Hindawi
Security and Communication Networks
Volume 2021, Article ID 9974604, 15 pages
https://doi.org/10.1155/2021/9974604

Research Article

Postquantum Cut-and-Choose Oblivious Transfer Protocol Based on LWE

Hangchao Ding,[1] **Han Jiang**,[2,3] **and Qiuliang Xu**[2,3]

[1]*School of Mathmatics, Shandong University, Jinan 250100, China*
[2]*School of Software, Shandong University, Jinan 250101, China*
[3]*Key Laboratory of Shandong Province for Software Engineering, Jinan 250101, China*

Correspondence should be addressed to Han Jiang; jianghan@sdu.edu.cn and Qiuliang Xu; xql@sdu.edu.cn

Received 5 April 2021; Accepted 4 August 2021; Published 10 September 2021

Academic Editor: Vincenzo Conti

We propose postquantum universal composable (UC) cut-and-choose oblivious transfer (CCOT) protocol under the malicious adversary model. In secure two-party computation, we construct s copies' garbled circuits, including half check circuit and half evaluation circuit. The sender can transfer the key to the receiver by CCOT protocol. Compared to PVW-OT [6] framework, we invoke WQ-OT [35] framework with reusability of common random string (crs) and better security. Relying on LWE's assumption and the property of the Rounding function, we construct an UC-CCOT protocol, which can resist quantum attack in secure two-party computation.

1. Introduction

1.1. Background. In secure two-party computation, sender P_1 and receiver P_2 jointly compute the value of function $f(x, y)$. P_1 inputs x and P_2 inputs y. Then, P_1 and P_2 obtain the value of $f(x, y)$. Yao's garbled circuits, used in secure two-party computation, are only secure in the semihonest adversary model. In Yao's protocol, a single garbled circuit is constructed to evaluate. For better security, we apply s copies garbled circuits in secure two-party computation. For constructing secure protocol under the malicious adversary model, cut-and-choose methodology is used to prevent malicious party from cheating by constructing incorrect garbled circuits in secure two-party computation. This methodology needs to construct s copies' garbled circuits.

Secure two-party computation protocol is implemented by Garbled Circuit (GC). Sender P_1 and receiver P_2 jointly compute the value of $f(x, y)$ by computing $C(x, y)$ ($f(x, y) = C(x, y)$), which satisfy security, privacy, correctness, and input's independence. In 1986, Yao [1] proposed a secure two-party computation protocol, which is mainly based on GC and Oblivious Transfer (OT).

Yao's protocol is only secure and efficient in the semihonest adversary model. However, this protocol cannot obtain the security of the malicious adversary model. In 1987, Goldreich and Micali [2] proposed a GMW compiler, which can compile protocols under the semihonest adversary model to protocols under the malicious adversary model. Application for the GMW compiler needs a number of zero-knowledge proof and commitment mechanisms. This operation results in high complexity and low efficiency. Construction of Universal Composable (UC) protocol under the malicious adversary model has great significance in secure two-party computation.

In Yao's protocol, sender P_1 constructs a garbled circuit, including $2n$ wires. In these wires, each wire corresponds to a garbled circuit key $k_i^b, b \in \{0, 1\}$, where b represents the corresponding value of wires. Sender P_1 inputs $X = (x_1, x_2, \ldots, x_n)$, $x_i \in \{0, 1\}$, in n copies' wires, Receiver P_2 inputs $Y = (y_1, y_2, \ldots, y_n)$, $y_i \in \{0, 1\}$, in remaining n copies' wires. P_1 sends X and $(k_1^{x_1}, k_2^{x_2}, \ldots, k_n^{x_n})$ to P_2. P_2, as the circuit evaluator, needs P_1's input's value X and corresponding garbled circuit key's value. For computing $C(X, Y)$, P_2 inputs Y and needs corresponding key's value in

remaining wires. P_2 needs to obtain corresponding key's value about $Y = (y_1, y_2, \ldots, y_n)$. It means that P_2 needs to obtain corresponding $k_{n+i}^{y_i}$ from $(k_{n+i}^0, k_{n+i}^1), i = \{1, 2, \ldots, n\}$. So, P_2 can apply OT protocol to obtain corresponding garbled key's value.

1.2. Related Work

1.2.1. Oblivious Transfer. OT protocol, as a basic building primitive in cryptography, has great security significance in secure two-party computation or multiparty computation. For better, intuitively, understanding OT_2^1 protocol, we give a brief introduction about its ideal function F_{OT} in Figure 1.

Denote ideal function F_{OT}: $\{m_0, m_1\} \times \{\sigma\} \longrightarrow \{\perp, m_\sigma\}$. For better understanding this process on the Internet, we consider session identity sid for our description. This function is implemented by the interaction between the sender S and the receiver R.For understanding the ideal function F_{OT}, we introduce an OT_2^1protocol for understanding OT's role in Figure 2. The sender has two-message m_0 and m_1. The receiver wants to obtain message m_σ, $\sigma \in \{0, 1\}$. Then, the sender and the receiver apply OT_2^1 protocol. The receiver can obtain message m_σ. In this process, the sender has no information about receiver's selected bit σ. The receiver has no information about $m_{1-\sigma}$.

The OT protocol can be constructed by the public key encryption (PKE) system or trapdoor permutation function. We consider constructing a PKE-based OT protocol. Next, we introduce this construction, given a series of PKE algorithms (KeyGen, Enc, Dec) as follows:

1.2.2. Cut-and-Choose Technique. For better understanding the importance of cut-and-choose technique in secure two-party computation, firstly, we give a brief description about basic Yao's protocol in Figure 3, which is the original secure two-party computation protocol.

Yao's protocol, used to compute the function, is only secure in the semihonest adversary model, which is just based on a single garbled circuit for evaluation. The circuit of this construction is not enough secure, which cannot obtain malicious adversary's security. Considering its single garbled circuit, if someone cheats in this protocol, it cannot be detected. It can be solved by applying the GMW compiler to obtain malicious adversary's security. However, this needs extracomputation.

For correctly computing the value of function $f(x, y)$, it is better to construct many circuits for computation. Circuit constructor constructs many garbled circuits and sends these circuits to the circuit evaluator. Then, some circuits are used for check circuit and used to check the correctness of garbled circuits. Some circuits are used for evaluation circuit, for evaluating the value of function. This technique is called cut-and-choose methodology, meaning cutting some garbled circuits in the first step and then choosing some circuits for checking in the second step.

Cut-and-choose methodology, as a tool used in secure two-party computation, can prevent the circuit constructor from cheating in constructing incorrect circuits. This technique can reduce the use of zero-knowledge proof techniques, which can improve the efficiency of secure computation protocols.

For better understanding this process, we give a brief description about cut-and-choose process in Figure 4. Here, we mainly introduce a universal technique "cut-and-choose" methodology.

Firstly, P_1 constructs s copies' garbled circuits and sends these circuits to P_2. Secondly, P_2 chooses $s/2$ circuits for checking. Denote set CGC as check-circuit set, including $s/2$ copies' check circuits. Otherwise, the rest of $s/2$ circuits are used for evaluating circuits. Define EGC as evaluation-circuit set. P_2 obtains check circuits for checking correctness of half circuits and then evaluates $f(x, y)$ in remaining evaluation circuits. Some garbled circuits maybe incorrectly constructed, so evaluator P_2 can use majority of evaluation-circuit output as value of $f(x, y)$.

Considering cut-and-choose technique applied on secure two-party computation, P_1 may carry out select failure attack for P_2. P_1 may use different input values to obtain different ciphertexts and confuse some values about index bit j from P_2's evaluation set. The main reason about this attack is the separation between cut-and-choose methodology with oblivious transfer process. So, it is crucial to combine cut-and-choose methodology with oblivious transfer protocol, called the CCOT protocol.

OT is a basic protocol in secure two-party computation, where P_1 sends garbled key's value of every wire in the garbled circuit to P_2. If cut-and-choose methodology is separated from the OT protocol, this separation may result in selection failure attack. It is crucial to combine cut – and – choose methodology with the OT protocol, that is, cut – and – choose oblivious transfer (CCOT) protocol. This has crucial significance about security in secure two-party computation protocol.

1.2.3. Related Reference. OT was firstly proposed by Rabin [3]. OT is a fundamental primitive in secure two-party and multiparty computation. In secure two-party computation, receiver P_2 obtains one or two values from sender P_1 through the OT protocol. As a result, receiver P_2 only obtains corresponding values and has nothing about other information. Sender P_1 is oblivious to P_2's selection bit. In 2007, Peikert and Waters [4] proposed a primitive 'lossy trapdoor functions' (lossy TDFs) and applied 'lossy TDFs' to construct trapdoor function. In 2008, Peikert and Vaikuntanathan [5] proposed a framework for efficient and composable OT, which is constructed by the dual-mode PKE System, called PVW-OT framework. In Peikert's dual-mode encryption system, it includes messy mode and decryption mode (called Dec mode). However, common random string (crs) can be reused with bounded limitation. Sender's computational security in messy mode and receiver's computational security in Dec mode can be obtained in this scheme. However, it cannot suffice for each party's statistical security in both modes. Peikert also construct corresponding schemes based on DDH, QR, and LWE's assumption. Fully simulatable PVW-OT protocol's security is

Sender S sends $(sid, sender, m_0, m_1)$ to storage unit of ideal function, store m_0 and m_1; Receiver R inputs $(sid, receiver, \sigma)$ to ideal function's storage unit;
If this message exists, storage unit sends (sid, m_σ) to receiver R. Otherwise, there is nothing for outputting. And Sender S outputs $\perp$.

