IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 4, 2020, accepted January 14, 2020, date of publication January 20, 2020, date of current version January 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2967812

Permutable Cut-and-Choose Oblivious Transfer and Its Application

XIAOCHAO WEI, LIN XU, HAO WANG, AND ZHIHUA ZHENG
College of Information Science and Engineering, Shandong Normal University, Jinan 250358, China

Corresponding author: Xiaochao Wei (wxc@sdnu.edu.cn)

This work was supported in part by the China Postdoctoral Science Foundation under Grant 2018M632712, in part by the National Natural Science Foundation of China for Young Scientists under Grant 61802235 and Grant 61602287, in part by the Key Research and Development Plan of Shandong Province under Grant 2018GGX101037, and in part by the Major Innovation Project of Science and Technology in Shandong Province under Grant 2018CXGC0702.

ABSTRACT Oblivious transfer (OT) is a significant primitive with wide use in secure two-party computation, private set intersection private and other cryptographic schemes. In the past ten years, different variants of OT primitive like cut-and-choose OT (CCOT) and outsourced OT (OOT), have been proposed so as to satisfy various emerging models. In this paper, we firstly propose and formalize a new primitive called permutable cut-and-choose OT (PCCOT) which generalizes the original CCOT functionality. Furthermore, we construct an efficient PCCOT protocol in the presence of malicious adversaries using the Decisional Diffie-Hellman (DDH) hard assumption. It is worth mentioning that we apply the PCCOT primitive to the efficient construction of secure wildcard pattern matching (WPM) protocol. The WPM functionality allows a party to determine the locations of its pattern with wildcard characters occurs in a long text of another party while revealing nothing to either party in addition to the length of their own inputs. Our proposed secure WPM protocol via PCCOT is secure against semi-honest adversary with 2 rounds and has identical communication cost as the the state-of-the-art scheme.

INDEX TERMS Secure two-party computation, cut-and-choose oblivious transfer, permutable cut-and-choose oblivious transfer, secure wildcard pattern matching.

I. INTRODUCTION

A. BACKGROUNDS

In secure two-party computation (STPC) [1], [2] setting, two separate parties P_1 and P_2 cooperate to accomplish a concrete function f using their secret inputs. Meanwhile, the two parties obtain nothing rather than relevant outputs. The formal security definition of STPC relies on the *ideal/real simulation paradigm* [3]–[5] where a real protocol execution ''emulates'' the ideal-world setting. Concretely, the ideal world contains a trusted third party which receives input from the parties, computes a target function and finally gives output to the corresponding party. We say the trusted third party can not be corrupted such that the ideal world holds perfect security beyond that the adversary can only changes its input value. Informally speaking, a secure protocol must hold that no real-world adversary can do more harm than ideal-world adversaries. Formally, we compare the joint output of a real protocol executed by the adversary and an honest party, to the corresponding joint output of the ideal-world execution. Another important property involved in STPC is the adversary's power which indicates two different models. The semi-honest model forces the adversary to follow the protocol's exact specification, however the adversary may want to obtain some extra information from its received messages. The malicious model allows the corrupted party to operate in any strategy. In this paper, the above two models both are considered in our protocols.

The associate editor coordinating the review of this manuscript and approving it for publication was Chien-Ming Chen.

1) OBLIVIOUS TRANSFER

Oblivious transfer was proposed by Rabin [6] in 1981. As an important cryptographic primitive, it has been widely used in STPC/SMPC protocols [1], [2], [7]–[10], private set intersection [11]–[14], oblivious pseudorandom function valuation [12] and other scenarios. The original OT functionality involves a sender S and a receiver R, where S has an ordered value-pair (x_0, x_1) and R holds a choice $\sigma \in \{0, 1\}$. The OT functionality enables the receiver to obtain x_σ meanwhile guaranteing that S does not know what R chooses and R

