SCIENCE CHINA
Information Sciences

• **Supplementary File** •

Securely min and k-th min computations with fully homomorphic encryption

Bingbing JIANG[1] & Yuan ZHANG[1]*

[1]*Department of Computer Science and Technology, Nanjing University, Nanjing* 210046*, China*

Appendix A Preliminaries

In this section, we introduce some basic knowledge of semi-honest model, fully homomorphic encryption and learning with error. Then, we will review the GSW13 fully homomorphic encryption scheme. Firstly, we simply introduce the definition of the semi-honest model.

Appendix A.1 Semi-honest Model

In the semi-honest model, we assume each party is honest but curious. That is, all parties abide by the protocol and cannot abort its execution. Nevertheless, each party is curious to learn others' data and they can perform computations on data received from others. Suppose a protocol is secure under this model, if each party cannot obtain extra knowledge about others' private data in the process of executing the protocol's execution.

Definition 1. The protocol $\mathcal{M}$ is said to be perfect privacy-preserving computation of the minimum value against party P_i if it reveals nothing more than the final result to P_i in the execution of $\mathcal{M}$. That is, given all inputs $\{x_0, \ldots, x_n\}$, there exists a polynomial time simulator S_i that can simulate the party P_i. We denote $\{S_i(x_i, \mathcal{M}(x_0, \ldots, x_n), K_i)\}_{\{x_0,\ldots,x_n\}}$ to S_i's view and $\{VIEW_i(x_0, \ldots, x_n)\}_{\{x_0,\ldots,x_n\}}$ represents the party P_i's view in $\mathcal{M}$'s execution. Suppose $\mathcal{M}$ is secure in the semi-honest model, if S_i's view is computationally indistinguishable with P_i's view and $K_i = \varnothing$, where K_i represents the party P_i's extra information about others' data. Namely,

$$\{S_i(x_i, \mathcal{M}(x_0, \ldots, x_n))\}_{\{x_0,\ldots,x_n\}} \overset{c}{\equiv} \{VIEW_i(x_0, \ldots, x_n)\}_{\{x_0,\ldots,x_n\}}$$

If $K_i \neq \varnothing$, then $\mathcal{M}$ is weakly privacy-preserving against P_i, in the sense that P_i learns no more information than K_i about others' private data in the execution of $\mathcal{M}$.

Appendix A.2 Learning with Errors, SIVP and GapSVP

Regev firstly introduced the learning with errors (LWE) problem in 2005 and presented that the hardness of LWE can be reduced quantum to the lattice hard problems. Then, Peikert introduced an efficient classical reduction between LWE and the lattice intractable problems. The details as follows.

Definition 2 (Learning with Errors)**.** Let λ be the security parameter, $n = n(\lambda)$ be an integer dimension of a lattice, $q = q(\lambda) \geqslant 2$ be an integer and $\chi = \chi(\lambda)$ be an error distribution over $\mathbb{Z}$.
—(Searchable LWE) Sample $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ uniformly and then draw $\mathbf{a}_i \leftarrow \mathbb{Z}_q^n$ uniformly, $e_i \leftarrow \chi$. Set $b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i$. The searchable LWE is to find $\mathbf{s}$, given $m = m(\lambda)$ samples $\{(\mathbf{a}_i, b_i)\}_{i=1}^m$, called LWE$_{n,m,q,\chi}$.
—(Decision LWE) The decision LWE, denoted $LWE_{n,q,\chi}$, is to distinguish two distributions: The first one is a uniform distribution over $\mathbb{Z}_q^{n+1}$. The second is that one first samples $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and then draws $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^{n+1}$ by sampling $\mathbf{a}_i \leftarrow \mathbb{Z}_q^n$ uniformly, $e_i \leftarrow \chi$ and setting $b_i = \langle \mathbf{a}_i, \mathbf{s}_i \rangle + e_i$.
The learning with errors (LWE) assumption is that the $LWE_{n,m,q,\chi}(LWE_{n,q,\chi})$ is intractable.