Figure 1: The ideal function F_{OT}.

Input: Sender S inputs (sid, m_0, m_1), Receiver R inputs (sid, σ), $\sigma \in \{0, 1\}$;
Protocol Process: Receiver R invokes $KeyGen(\sigma) \rightarrow (pk, sk)$, stores secret key sk in storage unit, then sends public key (sid, pk) to Sender S;
Sender S obtains (sid, pk) from R, then invokes Enc algorithm to get $c_0 \leftarrow Enc(pk, 0, m_0)$ and $c_1 \leftarrow Enc(pk, 1, m_1)$. Next S sends c_0 and c_1 to receiver R;
Receiver R obatins (c_0, c_1) from S, then invokes Dec algorithm to obtain $m_\sigma \leftarrow Dec(sk, c_\sigma)$;
Output: Sender outputs $\perp$, Receiver outputs m_σ.

Figure 2: The process of the OT protocol.

Input: Circuit Constructor P_1 inputs $x \in \{0, 1\}^n$, Circuit Evaluator P_2 inputs $y \in \{0, 1\}^n$, which their binary representation are $x_1 x_2 \cdots x_n$ and $y_1 y_2 \cdots y_n$.
Auxiliary Input: Given Boolean Circuit used to compute function $f(x, y)$, which can obtain $C(x, y) = f(x, y)$, $f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$.
Protocol:

Step1: P_1 constructs Garbled Circuit (GC) based on Boolean Circuit, and sends this garbled circuit to P_2.

Step2: For P_1's input $x_1, x_2, \cdots x_n$, there are circuit-input wires $w_1, w_2, \cdots, w_n$, which encrypted by garbled keys $k_1^{x_1}, k_2^{x_2}, \cdots k_n^{x_n}$ in these circuit-input wires. For P_2's input $y_1 y_2 \cdots y_n$, there are circuit-input wires $w_{n+1}, w_{n+2}, \cdots, w_{2n}$, which encrypted by garbled keys $k_{n+1}^{y_1}, k_{n+2}^{y_2}, \cdots k_{2n}^{y_n}$ in these circuit-input wires. Corresponding garbled key's value $k_{n+1}^{y_1}, k_{n+2}^{y_2}, \cdots k_{2n}^{y_n}$ are obtained by OT_2^1 in P_1's input (k_{n+i}^0, k_{n+i}^1), $y_i \in \{0, 1\}$.

Step3: Circuit Evaluator P_2 has obtained $2n$ garbled keys corresponding to $2n$ input wires, which are GC's input wires. Finally, P_2 computes circuit by corresponding garbled values, obtain $f(x, y)$.

Figure 3: Yao's protocol.

universally composable (UC), which can compose securely other protocols in complex Internet. UC security is proposed by Canetti [6]. This guarantees security when many protocols are executed in parallel under malicious adversary's environment. Some lattice-based oblivious transfer protocols are proposed in postquantum era; most of these protocols are based on LWE's assumption under the semihonest, malicious, or covert adversary model [7–9].

In 2020, Quach [10] proposed a UC-secure OT protocol based on LWE's assumption and rounding function, which can be seen as a modified framework of PVW-OT, called WQ-OT. In WQ-OT protocol, the rounding function is applied on constructing UC-OT. Considering that the rounding function is a smooth projective hash function (SPHF), it can be applied on our scheme with the property of rounding function's hash key and projective key. In WQ-OT, crs can be reused many times without limitation, and statistical security of the sender and the receiver can be obtained in both messy mode and Dec mode.

The Process:

— P_1 constructs s copies garbled circuits and define these garbled circuits' index as $\{1, 2, \cdots, s\}$;
— P_1 sends all circuits index to P_2, P_2 chooses $s/2$ circuits for checking, included in set CGC;
— P_1 opens these $s/2$ check-circuits, and sends corresponding garbled key values to P_2;
— P_2 checks the correctness of above circuits, if they are correct circuits, continue to evaluate remaining circuits for computing value of $f(x, y)$. Otherwise, termination, output $\perp$.

Figure 4: Cut-and-choose methodology.

SPHF has a wide range of applications, such as key exchange and oblivious transfer [11, 12]. In 2012, Halevi and Kalai [13] proposed a two-message OT, which is based on projective hashing function. Cramer and Shoup [14] proposed a universal hash proof in the standard model, which corresponds to adaptive-CCA secure public-key encryption. Kalai [13] proposed a two-message oblivious transfer based on modification of Cramer and Shoup's SPHF. In 2018, Benhamouda and Blazy [15] proposed a hash proof system or SPHF. It gives an SPHF under standard LWE ciphertext's languages, which is based on IND-CCA2 MP's encryption [16]. Before this SPHF proposed by Benhamouda, Katz and Vaikuntanathan [11, 17] proposed a SPHF based on lattice in the standard model, whose language is not valid in standard LWE's ciphertext. Zhang and Yu [18] proposed a SPHF based on LWE's assumption under random oracle. Brakerski [19] proposed a two-message OT based on LWE's assumption which guarantees sender's statistical privacy under the model of malicious adversary.

In 2007, Lindell and Pinkas [20] proposed cut-and-choose technique for secure two-party computation under the malicious adversary model. Circuit constructor P_1 constructs s copies' GC. Circuit evaluator P_2 chooses $s/2$ copies' GC for check circuit and remaining half $s/2$ copies' GC for evaluation circuit. P_2 checks correctness of the key's value in each wire of check garbled circuit. P_1 and P_2 apply remaining half garbled circuits for computing $f(x, y)$.

In secure two-party computation, cut-and-choose methodology can be applied to prevent malicious adversary from cheating in this process. As an important technique in secure two-party computation, cut-and-choose is applied to normalize and constrain parties for honestly executing protocols in garbled circuits. In cut-and-choose methodology, constructor P_1 constructs s copies' garbled circuits. Evaluator P_2 chooses some garbled circuits for checking. When these check circuits are correctly constructed, the evaluator applies remaining garbled circuits to evaluate corresponding function by evaluation circuits.

In this process, the OT protocol is applied for transferring corresponding key's value through wires of the garbled circuit. The OT protocol can be applied on transferring sender P_1's key to receiver P_2 through garbled circuit's wires. If these two processes are done separately, the overall protocol may lead to selective-failure attacks, which are introduced in [20, 21]. Combining cut-and-choose detection with oblivious transfer, we can transfer keys by wires between the sender (circuit constructor) and the receiver (circuit evaluator).

Lindell [22] applied cut-and-choose methodology [20] for constructing fully simulatable OT protocols. Lindell [23] proposed a new primitive in secure two-party computation, called CCOT protocol, which can avoid selective-failure attacks. After CCOT primitive proposed by Lindell, some scholars have proposed some schemes about construction of CCOT, which are mostly modified CCOT protocol, such as batch CCOT and bilateral CCOT [24–28].

Traditional OT_2^1 are applied in transferring key or message by number theory's assumption, such as DDH and QR assumption. Classical number theory assumptions cannot resist quantum attacks. It is necessary to design postquantum cryptography schemes. Considering lattice's specific linear structure, lattice-based protocols can be applied to resist quantum attacks.

There exist some cryptographic protocols based on lattice's assumption, which can resist quantum attack with the specific construction of lattice. Reduction from worst case to average case in lattice, trapdoors algorithm and some lattice theory are mentioned in [29–32].

In secure two-party computation, the CCOT protocol can resist malicious adversary's attack. Considering postquantum era, designing the CCOT protocol based on lattice assumption can resist quantum attacks. Combining with lattice theory, designing LWE-based CCOT protocol is of great significance to resist quantum attacks in secure two-party computation. Then, we can expand CCOT to batch-CCOT protocol, which can be applied on secure multiparty computation.

1.3. Our Contribution

(i) We construct a CCOT protocol based on LWE's assumption and rounding function. Applying WQ-OT [10] encryption scheme based on the rounding function and combining with PVW's dual framework [5], we design a UC-secure cut-and-choose OT protocol under the malicious adversary model.

(ii) Our CCOT protocol has better security property. For better understanding CCOT's security, we give a security analysis under the malicious adversary's corruption in smooth projective hash proof system, which is mainly based on simulation proof methodology.

(iii) In our scheme, crs can be reused many times, and all parties can achieve statistical security. The rounding function, as smooth projective hash function (SPHF), has better security in transferring P_1's garbled key to P_2. Due to the special property of the hash key and the projective key, this rounding function can guarantee CCOT protocol's correctness, privacy, and indistinguishability between the Messy mode and Dec mode.

(iv) Apply the CCOT protocol on secure two-party computation, which is mainly based on garbled circuits.

1.4. Organization

(i) In Section 1, we give an overall introduction about background, related work about the CCOT protocol. Finally, we give our contribution and paper's organization.

(ii) In Section 2, we mainly introduce some preliminaries about lattice theory and some knowledge used in scheme's construction.