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

gets no information about $x_{1-\sigma}$. The security and efficiency of OT protocols have been studied since the beginning, and numerous works especially the OT extension [15], promote the development in this research field. Some other work focus on extending the functionality of original OT such as cut-and-choose OT (CCOT) [7], [16], cut-and-choose bilateral OT (CCBOT) [17], [18] and outsourced OT (OOT) [20], which can be used in some newly proposed models. Concretely, CCOT was firstly proposed by Lindell and Pinkas [7] in TCC 2011 which is used to improve efficiency of the secure protocol for general two-party functionality using Yao's Garbled Circuit (GC) and the cut-and-choose technique. CCOT is a combination of original OT and cut-and-choose technique where a cut-and-choose bit $j \in \{0, 1\}$ is involved so as to allow the receiver to obtain either both (x_0, x_1) if $j = 0$ or only x_σ if $j = 1$. The original intention of CCOT is to transfer keys associated to the circuits for checking and evaluating together so as to obtain efficiency improvement of the general STPC protocol. Besides, Lindell and Pinkas [7] presented an efficient protocol of CCOT functionality based on the work in [19] using Decisional Diffie-Hellman (DDH) assumption. However, whether their work can apply OT extension proposed in [15] to improve efficiency is unknown. Fortunately, in 2015, Kolesnikov and Kumaresan improved the efficiency of CCOT using OT extension technique which can achieve the result of a large number OT instances but using only less basic OT with some symmetric cryptographic operations like Hash.

Afterwards, the works [17], [18] extended the functionality of CCOT and proposed cut-and-choose bilateral OT (CCBOT) which can be used for constructing secure two-party protocol based on GC with optimal rounds complexity. This new primitive enables that the circuits check and circuits evaluation can be accomplished in an one-off way. Furthermore, the protocol in [18] is proven secure in malicious adversaries model and mainly relies on the DDH hard problem.

Different to the above cut-and-choose scenario, outsourced OT (OOT) was proposed in [20] so as to satisfy the outsourced (or cloud-assisted) model. Formally, the OOT functionality involves three parties called sender, receiver and cloud where the cloud finally obtains the output values related to the receiver's choice. We emphasize that this new extension enables participants to outsource their work to an untrusted cloud while preserving their own privacy. It's worth mentioning that OT extension can also play a significant role when improving the real efficiency of the OOT protocols.

2) SECURE WILDCARD PATTERN MATCHING

As a extension of the exact matching, wildcard pattern matching (WPM) requires that the wildcard character matches *any* possible ordinary characters. For example, when considering the binary strings, the wildcard character $*$ can match both 0 and 1. The WPM functionality involves two parties Alice and Bob with input $t \in \{0, 1\}^n$ and $p \in \{0, 1, *\}^m$ (denoted as a pattern with τ wildcards) respectively. In 2010, Hazy and Toft [21] firstly considered this problem and modified the wildcard characters obliviously and synchronously between Alice and Bob so as to transform the wildcard matching into exact matching. Their protocol relies on distributed ElGamal encryption scheme and is proven secure against malicious adversaries. Assuming the input sizes of Alice and Bob are n and m separately, their work requires $O(n + m)$ communication and $O(nm)$ computation cost. Based on this work, Vergnaud [22] obtained more efficient protocol whose communication and computation cost is $O((n + m)k^2)$ and $O(n\log m)$ respectively using Fast Fourier Transform (FFT), in which $1024 \leq k \leq 2048$ denotes as the security parameter.

In 2012, Baron *et al.* [23] firstly considered non-binary alphabet and constructed wildcard pattern matching protocol using additive homomorphic encryption scheme. Their scheme has the identical complexity as [21] and security in semi-honest adversary model.

In 2014, Yasuda *et al.* [24] considered how to compute many Hamming distances for two encrypted strings using a packing method proposed in [25]. Then, they combined this method with symmetric somewhat homomorphic encryption scheme so as to construct secure wildcard pattern matching protocol. Considering outsourced computation model, Saha and Koshiba [26] made a change of the above packing method and achieved outsourced wildcard pattern matching. Their new packing method improves the efficiency of protocol in k-times, however with information reveal about the pattern itself.