Definition 3 ($SIVP_{\gamma(n)}$)**.** Let Λ be a $n-$dimension lattice. The $SIVP_{\gamma(n)}$ problem is to output n linearly independent vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ such that $\max\limits_i\{v_i\} \leqslant \gamma(n) \cdot \lambda_n$, where $\lambda_n = \min\limits_r\{r : dim(span(B(0, r) \cap \Lambda)) \geqslant n\}$.

* Corresponding author (email: zhangyuan05@gmail.com)

Definition 4 ($GapSVP_{\gamma(n)}$)**.** Let Λ be a $n-$diemension lattice and d be a real number. The $GapSVP_{\gamma(n)}$ is to distinguish whether $\lambda_1 < d$ or $\lambda_1 \geqslant \gamma(n) \cdot d$, where λ_1 is the length of the shortest vector in Λ.

Definition 5 (B-bounded distributions)**.** A distribution ensemble $\{\chi_n\}_{n\in\mathbb{N}}$ over the integers is called B-bounded distribution if

$$\Pr_{e\leftarrow\chi_n}[|e| > B] = negl(n)$$

.

Theorem 1. Let $q = q(n)$ be either a prime power or a product of small (size poly(n)) distinct primes, $B \geqslant \omega(logn) \cdot \sqrt{n}$, and χ is an efficiently sampleable B-bounded distribution. If there exists an efficient algorithm solving the $LWE_{n,q,\chi}$ problem, then:

- There is an efficient quantum algorithm for $GapSVP_{\widetilde{O}(nq/B)}$ on any n-dimension lattice.
- There is an efficient classical algorithm that solves $GapSVP_{\widetilde{O}(nq/B)}$ on any n-dimension lattice.

In both cases, if one also considers solving $LWE_{n,q,\chi}$ with sub-polynomial advantage, then request $B \geqslant \widetilde{O}(n)$ and $\gamma(n) \geqslant \widetilde{O}(n^{1.5}q/B)$.

Appendix A.3 Fully Homomorphic Encryption

A homomorphic encryption scheme consists of four probabilistic polynomial time algorithms ($Gen, Enc, DEC, Eval$), where $Eval$ is an algorithm that supports homomorphic computations on ciphertexts.

Gen(1^λ): $(pk, evk, sk) \leftarrow Gen(1^\lambda)$. Choose the security parameter λ. Generate a public encryption key pk, a public evaluation key evk and a private key sk.

Enc(pk, μ): $\mathbf{c} \leftarrow Enc(pk, \mu)$. Encrypt a message $\mu \in \{0, 1\}$ and output the ciphertext c.

Dec($sk, \mathbf{c}$): $\mu^* \leftarrow Dec(sk, \mathbf{c})$. Decrypt a ciphertext $\mathbf{c}$ and obtain a corresponding plaintext $\mu^* \in \{0, 1\}$.

Eval($evk, f, \mathbf{c}_1, \dots, \mathbf{c}_l$): $\mathbf{c}_f \leftarrow Eval(evk, f, \mathbf{c}_1, \dots, \mathbf{c}_l)$. Homomorphically evaluate the function f and the inputs $\mathbf{c}_1, \dots, \mathbf{c}_l$, and output $\mathbf{c}_f$ that is a valid ciphertext of $f(\mu_1, \dots, \mu_l)$, where $\mathbf{c}_i$ is a valid ciphertext of μ_i, $i = 1, \dots, l$.

Definition 6 ($\mathcal{C}$-homomorphic)**.** Denote a class of functions $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$. Suppose a scheme $\Pi = (Gen, Enc, Dec, Eval)$ is $\mathcal{C}$-homomorphic, if for any arbitrary function $f_\lambda \in \mathcal{C}_\lambda$, and inputs $\mathbf{c}_1, \dots, \mathbf{c}_l$ such that $\mathbf{c}_i \leftarrow Enc(pk, \mu_i)$, $i = 1, \dots, l$, it satisfies that the probability of incorrectly decrypting the homomorphically evaluated result is negligible, that is

$$Pr[Dec(sk, Eval(evk, f_\lambda, \mathbf{c}_1, \dots, \mathbf{c}_l)) \neq f_\lambda(\mu_1, \dots, \mu_l)]$$

$$= neg(\lambda)$$

where $(pk, evk, sk) \leftarrow Gen(1^\lambda)$.