(iii) In Section 3, we introduce some basic tools applied on our scheme. It includes OT-based dual-mode encryption, which is initiated by Regev's encryption and rounding function. As an important methodology in secure two-party computation, cut-and-choose technique is also introduced in this part. This dual-mode encryption's framework security is mostly based on LWE's assumption, where indistinguishability between the Messy Mode and the Dec Mode is based on DLWE's assumption.

(iv) In Section 4, cut-and-choose oblivious transfer (CCOT) protocol, as an important protocol, is applied on secure two-party computation. We construct a CCOT protocol and embed this CCOT protocol into secure two-party computation. Then, we expand CCOT to BCCOT by batch operation and embed this BCCOT protocol into secure two-party computation.

2. Preliminary

2.1. Notation. Denote n as the security parameter throughout this paper and also meaning the dimension of LWE's assumption. We denote a negligible function as in polynomial function $\mathrm{negl}(n)$, which is much smaller than the function close to zero, such as $f = n^{-c}$, where c is a positive constant close to ∞. Similarly, we denote function $1 - \mathrm{negl}(n)$ as an overwhelming function. Denote bold lowercase letter as the vector, e.g. $\boldsymbol{v}$, and denote bold uppercase letter as matrix, e.g. $\boldsymbol{M}$. Denote $a \bmod b = a - \lfloor a/b \rfloor b$ and $\lfloor a \rceil = \lfloor a + 1/2 \rfloor$. Denote $[k] = \{0, 1, \ldots, k-1\}$ as a residual class set, which can be obtained by any integers' mod integer k. Denote quotient ring $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ as a residual class set, where $\mathbb{Z}$ is modulo prime integer $q\,(q \geq 2)$. Denote $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ as group of reals $[0, 1)$, according to modulo 1 addition. Define Ψ_α as the distribution on $\mathbb{T}$, which has mean 0 and standard deviation $\alpha/\sqrt{2\pi}$. Denote v^T and M^T as an transpose operation of vector v and matrix M. Given probability distribution D, denote variable $x \longleftarrow D$ as sampling variable x from certain distribution D. Usually, denote $x \longleftarrow \mathcal{U}(\mathbb{Z})$ as sampling from uniform distribution in $\mathbb{Z}$.

2.2. Lattice Theory. Lattice, as a linear algebraic structure, can resist quantum attacks. Some lattice schemes are constructed by reduction from worst case to average case, such as reduction from SVP/CVP to LWE/SIS. Considering the size of keys and ciphertext and the structure of lattice, LWE's assumption is more used in key exchange (KE), oblivious transfer (OT), and public key encryption (PKE). And, SIS's assumption is more used in signature schemes. We apply LWE's assumption to design an OT protocol.

Lattice is a discrete additive subgroup. And, lattice is also a linear structure, which is constructed by lattice basis and integral coefficient.

Definition 1 (LWE). Learning with errors (**LWE**) assumption can be regarded as an output by an random algorithm, which outputs $(a, \langle a, s \rangle + e)$, $a, s \longleftarrow \mathcal{U}$ and $e \longleftarrow D$, such as Gaussian distribution and, centered binomial distribution. In this assumption, LWE's pairs are indistinguishable from uniform distribution. Usually, we classify LWE's assumption as SLWE and DLWE.

Definition 2 (search-LWE). Given some LWE's pairs, the probability of finding s is negligible.

Definition 3 (decision-LWE). Given some LWE's pairs and uniform pairs, it is indistinguishable from LWE's pairs to uniform pairs.

Definition 4 (Gaussian probability function). Gaussian distribution means that variable x samples from $\mathbb{R}$ based on Gaussian function. Usually, denote function $\rho_s(x) = \exp(-\pi\|x^2\|/s^2)$ with mean 0 and variance s^2.

Definition 5 (ideal lattice). Ideal lattice can be regarded as an algebraic structure based on cyclic basis, which is constructed in quotient ring $\mathbb{Z}_q$. It has some advantages, such as shortening the size of keys and ciphertext.

Definition 6 (ring-LWE). Given $a, s \in \mathbb{R}_q$, certain distribution D, and output $(a, b = \langle a, s \rangle + e) \in \mathbb{R}_q \times \mathbb{R}_q$, let us denote $A_{s,\chi}$ as the distribution of LWE's pairs.

Definition 7 (search-RLWE). Given RLWE's pairs $(a, b = \langle a, s \rangle + e) \in \mathbb{R}_q \times \mathbb{R}_q$, it is difficult in finding s.

Definition 8 (decision-RLWE). Let RLWE's pairs be sampled from distribution $A_{s,D}$. DRLWE assumption means that it is indistinguishable from $A_{s,D}$ to uniform distribution.

Given lattice basis A, we also define lattice as $\Lambda(A) = \{z \in \mathbb{Z}^m | \exists s \in \mathbb{Z}_q^n, \text{st.} z = As \bmod q\}$ and $\Lambda^{\perp}(A) = \{z \in \mathbb{Z}^m | \langle z, A\rangle = 0 \bmod q\}$, and dual lattice $\Lambda^*(A) = \{x \in \mathbb{R}^n : \forall v \in \Lambda, \langle x, v\rangle \in \mathbb{Z}\}$.

Let $\lambda_1(\Lambda)$ be the shortest nonzero vector in lattice $\Lambda(A)$, which is denoted as $\lambda_1\{\Lambda\} = \min_{x \in \Lambda, x \neq 0} \|x\|$.

Definition 9 (smoothing parameter [29]). Given n-dimensional lattice $\Lambda(A)$, positive real $\epsilon \geq 0$, and Gaussian function $\rho_{1/s}(x)$, $x \in \Lambda^* \setminus \{0\}$, define smoothing parameter $\eta_\epsilon(\Lambda)$ as the smallest s, which satisfy $\rho_{1/s}(x)(\Lambda^* \setminus \{0\}) \leq \epsilon$.

Definition 10 (noise flooding [33]). Given two integers $B, B' \in \mathbb{Z}$ and $e \in [-B', B']$, assuming that B' is negligible compared to B, satisfying $B'/B = \text{negl}(n)$. Then, following two distributions are indistinguishable between $U([-B, B])$ and $U([-B, B]) + e$, meaning that $U([-B, B])^s \approx U([-B, B]) + e$, $e \in [-B', B']$.

Lemma 1 (see [31]). *Given any n-dimensional lattice* Λ *and* $\epsilon \leq 0$, *obtain* $\eta_\epsilon(\Lambda) \leq (\sqrt{\log(2n/(1 + 1/\epsilon)\pi)}/\lambda_1(\Lambda))$. *For any* $\omega(\sqrt{\log n})$ *function, there exists negligible* $\epsilon(n)$, *satisfying* $\eta_\epsilon(\Lambda) \leq \omega(\sqrt{\log n})\lambda_1\{\Lambda^*\}$ *or* $\eta_\epsilon(\Lambda^{\perp}) \leq \omega(\sqrt{\log n})$.

Lemma 2 (see [31]). *Given* $m \geq 2n \log q$, *for any A from* $\mathbb{Z}_q^{m \times n}$, *define event* $E = \{A \text{ is full rank}\}$ *and obtain* $Pr[E] \geq 1 - q^{-n}$. *Given* $m \geq 2n \log q$, *for any A from* $\mathbb{Z}_q^{m \times n}$, *obtain* $Pr[\lambda_1(\Lambda) \geq q/4] \geq 1 - q^{-n}$. *So, we can obtain* $Pr[E \wedge \{\lambda_1(\Lambda) \geq q/4\}] \geq 1 - 2q^{-n}$.

3. Basic Tools

3.1. Dual-Mode PVW-PKE and Related PVW-OT Protocol

3.1.1. Dual-Mode PVW-PKE Encryption System. We introduce a dual-mode encryption cryptosystem proposed by Peikert et al., called PVW framework [5]. This cryptosystem is usually applied on constructing the OT protocol. It includes messy-encryption mode (or Messy mode) and decryption-encryption mode (or Dec mode).

We introduce relevant probability probabilistic algorithms, which include (Setup, KeyGen, Enc, Dec, FindMessy, TrapKeyGen) algorithms. In these algorithms, message space is $\{0, 1\}^n$ and string crs is generically common in all algorithms, and we often omit them. Next, we introduce these Algorithms in Figure 5.

Firstly, this cryptosystem can be initialized by a trusted setup phase, Setup algorithm, which outputs a string *crs* and a trapdoor t. When crs is uniformly distributed (Setup = SetupMessy), invoke Messy branch for our encryption (Setup = SetupDec). When crs is distributed by certain distribution, invoke decryption branch for our encryption. Considering the generation of crs, the property of dual-mode cryptosystem is that the distribution of crs in SeupMessy and SeupDec branch is indistinguishable.

Secondly, we invoke corresponding public key encryption (PKE) scheme, which includes KeyGen, Enc, and Dec algorithm. In key generation phase, input a branch parameter σ and output (pk, sk). The encrypter encrypts a message under chosen branch b ($b, \sigma \in \{0, 1\}$).

When $b \neq \sigma$, we denote this mode as the messy mode. In this mode, the sender encrypts the message under branch b, and the receiver decrypts ciphertext under branch σ. Apparently, the decrypter cannot obtain the corresponding message. Usually, we can use a FindMessy algorithm to find messy branch.