In 2017, Kolesnikov *et al.* [27] gave construction using OT and secure string equality test protocol. The core idea is similar to the method in [28] which represents each bit of $t \in \{0, 1\}^n$ using a pair of values in order. Each pair contains a random value and a well-chosen value, and furthermore all the well-chosen values are related to a secret value chosen by the party Alice. Bob can obtain all the well-chosen values via OT protocols if and only if its pattern p matches the text substrings, which also means that Bob can obtain the secret value. Finally, Alice and Bob execute a secure string equality test protocol using their secret values in hand, and Bob determines whether the match successes or not by receiving 1 or 0 from the secure string equality test protocol. In their secure WPM protocol, the two parties interacts in 4 rounds (2 for offline and 2 for online). Besides, it requires $O(nm)$ communication and $O(k)$ (using OT extension) computation cost, and also n-m+1 instances of secure string equality test protocol.

Recently in 2018, Darivandpour et al. [29] proposed a secure WPM protocol using lightweight cryptography. Their protocol is suitable for arbitrary alphabet and input sizes without using expensive cryptographic operations. Unfortunately, it requires an off-line cloud server.

We emphasize that wildcard pattern matching has numerous application scenarios not only for data locally but also for encrypted data in outsourced model, and new efficient constructions are required to promote its development.

B. CONTRIBUTIONS

In this paper, we firstly propose and formalize a brand new cryptographic primitive denoted as permutable cut-and-choose oblivious transfer (PCCOT) which generalizes the original CCOT functionality. Then, we construct a secure wildcard pattern matching protocol based on our proposed PCCOT primitive and zero-sharing scheme. The efficiency of our secure WPM protocol has large improvement by when running the PCCOT instances in batch. Concretely, our contributions mainly contain the following aspects:

1) **New primitive PCCOT.** Our first contribution is to come up with a new cryptographic primitive denoted as PCCOT. As a general variant of CCOT propose by Lindell and Pinkas in [7], PCCOT enables the receiver to obtain the input (x_0, x_1) of the sender in a random order if cut-and-choose bit equals 0, however in CCOT functionality, the receiver obtains (x_0, x_1) in a certain order (0, 1). We emphasize that the certain order reveals no information when checking circuits in GC-based general STPC protocol, however in some concrete protocols, the order may leak some important information and furthermore leads to the proof of security fails.
   Concretely, we formalize the functionality of PCCOT primitive based on CCOT functionality by involving a permutable bit b for the sender so as to randomize the order (i.e. the order is $(0, 1)$ if $b = 0$ and is $(1, 0)$ if $b = 1$). Then, we construct a secure protocol of PCCOT functionality in malicious adversaries model using DDH hard problem assumption. Our protocol requires only constant round, communication and computation cost for each pair of values.
2) **Application to Secure WPM.** The key for secure pattern matching with wildcard is to transfer the wildcard characters using some methods so as to match the ordinary characters meanwhile preserving the privacy of the locations and numbers of those wildcard characters. We found that the newly proposed PCCOT satisfies miraculously the above requirements such that we construct a secure wildcard pattern matching protocol in semi-honest model based on PCCOT and another primitive called zero-sharing.
   Our secure WPM protocol requires 2 rounds, $O(nm)$ communication cost and computation cost, where n and m denote as the input sizes of the two parties respectively. Furthermore, the proposed protocol can achieve one-side simulation security by using a PCCOT protocol with malicious security, which may be of independent interest.

C. PAPER ORGANIZATIONS

The rest content of this paper is organized as follows. In section 2 we introduce some cryptographic preliminaries and security definition. Then we propose and formalize a new primitive denoted as PCCOT in section 3. In addition, we also construct an efficient protocol for PCCOT based on DDH assumption. In section 4, we apply PCCOT to construct a secure WPM protocol in semi-honest model. Finally, we present the conclusion and future work in section 5.

