by one addition/multiplication of FHE of parameter q, r. Homomorphic computation of greater-than relation GT depends on the number of multiplications of ciphertexts and the circuit of depth L because it is dominated by homomorphic bit decomposition $\mathsf{homBitDecomp}$. In the $\mathsf{homBitDecomp}$, it recursively needs to repeat squaring. Therefore, for bit length t of plaintext, since we use the plaintext modulus $2^r = 2^{t+1}$, homomorphic computation of greater-than relation needs the circuit of depth corresponding to L

$$\mathsf{depth}_{\mathsf{GT}} = O(r^2).$$

IV. Classification

In this section, we propose the homomorphic classifier $\mathsf{homClassify}$ based on the $\mathsf{FHE4GT}$ scheme. Let us consider the case of binary classification (i.e. a machine learning algorithm like SVM[16]).

$$g(\mathsf{x}) = \mathsf{w} \cdot \mathsf{x} + b = \sum_{i=1}^{d} w_i x_i + b.$$

Here, w is the weight vector and b is the bias. A hyperplane is a generalization of a straight line to > 2 dimensions. We consider a linear discriminant function. A hyperplane contains all the points in a d dimensional space satisfying the following equation.

$$w_1 x_1 + w_2 x_2 + ... + w_d x_d + b = 0.$$

By identifying the components of w with the coefficients w_i, we can see the weight vector and the bias define a linear decision surface in d dimensions. It is common to simplify notation by including the bias in the weight vector, i.e. $b = w_0, x_0 = 1$.

$$g(\mathsf{x}) = \mathsf{w} \cdot \mathsf{x} = \sum_{i=0}^{d} w_i x_i.$$

We consider a linear classifier of the following form.

$$f(\mathsf{x}; \mathsf{w}) = \begin{cases} 1 & \text{if } \ g(\mathsf{x}) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

In our proposed homomorphic classification (Fig. 3), we treat given test data x and output of classifier f as ciphertexts (i.e. only given $\mathsf{Enc}(x_i)$, it outputs $\mathsf{Enc}(f(\mathsf{x}; \mathsf{w}))$.

Therefore, we consider the testing stage where the client's data is encrypted with its own public key.

Efficiency: In the case of homomorphic classification, it needs the circuit of depth $\mathsf{depth}_{\mathsf{homClassify}} = \mathsf{depth}_{\mathsf{GT}} + 1$.

V. Benchmark Results

We implemented the $\mathsf{FHE4GT}$ scheme described in Section III using homomorphic encryption library HElib [11].

In this section, we show experimental results of $\mathsf{FHE4GT}$ and $\mathsf{homClassify}$. We implemented our proposed scheme using homomorphic encryption library HElib by Halevi and Shoup [11], which is based on the BGV scheme [3].

```
homClassify (c, w) :
  $g \leftarrow \mathsf{Enc}(0)$
  For $i \in [0..d]$:
    $g \leftarrow \mathsf{Add}(g, w_i c_i)$
  $y \leftarrow \mathsf{GT}(g, \mathsf{Enc}(0))$
  return $y$
```

Fig. 3. The $\mathsf{homClassify}$ algorithm

All timings were obtained running on an Intel Core i7 at 2.4 GHz with 16GB of memory. Timings are given in seconds. These results are the averages of 100 times trials.

Choise of Parameters: We implemented three cases where the size of plaintext t is 2bits, 4bits and 8bits (did not use slots). Each parameter sets are shown in the Table I. For these experiments, we fix the security parameter $\lambda = 80$, the standard deviation of Gaussian distribution (i.e. a noise parameter) $\sigma = 3.2$, and the secret key is chosen uniformly random among ternary vector (i.e. $\{-1, 0, 1\}$) of Hamming weight 64. In theory, the level L of ciphertext is roughly $O(r^2)$, but in fact the smaller value shown in the Table I is sufficient.

Timings for $\mathsf{FHE4GT}$ and $\mathsf{homClassify}$: Table II shows timing s for the $\mathsf{FHE4GT}$ operations: encryption, decryption, addition, and multiplication. Table III shows timings for the GT and $\mathsf{homClassify}$. For the $\mathsf{homClassify}$, we experimented in three cases of dimensions $d = 10, 20, 30$ for each parameter set.

The time spent on homomorphic computation of greater-than relation is about 0.02 seconds for parameter set P_1, 0.3 seconds for P_2 and 3.2 seconds for P_3. In addition, the time spent on homomorphic classification for dimension $d = 10$ is about 0.03 seconds for P_1, 0.3 seconds for P_2 and 3.3 seconds for P_3. As the circuit of depth L increases, the ciphertext modulus q also increases, which affects the calculation speed.

VI. Conclusion

In order to protect a private data from leakage when classifying in machine learning, we should encrypt the data.