When $b = \sigma$, we denote this mode as Dec Mode. In this mode, the sender encrypts the message under branch b, and the receiver can correctly decrypt ciphertext under corresponding branch $\sigma(b = \sigma)$. In Dec Mode, we apply a trapdoor generation algorithm TrapKeyGen in security proof. In security proof, we should notice that it is indistinguishable between the key pair from TrapKeyGen and the key pair from KeyGen.

The p**roperties of PVW framework** are as follows:

(1) **Completeness**: for any branch $b = \sigma \in \{0, 1\}$, the receiver can correctly decrypt ciphertext, meaning $\text{Dec}(\text{sk}, \text{Enc}(\text{pk}, \sigma, m)) = m$.

(2) **Indistinguishability between two modes**: it is indistinguishable between the Messy mode and the Dec mode, which is mainly indistinguishable between crs_M and crs_D.

(3) **The property of messy mode**: given (crs_M, t_M) from SetupMessy and any public key pk (including malformed pk) from the key generation phase under corresponding mode, invoke $\text{FindMessy}(t_M, \text{pk})$ algorithm to obtain messy branch $b\prime$. In Messy mode, it can obtain statistical security, which can hide some information about ciphertext, meaning $\text{Enc}(\text{pk}, b\prime, m_0)^s \approx \text{Enc}(\text{pk}, b\prime, m_1)$.

(4) **The property of decryption mode**: it is indistinguishable between key pairs generated by $\text{KeyGen}(\sigma)$ and key pairs generated by TrapKeyGen, meaning $(\text{pk}, \text{sk}_\sigma)^s \approx \text{KeyGen}(\sigma)$.

3.1.2. PVW-OT Framework. Peikert proposed $\mathbf{dm}^{\text{mode}}$ protocol in Figure 6, which applies any mode in the dual-mode encryption system under the $\mathcal{F}_{\text{CRS}}$-hybrid UC model [6]. The $\mathbf{dm}^{\text{mode}}$ protocol achieves the function of ideal F_{OT} in Figure 1.

To achieve the messy and decryption mode, define $\mathcal{F}_{\text{CRS}}^{\text{mode}}$ to produce common string, which corresponds to relevant setup algorithm.

Lemma 3 (see [5]). *In static corruption model, the protocol* $\mathbf{dm}^{mode}$ *securely emulates ideal function* F_{OT} *in the universal composable* $\mathcal{F}_{CRS}^{mode}$*-hybrid model.*

3.2. Dual-Mode WQ-PKE and Related WQ-OT Protocol. UC-OT based on PVW's framework [5] can provide sender's statistical security and receiver's computational security in the

Setup$(\mathbf{1^n}, \mathbf{b})$: input parameter n and choosen branch b, output (crs, t). • When $b = 0$, we denote $Setup(1^n, 0) = SetupMessy(1^n)$. • When $b = 1$, we denote $Setup(1^n, 1) = SetupDec(1^n)$. • These two setup algorithm can generate two types crs and trapdoor t, called (crs_M, t_M) and (crs_D, t_D). Common string crs is generically known in all algorithms, and we often omit them. Trapdoor t is used in $FindMessy, TrapKeyGen$ algorithm. • When $b = 0$, trapdoor t_M is used in $FindMessy$ to find Messy branch. • When $b = 1$, trapdoor t_D is used in $TrapKeyGen$ for applying on security proof.
KeyGen(σ): Receiver chooses a branch σ, and inputs σ to $KeyGen$ algorithm to obtain a pair of keys (pk, sk). Public key pk is used to encrypt message by Sender, and sk is the corresponding secret key used to decrypt ciphertext by Receiver.
Enc$(\mathbf{pk}, \mathbf{b}, \mathbf{m})$: Sender encrypts message m under pk in branch b to obtain ciphertext c.
Dec$(\mathbf{sk}, \mathbf{c})$: Receiver decrypts ciphertext c under the secret key sk.
FindMessy$(\mathbf{t}, \mathbf{pk})$: Given some public key pk, input t and pk, output Messy branch, which is relevant with corresponding branch.
TrapKeyGen$(\mathbf{t})$: Input trapdoor t, output (pk, sk_0, sk_1), which pk is a base public key, sk_0 and sk_1 are corresponding secret key in branch 0 and 1.

Figure 5: Dual-mode PVW-PKE algorithms.

Protocol $\mathbf{dm}^{mode}$ has parameter $mode \in \{ext, dec\}$ to decide crs's type.
Input: Sender S inputs (sid, m_0, m_1), Receiver R inputs (sid, σ), $\sigma \in \{0, 1\}$;
Protocol Process:
When activated with sender and receiver's input, sender S queries $\mathcal{F}_{CRS}^{mode}$ to obtain (sid, crs). Similarly, receiver R queries $\mathcal{F}_{CRS}^{mode}$ to obtain (sid, crs);
Receiver R invokes $KeyGen(crs, \sigma)$ to obtain (pk, sk), stores secret key sk in storage unit, then sends public key (sid, pk) to Sender S;
Sender S obtains (sid, pk) from R, then invokes Enc algorithm to get $c_0 \leftarrow Enc(pk, 0, m_0)$ and $c_1 \leftarrow Enc(pk, 1, m_1)$. Next S sends c_0 and c_1 to receiver R;
Receiver R obatins (c_0, c_1) from S, then invokes Dec algorithm to obtain $m_\sigma \leftarrow Dec(sk, c_\sigma)$;
Output: Sender outputs $\perp$, Receiver outputs m_σ.

Figure 6: dm^{mode} for oblivious transfer.

messy mode. In Dec mode, sender's security is computational and receiver's security is statistical. This construction can only provide receiver's computational security in Messy mode and sender's computational security in Dec Mode. In addition, it has bounded limitation about reusability of crs.

Considering about these limitations, apply superpolynomial LWE modulus and single 'short' crs and then achieve statistical security in both mode and unbounded crs's reusability.

Apply the WQ-OT scheme [10] for our CCOT's construction. WQ-OT is a two-round UC-OT based on Common References String (crs), and it is based on LWE assumption with subexponential modulus-to-noise ratio.

Considering noise flooding technique is applied to strengthen reusability and statistical security, we need superpolynomial modulus q of LWE. However, this operation has negative impact on the security proof. And, the simulator of PVW-OT operates in linear q time, not superpolynomial, due to negative impact on security proof. For resolving this difficulty, apply randomized rounding function to PKE-based OT framework. Benhamouda [15] et al. proposed a rounding function.

Considering about security proof, we apply the hash proof system, which mainly refers to lattice-based SPHFs. In the following, we will introduce a rounding function, which is viewed as an approximate hash proof system. Given $c = As + e \in \mathbb{Z}_q^m$, $A \in \mathbb{Z}_q^{m\times n}$, $s \in \mathbb{Z}_q^n$, $e \in \mathbb{Z}_q^m$, and 'c' as a vector is close to $\Lambda(A)$, the prover knows s and e and the prover needs to prove 'c' is the corresponding ciphertext for the verifier. The verifier samples a uniformly random vector $r \longleftarrow D_{\mathbb{Z},s}^m$. Let r be a hash key, and compute $p = A^T r$ as a projection key. The verifier sends projection key p to the prover, and the prover computes projection hash value $pH = R(\langle p, s\rangle) = R(p^T s) = R(r^T As)$. The verifier computes hash value $H = R(\langle r, c\rangle) = R(r^T c)$. Then, the verifier sends H to the prover. The prover checks whether $H = pH$ to ensure the verifier is honest; then, the verifier approves the prover's proof. This progress is zero knowledge. The prover has not revealed secret information s and e.

The high probability of $H = pH$ implies the property of approximate correctness, which needs vector 'c' close to lattice $\Lambda(A)$, with distance less than B. For applying the approximate hash proof system better, the property of smoothness needs point 'c' far from $\Lambda(A)$, with minimum distance $q\sqrt{m}/s$.

3.2.1. Rounding Function. We describe a suitable q-periodic signal function, rounding function R, as follows. Given $m = \Theta(n \log q)$, define the rounding function R: $\mathbb{Z}_q \longrightarrow \{0, 1\}$ as follows:

$$R(x) = \lfloor \frac{2x}{q} \rfloor \bmod 2 \begin{cases} 1, & Pr = \frac{1}{2} + \frac{\cos(2\pi x/q)}{2}, \\ 0, & Pr = \frac{1}{2} - \frac{\cos(2\pi x/q)}{2}. \end{cases} \quad (1)$$

(i) **Smoothness**: given full rank matrix $A \in \mathbb{Z}_q^{n\times m}$, $s \geq \eta_\varepsilon(\Lambda(A))(\varepsilon = \mathrm{negl}(n))$, for all $c \in \mathbb{Z}_q^m$ satisfying $\|c, \Lambda(A)\| \geq q\sqrt{m}/s$, achieve $Pr[R(r^T c) = 1 | r^T A = p^T] \leq 1/2 + \mathrm{negl}(n)(r \in D_{\mathbb{Z},s}^m)$

Approximate correctness: given $s \in \mathbb{Z}_q^n, e \in \mathbb{Z}_q^m, B = O(q)/\sqrt{m}s$, for all $c = As + e$, satisfying $\mathrm{dist}(c, \Lambda(A)) \leq B$, achieve $Pr[R(r^T As) = R\ (r^T c)] \geq 2/3(r \in D_{\mathbb{Z},s}^m)$

3.2.2. Smooth Projective Hash Function Encryption System. Apply this rounding function on our encryption system, which is likely Regev's Encryption. Define this modified encryption system as the smooth encryption system. For better understanding the process of the smooth encryption system, we introduce the encryption process of single-bit message $u \in \{0, 1\}$ in Figure 7.