II. PRELIMINARIES AND DEFINITIONS

A. SECURE WILDCARD PATTERN MATCHING

As an important variant of exact pattern matching, wildcard pattern matching requires that the match process must success for the pattern with wildcard information. This special and magical property has wide use in numerous real applications which retrieve information with some same identities. Secure wildcard pattern matching is a concrete two-party functionality which is described in FIGURE 1. It is worth mentioning that the pattern has some wildcards in some positions which can match both 0 and 1 in the text holding by Alice.

- Wait for a text $t \in \{0,1\}^n$ and an integer m from Alice, a pattern $p \in \{0,1,*\}^m$ with τ wildcards and an integer n from Bob;
- Give an index i to Bob if it satisfies that p matches the m-bit substring of t starting at the position i.

FIGURE 1. The secure WPM functionality $\mathcal{F}_{WPM}$.

The security of WPM functionality requires that Alice should not learn the pattern p and also the positions of these wildcards. Besides, if a match successes Bob has no ability to know the corresponding text characters that match the revelent wildcards of the pattern.

B. CUT-AND-CHOOSE OBLIVIOUS TRANSFER

Lindell and Pinkas [7] firstly introduced and formalized the CCOT functionality in TCC 2011. As a new primitive which extends the standard oblivious transfer (OT) proposed by Rabin [6], it has important advantage in improving the efficiency of the STPC protocol using Yao's Garble Circuit. The original CCOT functionality of [7] is described in FIGURE 2.

- Wait for s pairs of values $\overline{x} = \{(x_0^i, x_1^i)\}_{i=1}^s$ from sender S, $\sigma_1, \cdots, \sigma_s \in \{0,1\}$ and a set of indexes $\mathcal{J} \subset [s]$ of size exactly $s/2$ from receiver R.
- If the set $\mathcal{J}$'s size does not equals $s/2$ then the two parties receive $\perp$ as output. Otherwise,
- Give (x_0^j, x_1^j) to R for each $j \in \mathcal{J}$, and $x_{\sigma_j}^j$ to R for each $j \notin \mathcal{J}$.

FIGURE 2. The cut-and-choose OT functionality $\mathcal{F}_{CCOT}$.

The CCOT functionality requires that the receiver R obtains the output values in different ways, which is different from the original OT. In order to achieve this, a set of indices $\mathcal{J} \subset [s]$ is involved to determine what R receives. Concretely, if $j \in \mathcal{J}$ R obtains both (x_0^j, x_1^j) no matter what R's choice is. However, if $j \notin \mathcal{J}$ R can only obtain $x_{\sigma_j}^j$ according to its choice σ_j. Therefore, the index j indicates the ways R obtains output. We emphasize that the size of set $\mathcal{J}$ may be arbitrary and $s/2$ is just required in the work [7].

For simplicity, we consider only one pair of values (x_0, x_1) for the sender and one choice σ (called choice-bit) for the receiver, besides the receiver also has one index j (called cut-and-choose index bit). The simple form of functionality requires that:

- if $j = 0$, R outputs (x_0, x_1);
- if $j = 1$, R outputs x_σ.

We emphasize that in the following section, our proposed PCCOT functionality is also described in this simple form.

C. THE DDH ASSUMPTION AND RAND FUNCTION

Assuming $\mathbb{G}$ is a finite group of prime order q with generator g and (g, g^a, g^b, g^c) is a quadruple in $\mathbb{G}$ where $a, b, c \in_R Z_q$. The decisional Diffie-Hellman (DDH) hard problem is to decide whether $ab \equiv c \bmod q$ or not. In other word, DDH problem indicates the computationally indistinguishability of the following two distribution ensembles:

- $X_1 = (g, g^a, g^b, g^{ab})$ where $a, b \in_R Z_q$.
- $X_2 = (g, g^a, g^b, g^c)$ where $a, b, c \in_R Z_q$ and $c \neq ab$.