Definition 7 (compactness)**.** Let λ be the security parameter. If there exists a polynomial $p = p(\lambda)$ holding that the output length of $Eval$ is at most $p(\lambda)$ bits long without relation to the function f or the numbers of inputs, we say the homomorphic encryption scheme $\Pi = (Gen, Enc, Dec, Eval)$ is compact.

Definition 8 (fully homomorphic encryption)**.** A scheme $\Pi = (Gen, Enc, Dec, Eval)$ is fully homomorphic, if Π is compact and it can homomorphically evaluate the arbitrary circuit taken from the class of all arithmetic circuits over $GF(2)$.

Appendix A.4 Review of the GSW13 Scheme

In this section, we formally describe the basic GSW fully homomorphic encryption scheme. At the beginning, we introduce three operations used in the encryption algorithm for slow noise-growth. Assume three vectors $\mathbf{a} = (a_0, \dots, a_{n-1}) \in \mathbb{Z}_q^n$, $\alpha = (\alpha_0, \dots, \alpha_{N-1}) \in \{0, 1\}^N$, $\beta = (\beta_0, \dots, \beta_{N-1}) \in \mathbb{Z}_q^N$.

$BitDecomp(a)$=$(a_{0,0}, a_{0,1}, \dots, a_{0,l-1}, a_{1,0}, \dots, a_{1,l-1}, \dots, a_{n-1,l-1})$, where $a_{i,j}$ is the j-th element of the binary representation of a_i.

$BitDecomp^{-1}(\alpha)$=$(\Sigma_{i=0}^{l-1}2^i\alpha_i, \Sigma_{i=l}^{2l-1}2^{i-l}\alpha_i, \dots, \Sigma_{i=(n-1)l}^{N-1}2^{i-(n-1)l}\alpha_i)$, where $\alpha \in \{0, 1\}^N$.

$Flatten(\beta)$=$BitDecomp(BitDecomp^{-1}(\beta))$.

We can see that $BitDecomp(\cdot)$ expands each element of a vector to its binary representation, $BitDecomp^{-1}(\cdot)$ can be seen as an inverse operation of $BitDecomp(\cdot)$ and it makes each l elements of a vector to a number in $\mathbb{Z}_q$. These three operations on a matrix are performed on each column vector of the matrix. That is, $BitDecomp(\mathbf{A}) = \begin{bmatrix} BitDecomp(\mathbf{A}_0) \\ \vdots \\ BitDecomp(\mathbf{A}_{n-1}) \end{bmatrix}$.

$BitDecomp^{-1}(\cdot)$ and $Flatten(\cdot)$ on a matrix are similar to $BitDecomp(\cdot)$ on a matrix.

We now describe the GSW fully homomorphic encryption scheme. It consists of six probabilistic polynomial time algorithms (**Setup**,**Gen**,**Enc**,**Dec**,**Add**,**Multi**).

Setup($1^\lambda, 1^L$): Let λ be the security parameter and L the max circuit depth. Choose appropriate LWE parameters: modulus $q = q(\lambda, L)$, lattice dimension $n = n(\lambda, L)$, and error distribution $\chi = \chi(\lambda, L)$. Choose parameter $m = O(n\ log\ q)$. Set $params = (q, n, \chi, m)$. Let $l = \lfloor log\ q \rfloor + 1$ and $N = n \times l$.

Gen($params$): Choose randomly $\mathbf{t} \leftarrow \mathbb{Z}_q^{n-1}$. Choose a random matrix $\mathbf{B} \leftarrow \mathbb{Z}_q^{m\times(n-1)}$ and a vector $\mathbf{e} \leftarrow \chi^m$. Set $\mathbf{b} = B \cdot \mathbf{t} + \mathbf{e}$. Output the secret key $sk = \mathbf{s} = (1, -t_1, \dots, -t_{n-1}) \in \mathbb{Z}_q^n$, and the public key $pk = \mathbf{A} = [\mathbf{b}|B]$. Let $\mathbf{v} = Powerof2(\mathbf{s})$. (Note that $\mathbf{A} \cdot \mathbf{s} = \mathbf{e}$.)