(i) **Parameters**: denote n as the security parameter in the whole scheme. Let prime integers $q \geq 2$ be modulus. Let $m \geq 2(n+1)\log q$ and $s \geq 4\sqrt{m}$. Distribution χ is a B-bounded distribution $(B \geq \Omega(\sqrt{n}))$. Usually, set $B = \Omega(\sqrt{n})$; the value of B is a security limitation value, which guarantees LWE's assumption for schemes. B' is a negligible value compared to B, satisfying $q \leq w(B' + \sqrt{n})m$.

(ii) **Correctness**: given $(B + B') \cdot s \cdot \sqrt{m} = O(q)$ and $s \geq \omega(\sqrt{\log m})$, decryption algorithm $\mathrm{Dec}(\mathrm{sk}, ct)$ can correctly decrypt with nonnegligible probability.

Security: given $m \geq 2(n+1)\log q$ and $s \geq 4\sqrt{m}$, the smooth encryption scheme based on LWE's assumption can achieve corresponding security.

3.2.3. WQ-OT Framework. We give a brief introduction about dual-mode encryption based on the SPHF encryption system in Figure 8, which is based on the hash key and projection key of SPHF-rounding function.

(1) **Completeness**: given $s \geq \omega(\sqrt{\log m})$ and $(B + B') \cdot s \cdot \sqrt{m} = O(q)$, then the scheme can correctly decrypt in the Dec mode.

(2) **Indistinguishability between Messy mode and Dec mode**: given string crs to any adversary, adversary cannot distinguish between the Messy mode with the Dec mode, implying $\mathrm{SetupMessy}^*(1^n)^c \approx \mathrm{SetupDec}^*(1^n)$. This indistinguishability implies the indistinguishability between LWE's vector pairs with uniform vector pairs.

(3) **Sufficient conditions for Messy key**: given public key pk in the smooth encryption system, it satisfies $\mathrm{SmoothEnc}^*(\mathrm{pk}, 0)^s \approx \mathrm{SmoothEnc}^*(\mathrm{pk}, 1)$. Given full rank uniform matrix $A \longleftarrow \mathbb{Z}_q^{m\times n}$ and $c \in \mathbb{Z}_q^m$, when $s \geq \eta_\epsilon(\Lambda(A))$ and $d(c, \Lambda(A)) \geq q\sqrt{m}/s$, then public key $\mathrm{pk} = (A, c)$ achieves the property of Messy key.

(4) **Requirements for Messy key**: given parameter $m \geq 2(n+1)\log q$, $s \geq 4\sqrt{m}$, and sample (A, c) from $\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$, satisfying $d(c, \Lambda(A)) \geq q/4$, we obtain messy public key (A, c) with nonnegligible probability.

(5) **Verification of Messy key**: given $s \geq 6m$ and (A, T) generated by $\mathrm{TrapGen}^*(1^n, 1^m, q)$, which A is a full rank matrix, invoke $\mathrm{IsMessy}^*(T, A, c)$ algorithm, which inputs vector c to decide distance between c and lattice $\Lambda(A)$. When $d(c, \Lambda(A)) \geq q\sqrt{m}/s$, output Messy public key (A, c) with overwhelming probability.

(6) **Messy mode's property**: given $s \geq 6m$ and $m \geq 2(n+1)\log q$, then the scheme achieves security in the Messy mode, implying $\mathrm{SmoothEnc}^*(\mathrm{crs}_M^*, \mathrm{pk}_M, 0)^s \approx \mathrm{SmoothEnc}^*(\mathrm{crs}_M^*, \mathrm{pk}_M, 1)$.

(7) **Dec**: given B', satisfying $B/B' = \mathrm{negl}(n)$, then the scheme achieve security in the Dec mode, implying the indistinguishability between $(\mathrm{pk}_0^*, \mathrm{sk}_b^*)$ generated from $\mathrm{KeyGen}^*(\mathrm{crs}_D^*, b)$ and $(\mathrm{pk}_0, \mathrm{sk}_b)$ generated from $\mathrm{TrapKeyGen}^*(td_D^*)$ in security proof.

Lemma 4 (see [10]). *Given $LWE_{q,\lambda,n}$ assumption with the corresponding parameter in WQ-OT's construction, an dual-*

KeyGen*: Given $A \in \mathbb{Z}_q^{n\times m}, s \in \mathbb{Z}_q^n, e \in \chi^m, f \in [-B^{'}, B^{'}]$, Let $c = As + e + f$, then output public key $pk = (A, c)$ and secret key $sk = s$. Output a pair of key (pk, sk).
Enc***(pk, u)**: For $i \in [1, 2, \cdots, n]$, sample hash key $r_i \in D_{\mathbb{Z},s}^m$. Encrypter computes projection key $p_i^T = r_i^T A$ and compute $\beta_i \leftarrow R(r_i^T A) \oplus u$, then output $ct = \{p_i, \beta_i\}, i \in [1, 2, \cdots, n]$.
Dec***(sk, ct)**: Decrypter applys projection key p_i and secret key s, then computes $b_i \leftarrow R(p_i^T s) \oplus \beta_i$. Finally, output b_i.

Figure 7: SPHF encryption algorithms based on the rounding function.

Parameter*: Let $s \geq 6m$, others parameter same as in Smooth Encryption system.
Setup***(1^n, b)**: Generate crs and trapdoor t. Corresponding to two mode, give two initialization types, as follows. **SetupMessy*****(1^n)**: Apply trapdoor generation algorithm to obtain (A, T). Sample uniform vector $c \leftarrow \mathbb{Z}_q^m$. Then output $crs_M^* = (A, V)$ and trapdoor $td_M^* = T$. **SetupDec*****(1^n)**: Sample uniform matrix $A \in \mathbb{Z}_q^{m\times n}$, $s' \in \mathbb{Z}_q^n$, and $e' \in \chi^m$, compute $v = As' + e'$. Then output $crs_D^* = (A, v)$ and trapdoor $td_D^* = s'$.
KeyGen***(crs, b)**: Sample uniform vector $s \in \mathbb{Z}_q^n$, $e \in \chi^m$, and $f \in [-B', B']$. Then output public key $pk_0 = As + e + f - b \cdot v$ and secret key $sk_b = s$. Let $pk_b = As + e + f$ and $pk_1 - pk_0 = v$. Output (pk_0, sk_b).
Enc***(crs, pk$_0$, b$'$, u)**: Let $pk_{b'} = c = pk_0 + b'v$. Choose random $r_i \in D_{\mathbb{Z},s}^m, i \in \{1, 2\cdots, n\}$, compute $p_i^T = r_i^T A \in \mathbb{Z}_q^{1\times n}$, then apply XOR operation to obtain $\beta_i \leftarrow R(r_i^T c) \oplus u$. Finally output ciphertext $ct_i = \{(p_i, \beta_i)\}$.
Dec***(crs, sk$_b$, ct$_i$)**: Firstly, parse ciphertext ct_i as (p_i, β_i). Apply projection key p_i to obtain $u' \leftarrow R(p_i^T s) \oplus \beta_i$. After repeated many times, output the majority bit of u'.
FindMessy***(td$_M^*$)**: Invoke lattice trapdoor algorithm $IsMessy^*(pk_0)$ to obtain b^*. Output $b^* = 0$, represents branch b^* as Messy branch. Otherwise, $b^* = 1$ represents Decryption branch. Output b^*.
TrapKeyGen***(td$_D^*$)**: Sample uniform vector $s \in \mathbb{Z}_q^n$, $e \in \chi^m$, and $f \in [-B', B']$, compute $pk_0 = As + e + f$. Then output pk_0 and $sk_0 = s, sk_1 = s + s'$. Output (pk_0, sk_0, sk_1).

Figure 8: Dual-mode WQ-PKE algorithms.

mode UC-secure oblivious transfer protocol under static corruption exists.

4. Cut-and-Choose Oblivious Transfer

Cut-and-choose oblivious transfer (CCOT) protocol can be applied on secure two-party computation, which transfers circuit constructor's garbled keys of wires to the circuit evaluator. Firstly, we introduce the ideal function of CCOT in secure two-party computation.

4.1. Ideal Function of CCOT. Lindell presented the concept of CCOT, which is an oblivious transfer protocol combined with cut-and-choose index bit. We give a brief introduction about its ideal function F_{CCOT}. Circuit constructor P_1 constructs one garbled circuit; circuit evaluator P_2 decides to obtain two key's value of each wire or one key's value of two wires, which is based on the index bit $j \in \{0, 1\}$. When $j = 0$, P_2 wants to obtain both key's value; when $j = 1$, P_2 wants to obtain one key's value from two keys. We give a brief introduction about the ideal function in Figure 9.