We say that a quadruple which has the form like X_1 is a DH tuple, and otherwise a quadruple like X_2 is a non-DH tuple. A function *RAND* is defined using the above quadruples. Concretely, $RAND(w, x, y, z) = (u, v)$ is denoted as $u = (w)^s \cdot (y)^t$ and $v = (x)^s \cdot (z)^t$, where $s, t \leftarrow \mathbb{Z}_q$ are randomly chosen. It satisfies different properties for DH tuple and non-DH tuple.

- Assuming (w, x, y, z) is a DH tuple where $x = w^a$ and $z = y^a$, $u^a = v$ holds in $(u, v) \leftarrow RAND(w, x, y, z)$.
- Assuming (w, x, y, z) is a non-DH tuple, $(w, x, y, z, RAND(w, x, y, z))$ and $(w, x, y, z, g^\alpha, g^\beta)$ are computationally indistinguishable, where $\alpha, \beta \leftarrow \mathbb{Z}_q$ are random.

D. ZERO-KNOWLEDGE PROOF OF KNOWLEDGE FOR DH TUPLE

The protocol for zero-knowledge proof of knowledge of DH tuple shown in [5] is described as follows.

ZK Proof of Knowledge of Diffie-Hellman Tuple

- **Joint statement:** The values $(\mathbb{G}, q, g_0, g_1, h_0, h_1)$ are elements of a group $\mathbb{G}$ whose order is q with generators g_0, g_1.
- **Auxiliary input for the prover:** A witness w such that $h_0 = (g_0)^w$ and $h_1 = (g_1)^w$.
- **The protocol:**
  - P chooses a random $a \in \{1, \cdots, q\}$, computes $\alpha = (g_0)^a$ and sends α to V.
  - The verifier V chooses random $s, t \in \{1, \cdots, q\}$, computes $c = (g_0)^s \cdot \alpha^t$ and sends c to P (this is a perfectly-hiding commitment to c).
  - P chooses a random $r \in \{1, \cdots, q\}$ and computes $A = (g_0)^r$ and $B = (g_1)^r$. It then sends (A, B) to V.
  - V sends t, c as chosen above to P.
  - P checks that $C = (g_0)^c \cdot \alpha^t$ and aborts if not. Otherwise it sends $z = s \cdot w + r$ to V. In addition, P sends a as chosen in the first step to V.
  - V accepts if and only if $\alpha = (g_0)^a$, $A = (g_0)^z / u^s$ and $B = (g_1)^z / v^s$.

The protocol requires 12 exponentiations in total, in which 8 are of the form $x^a \cdot y^b$. We emphasize that the double exponentiations requires 1.25 rather than 2 the cost of a standard exponentiations, such that the overall exponentiations required are 9. Besides, the prover and verifier exchange 8 group elements during the 5 rounds of communication.

E. COMPUTATIONALLY INDISTINGUISHABILITY

Two distribution ensembles $X = \{X(a, n)\}_{a \in \{0,1\}^*; n \in N}$, $Y = \{Y(a, n)\}_{a \in \{0,1\}^*; n \in N}$ are computationally indistinguishable, denoted by $X \overset{c}{\equiv} Y$, if for every probabilistic polynomial-time (**PPT**) algorithm D there exists a negligible function $\varepsilon(n)$ such that for every $a \in \{0, 1\}^*$ and every $n \in N$,

$$|\mathbf{Pr}[D(X(a, n)) = 1] - \mathbf{Pr}[D(Y(a, n)) = 1]| \leq \varepsilon(n).$$

F. SECURITY MODEL AND DEFINITION

The security of STPC protocols in the presence of semi-honest and malicious adversaries is formalized using the idea/real simulation paradigm [3], [4]. Formal definition sees the following description.