In this paper, using fully homomorphic encryption scheme, we constructed fully homomorphic encryption scheme $\mathsf{FHE4GT}$ that can homomorphically compute an encryption of the greater-than bit that indicates $x > x'$ or not, given two ciphertexts c and c' of x and x', respectively, without knowing the secret key. Then, we proposed homomorphic classifier $\mathsf{homClassify}$ that can homomorphically classify a given encrypted data without decrypting it, using machine learned parameters. By using homomorphic classifier, we are able to deal more safely with classification using the learning results.

Acknowledgment

This work was supported by JST CREST Grant Number JPMJCR1503, Japan.

TABLE I
PARAMETER SETS FOR FHE4GT.

	Parameters
P_1	$m = 4051, p = 2, r = 3, t = 2, L = 3, \sigma = 3.2, \lambda = 80$
P_2	$m = 7781, p = 2, r = 5, t = 4, L = 7, \sigma = 3.2, \lambda = 80$
P_3	$m = 14351, p = 2, r = 9, t = 8, L = 17, \sigma = 3.2, \lambda = 80$

TABLE II
TIMING FOR FHE4GT OPERATIONS (SEC): ENCRYPTION, DECRYPTION, ADDITION, AND MULTIPLICATION.

Parameter set	FHE4GT.Enc	FHE4GT.Dec	FHE4GT.Add	FHE4GT.Mult
P_1	0.0044	0.0016	3.654E-5	0.0096
P_2	0.0149	0.0062	1.514E-4	0.0449
P_3	0.057	0.0227	6.127E-4	0.1706

TABLE III
TIMING FOR GT AND HOMCLASSIFY (SEC).

	GT	homClassify		
		$d = 10$	$d = 20$	$d = 30$
P_1	0.0224	0.0317	0.0304	0.0312
P_2	0.2559	0.2593	0.2906	0.2932
P_3	3.235	3.3302	3.3754	3.4104

REFERENCES

[1] Raphaël. Bost, Raluca Ada Popa, Stephen Tu and Shafi Goldwasser, Machine learning classification over encrypted data. IACR Cryptology ePrint Archive, (2014/331).

[2] Zvika Brakerski, Fully homomorphic encryption without modulus switching from classical GapSVP. Crypto 2012, LNCS 7417, pages 868-886. Springer, 2012.

[3] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan, (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, ITCS, pages 309-325. ACM, 2012.

[4] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun, Batch fully homomorphic encryption over the integers. In Thomas Johansson and Phong Q. Nguyen, editors, EUROCRYPT 2013, LNCS 7881, pages 315-335. Springer, 2013.

[5] Jung Hee Cheon, Miran Kim, Kristin Lauter, Homomorphic Computation of Edit Distance. Financial Cryptography and Data Security 2015, LNCS 8976, pp 194-212, 2015.

[6] Junfeng Fan and Frederik Vercauteren, Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive, (2012/144).

[7] Craig Gentry, Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, STOC, pages 169-178. ACM, 2009.

[8] C. Gentry, S. Halevi, and N. P. Smart, Better bootstrapping in fully homomorphic encryption. In Public Key Cryptography, pages 1-16. 2012.

[9] Philippe Golle, A private stable matching algorithm. In Financial Cryptography and Data Security, pages 65-80. Springer, 2006.

[10] Thore Graepel, Kristin Lauter, Michael Naehrig, ML Confidential: Machine Learning on Encrypted Data. ICISC 2012, LNCS 7839, Springer-Verlag (2013), pp 1-21.

[11] Shai Halevi, Victor Shoup, Algorithms in HElib. CRYPTO2014, LNCS 8616, pp 554-571.

[12] Yu Ishimaki, Kana Shimizu, Koji Nuida, Hayato Yamana, Faster privacy-preserving search for genome sequences using fully homomorphic encryption, SCIS'16, 2016, Japan.

[13] Wen-jie Lu, Shohei Kawasaki, Jun Sakuma, Cryptographically-secure Outsourcing of statistical Data Analysis I: Descriptive Statistics. CSS'15, 2015, Japan.

[14] K. Lauter, A. López-Alt, M. Naehrig, Private Computation on Encrypted Genomic Data. LATINCRYPT 2014, LNCS 8895, Springer-Verlag, pp 3-27.

[15] Junqiang Liu, Jiuyong Li, Shijian Xu, and Benjamin C.M. Fung, Secure Outsourced Frequent Pattern Mining by Fully Homomorphic Encryption. S. Madria and T. Hara (Eds.): DaWaK 2015, LNCS 9263, pp. 70-81, 2015.

[16] V. Vapnik, and A. Lerner, Pattern recognition using generalized portrait method. Automation and Remote Control, 24, pp. 774-780, 1963.