Input:

- Sender (or Circuit Constructor P_1) inputs $y_0, y_1 \in \{0,1\}^n$;
- Receiver (or Circuit Evaluator P_2) sends $(j, \sigma) \in \{0,1\}$ to Sender. j is index bit for checking two key's value or obtaining one key's value by OT_2^1 process. σ is the choosen bit in Obilivious Transfer's process.

Output:

- Sender outputs $\perp$;
- When $j = 0$, Receiver obtains (y_0, y_1); When $j = 1$, Receiver obtains y_σ.

Figure 9: The ideal function F_{CCOT}.

4.2. Construction of CCOT Protocol. Cut− and− choose oblivious *t*ransfer (CCOT) is a new primitive for secure two-party computation. For better understanding the role of CCOT protocol in secure two-party computation, we give a general introduction about CCOT protocol's construction.

Firstly, we give the construction of the CCOT protocol corresponding to a single garbled circuit in Figure 10.

Given setup algorithm SetupMesssy and SetupDec algorithm, σ is selected bit from the receiver's input.

(1) The receiver R is initialized with index bit j. When $j = 0$, R invokes KeyGen* (crs, b) algorithm to obtain $(\text{pk}_0, \text{sk}_b)$, which is $\text{pk}_0 = As + e + f - b \cdot v$, $\text{sk}_b = s$, $\text{pk}_1 = \text{pk}_0 + v$, and $\text{pk}_b = As + e + f$.

When $j = 1$, R invokes KeyGen* algorithm to obtain $(\text{pk}_0, \text{sk}_\sigma)$, which is $\text{pk}_0 = As + e + f - \sigma \cdot v$, $\text{pk}_1 = \text{pk}_0 + v$, $\text{sk}_\sigma = s$, and $\text{pk}_\sigma = As + e + f$.

(2) R sends relevant public key to sender S in corresponding index bit. When $j = 0$, R sends $(A, \text{pk}_0, \text{pk}_1)$ to S. S encrypts message y_0 and y_1 under pk_0 and pk_1. S samples $r_b \longleftarrow D^m_{\mathbb{Z},r}$ and computes $p_b^t = r_b^t \cdot A$; then, we obtain $\beta_i \longleftarrow R\ (r_b^t \cdot \text{pk}_i) \oplus y_i, i \in \{0,1\}$. Finally, S sends $ct_i = (p_b, \beta_i)$ to R.

When $j = 1$, R sends $(A, \text{pk}_0, \text{pk}_1)$ to S. S encrypts message y_0 and y_1 under pk_0 and pk_1. S samples $r_b \longleftarrow D^m_{\mathbb{Z},r}$ and computes $p_b^t = r_b^t \cdot A$; then, we obtain $\beta_i \longleftarrow R(r_b^t \cdot \text{pk}_i) \oplus y_i, i \in \{0,1\}$. Finally, S sends $ct_i = (p_b, \beta_i)$ to R.

(3) When $j = 0$, receiver R receives ct_i from sender S. R parses ct_i as (p_b, β_i). R invokes Dec* (sk_b, ct_i) to obtain y_i by computing $y_i \longleftarrow R(p_b^t \cdot \text{sk}_b) \oplus \beta_i$. Finally, R obtains y_0 and y_1.

When $j = 1$, R receives ct_i from S. R parses ct_i as (p_b, β_i). R invokes Dec* (sk_σ, ct_i) to obtain y_σ by computing $y_i \longleftarrow R(p_b^t \cdot \text{sk}_\sigma) \oplus \beta_i$.

Lemma 5 (correctness [10]). *Given $s \geq \omega(\sqrt{\log m})$ and $(B + B') \cdot s \cdot \sqrt{m} = O(q)$, then the scheme can correctly decrypt in Dec branch.*

In the Dec mode, given the corresponding parameter, due to rounding function's property, the receiver can correctly decrypt corresponding two ciphertext. In the Messy mode, when $\sigma = 0$, $\text{pk}_0 = As + e + f$, and secret key $\text{sk}_\sigma = \text{sk}_0 = s$ can be used to decrypt ciphertext ct_0. In the meantime, $\text{pk}_0 = As + e + f + v$ and secret key $\text{sk}_\sigma = \text{sk}_0 = s$ cannot be used to decrypt corresponding ciphertext ct_1. When $\sigma = 1$, $\text{pk}_1 = As + e + f$, and secret key $\text{sk}_\sigma = \text{sk}_1 = s$ can be used to decrypt ciphertext ct_1. However, $\text{pk}_0 = As + e + f - v$, and secret key $\text{sk}_\sigma = \text{sk}_0 = s$ cannot be used to decrypt corresponding ciphertext ct_0.

The correctness of this protocol is shown in Table 1, which is similar to the SmoothEnc System.

4.2.1. Security Proof

Theorem 1. *Given $s \geq 6m$ and $m \geq 2(n+1)\log q$, the above scheme is UC-secure CCOT protocol under static malicious adversary, assuming the hardness of learning with errors with the corresponding parameter.*

We mainly consider two corruption cases; the sender is corrupted; the receiver is corrupted. **Sender is corrupted**: firstly, we consider adversary $\mathcal{A}$ corrupt sender P_1, and we need to construct a simulator $\mathcal{S}$, who can invoke adversary $\mathcal{A}$'s input copies and make some operations as follows.

when $j = 0$, run Dec mode's setup algorithm SetupDec* and obtain $(\text{crs}_0, t_0) \longleftarrow \text{SetupDec}^*(1^n)$, which is well known to all parties. Honest receiver P_2 runs KeyGen* algorithm to obtain $(\text{pk}_0, \text{pk}_1, \text{sk}_b)$ and then P_2 sends pk_0 and pk_1 to sender P_1. Corresponding this situation, simulator $\mathcal{S}$ invokes TrapKeyGen* (crs_0, t_0) algorithm to obtain $(\text{pk}, \text{sk}_0, \text{sk}_1)$ and then $\mathcal{S}$ sends (sid, pk) to adversary $\mathcal{A}$, which simulates a interactive scenario between receiver P_2 and adversary $\mathcal{A}$. Then, simulator $\mathcal{S}$ stores $(\text{sk}_0, \text{sk}_1)$. Adversary $\mathcal{A}$ invokes Enc* (pk, y_0, y_1) algorithm to obtain ciphertext (ct_0, ct_1) and $\mathcal{A}$ sends (ct_0, ct_1) to receiver P_2. Simulator $\mathcal{S}$ simulates this process; Simulator $\mathcal{S}$ receives (ct_0, ct_1) from adversary $\mathcal{A}$; $\mathcal{S}$ checks corresponding secret key sk_0 and sk_1 and invokes Dec* $(\text{sk}_b, ct_b), b \in \{0,1\}$

Sender′s Input:
Sender inputs $Y = (y_0, y_1)$, $y_i \in \{0,1\}^n$. Receiver inputs index bit j and choosen bit $\sigma \in \{0,1\}$.
Auxiliary Input:
Input a prime integer modulus q. Input number of dimension n and m.
Setup Phase:

- For $j = 0$, invoke $SetupDec^*$ algorithm to obtain (crs_0, td_0). Receiver uniformly samples $A_0 \leftarrow \mathbb{Z}_q^{m\times n}$, then samples secret key $s_0 \leftarrow \mathbb{Z}_q^n$, error vector $e_0 \leftarrow \chi^m$, where χ distribution is always mentioned Gaussian Distribution. Compute $v_0 = A_0 s_0 + e_0$, obtain $crs_0 = (A_0, v_0)$ and trapdoor about check-circuit $td_0 = s_0$, which crs_0 is well-known to all parties.
- For $j = 1$, invoke $SetupMessy^*$ algorithm to obtain (crs_1, td_1). Receiver invokes $TrapGen^*(n, m, q)$ algorithm to obtain (A_1, T_1). Then Receiver uniformly picks v_1 from $\mathbb{Z}_q^m$ to obtain $crs_1 = (A_1, v_1)$ and trapdoor about evaluation-circuit $td_1 = T_1$, which crs_1 is well-known to all parties and td_1 is only for Receiver.