Definition 1: Assuming $f : \{0, 1\}^* \times \{0, 1\}^* \to \{0, 1\}^* \times \{0, 1\}^*$ is a concrete two-party functionality and π is a real-world two-party protocol. Protocol π is secure for computing f with abort in semi-honest and malicious adversaries models, if for every non-uniform polynomial-time adversary real-world $\mathcal{A}$, there must exist a non-uniform polynomial-time adversary ideal-world $\mathcal{S}$, satisfying that for each $i \in \{1, 2\}$,

$$\{IDEAL_{f,\mathcal{S}(z),i}(x, y, z, n)\} \overset{c}{\equiv} \{REAL_{\pi,\mathcal{A}(z),i}(x, y, z, n)\}$$

where $x, y \in \{0, 1\}^*$ are the two parties's input, z is the auxiliary input and $n \in \mathbb{N}$ is security parameter.

III. FUNCTIONALITY OF PCCOT AND ITS CONSTRUCTION

A. FUNCTIONALITY $\mathcal{F}_{PCCOT}$

Observing that in the original CCOT functionality, the receiver R obtains (x_0, x_1) in the order of $(0, 1)$ if $j = 0$. The order seems meaningless in checking the correctness of Garbled Circuits, however may reveal some essential information in some other concrete protocols. In order to

- Wait for $(x_0, x_1) \in \{0,1\}^*$ and a permutable-bit $b \in \{0,1\}$ from sender S, and (σ, j) from receiver R in which $\sigma \in \{0,1\}$ denotes as choice-bit and $j \in \{0,1\}$ denotes as cut-and-choose index-bit.
- Give (x_0, x_1) in random order to R when j equals 0, otherwise b and x_σ to R when j equals 1.

FIGURE 3. The functionality of $\mathcal{F}_{PCCOT}$.

TABLE 1. The relationship between two parties's input.

input of R		input of S		
σ	j	x_0	x_1	b
0	0	$\surd$	$\surd$	$\times$
0	1	$\surd$	$\times$	$\surd$
1	0	$\surd$	$\surd$	$\times$
1	1	$\times$	$\surd$	$\surd$

solve this drawback and generalize CCOT functionality, we propose PCCOT functionality which requires that the receiver R obtains (x_0, x_1) in a random order when $j = 0$. Note that the randomness is achieved using a permutable bit $b \in \{0, 1\}$ chosen by the sender S which is secret to R when $j = 0$. We emphasize that when $j = 1$, the receiver R must also obtain the permutable bit b so as to determine which value is related to x_σ. Our PCCOT is a general variant of the original CCOT functionality, and considers more comprehensively about the different cases. The formal description of $\mathcal{F}_{PCCOT}$ is shown in FIGURE 3:

B. PCCOT PROTOCOL AGAINST MALICIOUS ADVERSARIES

Lindell and Pinkas [7] firstly construct a protocol of CCOT functionality in malicious adversaries model based on the DDH-variant protocol [19]. The basic idea is to require the receiver to generate two quadruples according to its input (σ, j) satisfying that the two quadruples both are DH tuples when $j = 0$ and only one quadruple corresponding to σ is DH tuple when $j = 1$. Then the sender S transfers its input (x_0, x_1) separately using the two quadruples received from R. Considering output, R obtains both (x_0, x_1) in order (0,1) if $j = 0$ and obtains only x_σ if $j = 1$. In order to achieve security against malicious adversaries, zero-knowledge of proof of knowledge (ZKPOK) for DH tuple in [5] is involved for guaranteeing that R constructs the quadruples honestly.