Enc($params, pk, \mu$)**:** Choose randomly a matrix $\mathbf{R} \leftarrow \{0,1\}^{N\times m}$. Then encrypt the message μ as follows:

$$\mathbf{C} = Flatten(\mu \cdot \mathbf{I}_N + BitDecomp(\mathbf{R}\cdot \mathbf{A})) \in \mathbb{Z}_q^{N\times N}$$

Output the ciphertext $\mathbf{C}$.

Dec($params, sk, \mathbf{C}$)**:** Let $v_i \in (q/4, q/2]$. Output $\mu = \lfloor \langle \mathbf{C}_i, \mathbf{v}\rangle / v_i \rceil$.

Add($params, pk, \mathbf{C}_1, \mathbf{C}_2$)**:** To add two ciphertexts $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{Z}_q^{N\times N}$. Output $Flatten(\mathbf{C}_1 + \mathbf{C}_2)$.

Multi($params, pk, \mathbf{C}_1, \mathbf{C}_2$)**:** To multiply two ciphertexts $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{Z}_q^{N\times N}$. Output $Flatten(\mathbf{C}_1 \cdot \mathbf{C}_2)$.

Appendix B Analysis

We present the privacy-preserving minimum or k-th minimum computing protocols in the above section. Sequentially, we here provide a rigorous analysis on the protocols' complexity of round and communication, correctness and security. First, in two privacy-preserving min computing protocols, each user only does the following operations: encrypting and sending the data, and finally assisting the server to decrypt the ciphertext of the minimum value. So, our privacy-preserving min computing protocol only has two rounds. However, in the complex k-th minimum computation protocol, each user also helps the server to count *count*, and the number of the plaintext of $\mathbf{min}_j$ equal to the plaintext of $\mathbf{C}_{i,j}$. The k-th minimum computation protocol has $O(l)$ rounds where l is the bit length of the data. Second, our first protocol only needs $O(N_0 \cdot l \cdot N^2)$ where N_0 is the number of users and $N \times N$ is the size of a varGSW ciphertext: Each user sends an encryption of the input $\mathbf{C}_{i,j} = varGSW.Enc_{pk}(x_{i,j}), c_i = (\mathbf{c}_{i,1}, \ldots, \mathbf{c}_{i,l})$ and this is $N_0 \cdot l \cdot N^2$ bits; the server sends the final ciphertext to all users for decryption and this needs $l \cdot N^2$ bits; and each user sends the partial decryption of the received ciphertext to the server and it is $l \cdot N$ bits. Compared to previous schemes based on additionally homomorphic encryption or the secure bitwise XOR computations, our protocols are featured with either lower rounds or stronger security guarantee.

Next, we discuss the protocols' correctness and security as summarized in the following proposition and theorem.

Proposition 1. The accuracy of our privacy-preserving min or k-th computation protocols is exponentially approximate to 1 if all users and the server follow the protocol on the assumption of semi-honest model.

Proof. We can see that the protocol's accuracy is determined by users' decryption, encryption, communication with the server, and the server's homomorphic evaluation computations. The failure of the encryption, decryption and evaluation of the FHE scheme is negligible. On the condition of semi-honest model, all users and the server follow the execution of protocol. Though they are curious about others' private information, they cannot abort the process of the protocols' execution or send wrong information in the execution. So, the accuracy is exponentially approximate to 1.

Theorem 2. Our privacy-preserving minimum or k-th minimum computation protocols are perfectly privacy-preserving against all users and the server in the semi-honest model.

Proof. In our protocols, every user only receives the ciphertext of the minimum value, has only one secret key share of the fully homomorphic encryption scheme, and can't decrypt the ciphertext alone. Therefore, our protocols reveal no more information to the users.

Similarly, the server only receives the encryptions of all users' data, and performs homomorphic evaluation on the ciphertexts. If the fully homomorphic encryption scheme is secure, the server can't learn extra knowledge about all users' data.