Transfer Phase:

- When $j = 0$, Sender sends check-circuit to Receiver. Receiver input index $j = 0$, which represents to obtain both message y_0 and y_1. Receiver invokes $KeyGen^*(crs, b)$ algorithm to obtain (pk_0, sk_b), which is $pk_b = As + e + f$, $pk_0 = As + e + f - b \cdot v$, $sk_b = s$ $(s \leftarrow \mathbb{Z}_q^n, e \leftarrow \chi^m, f \leftarrow [-B', B'])$. Receiver sends pk_b to Sender. Sender invoke $Enc^*(crs, pk_b, y_i)$ $(i \in \{0,1\})$algorithm. Sender samples $r_b \leftarrow D_{\mathbb{Z},r}^r$ and compute $p_b^t = r_b^t A \in \mathbb{Z}_q^{1\times n}$. Sender invokes Rounding function to obtain ciphertext $\beta_i \leftarrow R(r_b^t \cdot pk_b) \oplus y_i$, $ct_i = (p_b, \beta_i)$ and sends ct_i to Receiver. Finally, Receiver invokes $Dec^*(crs, sk_b, ct_i)$ algorithm. Receiver parse ct_i as (p_b, β_i), then compute $y_i \leftarrow R(p_b^t \cdot sk_b) \oplus \beta_i$. Finally, Receiver obtains message y_0 and y_1.
- When $j = 1$, Sender sends evaluation-circuit to Receiver. Firstly, Receiver input choosen bit $\sigma \in \{0,1\}$. Receiver invokes $KeyGen^*(crs, \sigma)$ to obtain a pair of keys (pk_0, sk_σ), which is $pk_\sigma = As + e + f$ and $sk_\sigma = s$. For this base public key pk_σ, there are two derived public keys $pk_0 = As + e + f - \sigma \cdot v$ and $pk_1 = pk_0 + v$. Secondly, Sender encrypts message y_0, y_1 under pk_0 and pk_1, which is $Enc^*(crs, pk_0, b, y_b), b \in \{0,1\}$. Invoke algorithm $Enc^*(pk_\sigma, b, y_b)$ to obtain $ct_b = (\{p_b, \beta_b\}), b \in \{0,1\}$, sample $r_b \leftarrow D_{\mathbb{Z},r}^m$, $p_b^t = r_b^t \cdot A$ and compute $\beta_b \leftarrow R(r_b^t \cdot pk_b) \oplus y_b, b \in \{0,1\}$. Then Sender sends c_b to Receiver. Sender sends these values to Receiver. Considering Receiver's choosen bit σ, when $b = \sigma$, Receiver can decrypts $y_\sigma \leftarrow Dec_{sk_\sigma}^*(\beta_b)$, which only $b = \sigma$ is satisfied, Receiver can correctly decrypt. Receiver invokes $Dec^*(sk, \beta_b)$ algorithm, which is mainly obtained by Rounding function's projection key, $y_i \leftarrow R(p_b^t \cdot sk_\sigma) \oplus \beta_b, sk_\sigma = s$, Finally, Receiver obtains corresponding message y_σ.

Figure 10: CCOT protocol in secure two-party computation.

Table 1: Protocol's correctness.

(j, σ)	Input	Output
(0, σ)	$(\mathrm{crs}_0, td_0), \mathrm{crs}_0 = (A_0, v_0), v_0 = A_0 s_0 + e_0, td_0 = s_0$	$y_0 \longleftarrow R(p_0^t \cdot \mathrm{sk}_0) \oplus \beta_0,\ y_1 \longleftarrow R(p_1^t \cdot \mathrm{sk}_1) \oplus \beta_1$
(1, 0)	$(\mathrm{crs}_1, td_1), \mathrm{crs}_1 = (A_1, v_1), v_1 = A_1 s_1 + e_1, td_1 = s_1$	$y_0 \longleftarrow R(p_0^t \cdot \mathrm{sk}_0) \oplus \beta_0$
(1, 1)	$(\mathrm{crs}_1, td_1), \mathrm{crs}_1 = (A_1, v_1), v_1 = A_1 s_1 + e_1, td_1 = s_1$	$y_1 \longleftarrow R(p_1^t \cdot \mathrm{sk}_1) \oplus \beta_1$

Input:

- Sender inputs $Y_1 = (y_0^1, y_1^1)$, $Y_2 = (y_0^2, y_1^2)$, $\cdots$, $Y_s = (y_0^s, y_1^s)$;
- Receiver input choosen bit $\sigma_1, \sigma_2, \cdots, \sigma_s \in \{0, 1\}$ and index bit set $J \subset [1, 2, \cdots, s]$, $|J| = s/2$.

Output:

- Firstly, check whether $|J| = s/2$, if the length of J is $s/2$, continue to run this protocol. Otherwise, terminate.
- Sender outputs $\perp$;
- When $j \in J$, Receiver obtains (y_0^j, y_1^j); Otherwise, $j \notin J$, Receiver obtains $y_{\sigma_j}^j$.

Figure 11: The ideal function F_{BCCOT}.

algorithm to obtain $y_b', b \in \{0, 1\}$. Simulator $\mathcal{S}$ sends (y_0', y_1') to ideal function F_{ccot}. Receiver P_2 obtains the corresponding message based on mode index j and selection bit index σ.

Receiver is corrupted: adversary $\mathcal{A}$ corrupts receiver P_2, and we need to construct a simulator $\mathcal{S}$, who simulates the process between adversary $\mathcal{A}$ and sender P_1.

When $j = 1$, run Messy mode's setup algorithm SetupMessy* to obtain $(crs_1, t_1) \longleftarrow \mathrm{SetupMessy}^*(1^n)$, where crs_1 is well known to all parties. We should notice that the distribution of crs_1 is uniformly sampled. Honest sender P_1 interacts with P_2 corrupted by adversary $\mathcal{A}$. Adversary $\mathcal{A}$ chooses a selection bit σ and then invokes $\mathrm{KeyGen}^*(\mathrm{crs}_1, \sigma)$ algorithm to obtain $(\mathrm{sid}, \mathrm{pk}_\sigma, \mathrm{sk}_\sigma)$, which is public to sender P_1. Simulator $\mathcal{S}$ invokes $\mathrm{FindMessy}^*(\mathrm{crs}_1, t_1, \mathrm{pk}_\sigma)$ to obtain messy branch b. Then, simulator $\mathcal{S}$ sends $(\mathrm{sid}, \mathrm{receiver}, 1 - b)$ to ideal function of F_{OT}. Then, simulator $\mathcal{S}$ receives corresponding message y_{1-b}. Simulator $\mathcal{S}$ simulates the process between P_1 and $\mathcal{A}$.

Firstly, $\mathcal{S}$ computes $ct_b \longleftarrow \mathrm{Enc}^*(\mathrm{pk}_b, 0^n)$ and $ct_{1-b} \longleftarrow \mathrm{Enc}^*(\mathrm{pk}_{1-b}, y_{1-b})$. Secondly, $\mathcal{S}$ sends (ct_b, ct_{1-b}) to adversary $\mathcal{A}$, as if from sender P_1's value. Finally, adversary $\mathcal{A}$ decrypts ct_b with corresponding secret key sk_σ. Considering LWE's assumption and Messy mode's property, adversary $\mathcal{A}$ cannot obtain the correct message.

4.3. Ideal Function of BCCOT. In secure two-party computation, Garbled circuits' constructor P_1 constructed many copies circuits, which can resist one situation about P_1 constructing incorrect circuit. Let s be garbled circuits' parameter. P_1 constructs s copies circuits, half of these circuits are used for checking and another half are used for evaluation. After all check circuits pass detection, remaining half circuits are used for evaluating $f(x, y)$. Some garbled circuit maybe incorrectly constructed by P_1, and we can adopt majority value of $f(x, y)$. Considering circuits' parameter s, we introduce a 'batch cut-and-choose' oblivious transfer's ideal function F_{bccot} in Figure 11. In a special case, when $\sigma_1 = \sigma_2 = \cdots = \sigma_s = \sigma$, $j \notin J$, the receiver obtains y_σ^j, and this is single-choice CCOT function.

4.4. Construction of BCCOT Protocol. Considering that s copies of garbled circuits used in secure two-party computation based on CCOT protocol, BCCOT can be regarded as a series of oblivious transfer protocol, which can be implemented by batch operation.

Firstly, we give the construction of BCCOT protocol corresponding to s copies' garbled circuits in Figure 12. Sender P_1 constructs s copies' garbled circuits. Each circuit has $2n$ input wires, in which half of them are for P_1's input

Sender′s Input:
Sender P_1 inputs s copies circuit values, $Y_1 = (y_{01}, y_{11}), Y_2 = (y_{02}, y_{12}), \cdots, Y_i = (y_{0i}, y_{1i}), \cdots, Y_s = (y_{0s}, y_{1s})$.
Receiver′s Input:
Receiver P_2 input index bit set $J = (j_1, j_2, \cdots, j_{s/2})$ $(j_i \in \{0,1\})$ and selection bit set $S_\sigma = \sigma_1, \sigma_2, \cdots, \sigma_s$ $(\sigma_i \in \{0,1\})$.
Auxiliary Input:
Input a prime integer modulus q, value of dimension n and m, some parameter mentioned in *SmoothEncryption* System.
Setup Phase:

- Firstly, check whether $|J| = s/2$, otherwise, output $\perp$ and terminate this protocol.
- For $j_i = 0$, invoke Dec^* algorithm to obtain (crs_{0i}, td_{0i}), P_2 uniformly samples $A_{0i} \leftarrow \mathbb{Z}_q^{m\times n}, s_{0i} \leftarrow \mathbb{Z}_q^n, e_{0i} \leftarrow \lambda^m$, and computes $v_{0i} = A_{0i}s_{0i} + e_{0i}$. Then P_2 obtain $crs_{0i} = (A_{0i}, v_{0i})$ and value of trapdoor $td_{0i} = s_{0i}$, which crs_{0i} is public to all parties.
- For $j_i = 1$, invoke $SetupMessy^*$ algorithm to obtain (crs_{1i}, td_{1i}). P_2 invokes $TrapGen^*(n, m, q)$ algorithm to obtain (A_{1i}, T_{1i}). Receiver P_2 uniformly samples v_{1i} from $\mathbb{Z}_q^n$. Output $crs_{1i} = (A_{1I}, v_{1i})$, which is public to allparties.