Considering the PCCOT functionality, for the reason that S has an additional input b, we firstly attempt to find out the relationship between (x_0, x_1, b) and (σ, j). Concretely, the following TABLE 1 presents clearly what R obtains corresponding to its input:

As shown above, R obtains two values from (x_0, x_1, b) in each case. Therefore, we attempt to achieve transferring 3 values using two quadruples sent by the receiver R. Assuming that R generates two DH tuples (T_1, T_2) (the case $j = 0$), we can set a new tuple from (T_1, T_2) which satisfying that it is a non-DH tuple. Similarly, if one of (T_1, T_2) is a DH tuple (the case $j = 1$), then a non-DH tuple can also be generated. Without loss of generality, we set a DH tuple in the form as (g, g^a, g^b, g^{ab}) and a non-DH tuple as (g, g^a, g^b, g^{ab+1}). The new tuple is set to be $T_3 = \frac{T_1 \cdot T_2}{g}$ which satisfies our requirements and is used to transfer the value b. Besides, the tuples (T_1, T_2) are also used to transfer (x_0, x_1), however are sent in random order according to the permutable bit b. Unfortunately, this method only protects the privacy of the parties such that the protocol can be used in semi-honest model. If considering malicious adversaries, we also should invoke ZKPOK functionality for DH tuple to guarantee the success of simulation.

Generating T_3 from (T_1, T_2) by the sender S can be seen as a way of implicit authentication. Now, we let R to generate all the three tuples (T_1, T_2, T_3) itself and then proves to S that he acts honestly (i.e. only two are Diffie-Hellman tuples). After analysing the above table, we find out that some potential relationship exists between the tuples (T_1, T_2, T_3), that is

- if $j = 0$, (T_1, T_2) are DH tuples, T_3 is a non-DH tuple and $\frac{T_1 \cdot T_2 \cdot T_3}{g}$ is a DH tuple;
- if $j = 1$, one of (T_1, T_2) is a DH tuple, T_3 is a DH tuple and $\frac{T_1 \cdot T_2 \cdot T_3}{g}$ is a DH tuple.

Regarding this property, the receiver R can just prove to S that the tuple $\frac{T_1 \cdot T_2 \cdot T_3}{g}$ is a DH tuple so as to achieve the security in malicious model. In summary, our protocol for PCCOT functionality against malicious adversaries is shown as follows:

Protocol π_{PCCOT} for PCCOT functionality

- **Inputs:** The sender S holds $(x_0, x_1) \in \{0,1\}^n$ and a permutable bit $b \in \{0,1\}$, the receiver R holds $(\sigma, j) \in \{0,1\}$.
- **Auxiliary input:** Both two parties share a group $\mathbb{G}$ with prime order q and generator g_0, besides a security parameter 1^n.
- **Initial phase:**
  1) The receiver R randomly chooses an element $w \leftarrow \mathbb{Z}_q$ and computes $g_1 = (g_0)^w$.
  2) R then chooses random values $\alpha_1, \alpha_2, \alpha_3 \leftarrow \mathbb{Z}_q$ and computes $(h_1^0, h_1^1, h_2^0, h_2^1, h_3^0, h_3^1)$ in the following ways according to the input (σ, j):
     - if $j = 0$, R computes the following values no matter what σ is

$$\begin{bmatrix} h_1^0 = (g_0)^{\alpha_1}, h_1^1 = (g_1)^{\alpha_1} \\ h_2^0 = (g_0)^{\alpha_2}, h_2^1 = (g_1)^{\alpha_2} \\ h_3^0 = (g_0)^{\alpha_3}, h_3^1 = (g_1)^{\alpha_3+1} \end{bmatrix}$$

– if $j = 1, \sigma = 0$, R computes

$$\begin{bmatrix} h_1^0 = (g_0)^{\alpha_1}, h_1^1 = (g_1)^{\alpha_1} \\ h_2^0 = (g_0)^{\alpha_2}, h_2^1 = (g_1)^{\alpha_2+1} \\ h_3^0 = (g_0)^{\alpha_3}, h_3^1 = (g_1)^{\alpha_3} \end{bmatrix}$$

– if $j = 1, \sigma = 1$, R computes

$$\begin{bmatrix} h_1^0 = (g_0)^{\alpha_1}, h_1^1 = (g_1)^{\alpha_1+1} \\ h_2^0 = (g_0)^{\alpha_2}, h_2^1 = (g_1)^{\alpha_2} \\ h_3^0 = (g_0)^{\alpha_3}, h_3^1 = (g_1)^{\alpha_3} \end{bmatrix}$$

Then the receiver R sends the above values $(g_1, h_1^0, h_1^1, h_2^0, h_2^1, h_3^0, h_3^1)$ to the sender S.