Transfer Phase:

- Receiver P_2 sends $j_i = 0$ and selection bit σ_i to Sender P_1, which means to obtain message $Y_i = (y_{0i}, y_{1i})$. Sender P_1 constructs check-circuit for P_2, and P_2 obtain y_{0i} and y_{1i}, which correspond to each circuit. Receiver P_2 invokes $KeyGen^*(crs_{0i}, b_i)$ algorithm to obtain (pk_{0i}, sk_{b_i}), which is $pk_{0i} = A_{0i}s_{0i} + e_{0i} + f_{0i} - b_i v_{0i}, sk_{b_i} = s_{0i}, pk_{b_i} = A_{0i}s_{0i} + e_{0i} + f_{0i}, pk_{1i} = pk_{0i} + v_{0i}$. P_2 sends pk_{0i} to P_1, then P_1 invokes $Enc^*(crs_{0i}, pk_{b_i}, Y_i)$ to obatin (ct_{0i}, ct_{1i}). Then P_2 applys for $Dec^*_{sk_{b_i}}$ algorithm ang Rounding function System's projection key to obtain message y_{0i} and y_{1i}.
- Receiver P_2 sends $j_i = 1$ and selection bit $\sigma_i \in \{0,1\}$ to Sender P_1, it means that Receiver P_2 want to obtain corresponding message $y_{\sigma_i i}$ from Sender P_1. Sender P_1 constructs evaluation-circuit for P_2, and P_2 obtain $y_{\sigma_i i}$, which corresponds to selection wire's values. Receiver P_2 invokes $KeyGen^*(crs_{1i}, \sigma_i)$ to obtain a pair of key (pk_{1i}, sk_{1i}), which is $pk_{1i} = A_{1i}s_{1i} + e_{1i} + f_{1i}$ and $sk_{1i} = s_{1i}$. Based on these basic public key, there are two derived public key $pk_{\sigma_i,0} = pk_{1i} - \sigma \cdot v$ and $pk_{\sigma_i,1} = pk_{\sigma_i,0} + v$. P_2 sends these pubic keys to P_1, then P_1 encrypts y_{0i} and y_{1i} under corresponding public key. This process is similar to single *CCOT*'s process, which is mainly based on Rounding function's hash key and projection key in *SmoothEncryption* System. Sender P_1 sends ciphertext ct_b and ct_{1-b} to Receiver P_2. P_2 invokes $Dec^*(sk_\sigma, ct_b)$ and $Dec^*(sk_\sigma, ct_{1-b})$ algorithm to obtain corresponding message $y_{\sigma_i,i}$. When $b = \sigma_i$, P_2 can obtain corresponding value.

Figure 12: BCCOT protocol in secure two-party computation.

and the other half are for P_2's input wires. When all selection bit $\sigma_i = \sigma$ and $j_i \notin J$, receiver P_2 obtains $y_{\sigma,i}$ in every circuit.

Data Availability

The performance test data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was funded by China Scholarship Council (no. 202006220176) and was supported in part by the National Natural Science Foundation of China, under Grant 61632020, Science and Technology Innovation Base Special Project of Provincial Software Engineering Key Laboratory, under Grant 11480004042015, and Development and Construction Funds Project of National Independent Innovation Demonstration Zone in Shandong Peninsula, under Grant S190101010001.

References

[1] C. C. . Yao, "How to generate and exchange secrets," in *Proceedings of the Symposium on Foundations of Computer Science*, IEEE, Toronto, ON, Canada, October 2008.

[2] O. Goldreich, "How to play ANY mental game," in *Proceedings of the nineteenth annual ACM symposium on Theory of computing(STOC)*, Newyork, NY, USA, January 1987.

[3] M. O. Rabin, "How to exchange secrets by oblivious transfer," IACR, Lyon, France, Technical Memo TR-81, 1981.

[4] C. Peikert and B. Waters, "Lossy trapdoor functions and their applications," *SIAM Journal on Computing*, vol. 40, no. 6, p. 279, 2007.

[5] C. Peikert, V. Vaikuntanathan, and B. Waters, "A framework for efficient and composable oblivious transfer," in *Proceedings of the International Cryptology Conference*, pp. 554–571, Santa Barbara, CA, USA, August 2008.

[6] R. Canetti, "Universally composable security: a new paradigm for cryptographic protocols," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science(FOCS)*, Newport Beach, CA, USA, October 2001.

[7] W. Chun-Xiao and W. Ji-Zhong, "A lattice-based oblivious transfer protocol in malicious model," *Energy Procedia*, vol. 13, pp. 1095–1100, 2011.

[8] M. Liu and H. U. Yupu, "Universally composable oblivious transfer from ideal lattice," *Frontiers of Computer ence*, vol. 13, no. 4, pp. 879–906, 2019.

[9] Z. Li, C. Xiang, and C. Wang, "Oblivious transfer via lossy encryption from lattice-based cryptography," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 5973285, 11 pages, 2018.

[10] W. Quach, "UC-secure OT from LWE, revisited," in *Proceedings of the Security and Cryptography for Networks, 12th International Conference, SCN 2020*, Amalfi, Italy, September 14–16, 2020.

[11] J. Katz and V. Vaikuntanathan, "Smooth projective hashing and password-based authenticated key exchange from lattices," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, Melbourne, Australia, December 2009.

[12] F. Kiefer and M. Manulis, "Distributed Smooth projective hashing and its application to two-server password authenticated key exchange," in *Proceedings of the International Conference on Applied Cryptography & Network Security*, June 2014.

[13] S. Halevi and Y. T. Kalai, "Smooth projective hashing and two-message oblivious transfer," *Journal of Cryptology*, vol. 25, no. 1, pp. 158–193, 2012.

[14] R. Cramer and V. Shoup, "Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption," in *Proceedings of the International Conference on the Theory & Applications of Cryptographic Techniques: Advances in Cryptology*, May 2002.

[15] F. Benhamouda, O. Blazy, L. . Ducas, and W. Quach, "Hash proof systems over lattices revisited," *Cryptography – PKC 2018*, Springers, Berlin, Germany, 2018.

[16] D. Micciancio and C. Peikert, "Trapdoors for lattices: simpler, tighter, faster, smaller," in *Proceedings of the 31st Annual international conference on Theory and Applications of Cryptographic Techniques*Berlin, Heidelberg, April 2012.

[17] J. Katz and V. Vaikuntanathan, "Round-optimal password-based authenticated key exchange," *Journal of Cryptology*, vol. 26, no. 4, pp. 714–743, 2013.

[18] J. Zhang and Y. Yu, "Two-round PAKE from approximate SPH and instantiations from lattices," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, Hong Kong, China, December 2017.

[19] X. Wei, H. Jiang, C. Zhao, M. Zhao, and Q. Xu, "Fast cut-and-choose bilateral oblivious transfer for malicious adversaries," in *Proceedings of the 2016 IEEE Trustcom/BigDataSE/I SPA*, August 2016.

[20] Y. Lindell and B. Pinkas, "An efficient protocol for secure two-party computation in the presence of malicious adversaries," *Journal of Cryptology: The Journal of the International Association for Cryptologic Research*, vol. 28, 2015.

[21] M. Naor and B. Pinkas, "Efficient oblivious transfer protocols," in *Proceedings of the Symposium on Discrete Algorithms*, Washington D.C. USA, January 2001.

[22] Y. Lindell, "Efficient fully-simulatable oblivious transfer," *Journal of Cryptology*, vol. 2008, p. 35, 2008.

[23] Y. Lindell and B. Pinkas, "Secure two-party computation via cut-and-choose oblivious transfer," *Journal of Cryptology*, vol. 25, 2012.

[24] V. Kolesnikov and R. Kumaresan, "On cut-and-choose oblivious transfer and its variants," *Lecture Notes in Computer ence*, vol. 2757, pp. 242–266, 2015.

[25] Y. Lindell, "Fast cut-and-choose based protocols for malicious and covert adversaries," in *Proceedings of the 33rd Annual Cryptology Conference*, Santa Barbara, CA, USA, August 2013.

[26] Z. Chuan, J. Han, and W. Xiao-Chao, "Cut-and-Choose bilateral oblivious transfer," *Journal of Software*, vol. 1, no. 2, pp. 384–391, 2017.

[27] Z. Brakerski and N. Döttling, "Two-message statistically sender-private OT from LWE," *Theory of Cryptography*, Springers, Berlin, Germany, 2018.

[28] H. Jiang and Q. Xu, "Advances in key techniques of practical secure multi-party computation," *Journal of Computer Research and Development*, vol. 52, no. 10, pp. 2247–2257, 2015.

[29] D. Micciancio and O. Regev, "Worst-case to average-case reductions based on Gaussian measures," in *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, Rome, Italy, October 2004.

[30] R. Oded, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM*, vol. 56, 2009.

[31] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, May 17-20, 2008.

[32] B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang, "Adaptive oblivious transfer with access control from lattice assumptions," in *Proceedings of the international conference on the theory and application of cryptology and information security*, pp. 533–563, Hong Kong, China, December 2017.

[33] G. Asharov, A. Jain, and D. Wichs, "Multiparty computation with low communication, computation and interaction via threshold FHE," in *Proceedings of the International Conference on Theory & Applications of Cryptographic Techniques*Springer, Berlin, Germany, April 2011.