3) R proves to S using ZKPOK of DH tuple that the tuple

$$(g_0, g_1, h_1^0 \cdot h_2^0 \cdot h_3^0, \frac{h_1^1 \cdot h_2^1 \cdot h_3^1}{g_1})$$

is a DH tuple with witness w. If the proof fails, then S outputs $\perp$ and halts the protocol.

- **Transfer phase:**
  1) S chooses a random $p_b \in \{0,1\}^n$ whose last bit is b.
  2) S then computes $(u_1, v_1) = RAND(g_0, g_1, h_1^0, h_1^1)$, $(u_2, v_2) = RAND(g_0, g_1, h_2^0, h_2^1)$ and $(u_3, v_3) = RAND(g_0, g_1, h_3^0, h_3^1)$. Afterwards, S sets $w_1 = v_1 \cdot x_0$, $w_2 = v_2 \cdot x_1$ and $w_3 = v_3 \cdot p_b$.
  3) Finally, S sends the above values to R as follows:
     – if the permutable bit $b = 0$, which means that the order of (x_0, x_1) is not changed, then S sends the values in the order of $(u_1, w_1), (u_2, w_2), (u_3, w_3)$ to R;
     – if the permutable bit $b = 1$, which means that the order of (x_0, x_1) is changed, then S sends the values in the order of $(u_2, w_2), (u_1, w_1), (u_3, w_3)$ to R.

     For simplicity, the above values are denoted as $(c_1, \tilde{c}_1), (c_2, \tilde{c}_2)$ and $(c_3, \tilde{c}_3)$ respectively.
- **Output:**
  1) If $j = 0$, R directly computes $\frac{\tilde{c}_1}{(c_1)^w}$ and $\frac{\tilde{c}_2}{(c_2)^w}$ and obtains (x_0, x_1) in a random order (because b is secret for R).
  2) If $j = 1$, R firstly computes $p_b = \frac{\tilde{c}_3}{(c_3)^w}$ and obtains the last bit of p_b as the permutable bit b, then R operates as follows:
     – if $b = 0$ and $\sigma = 0$, R sets $x_0 = \frac{\tilde{c}_1}{(c_1)^w}$;
     – if $b = 0$ and $\sigma = 1$, R sets $x_1 = \frac{\tilde{c}_2}{(c_2)^w}$;
     – if $b = 1$ and $\sigma = 0$, R sets $x_0 = \frac{\tilde{c}_2}{(c_2)^w}$;
     – if $b = 1$ and $\sigma = 1$, R sets $x_1 = \frac{\tilde{c}_1}{(c_1)^w}$.

C. CORRECTNESS

The correctness of protocol means all the parties can always obtain relevant output values. We analyse the correctness for different cases respectively. For $j = 0$ (no matter what σ is), the tuples (g_0, g_1, h_1^0, h_1^1) and (g_0, g_1, h_1^0, h_1^1) are both generated as DH tuples such that R can just compute (x_0, x_1) in random order. In the case $j = 1$, one of the tuples (g_0, g_1, h_1^0, h_1^1) and (g_0, g_1, h_2^0, h_2^1) is a DH tuple (i.e. (g_0, g_1, h_1^0, h_1^1) is a DH tuple if $\sigma = 0$ and (g_0, g_1, h_2^0, h_2^1) is a DH tuple if $\sigma = 1$), besides (g_0, g_1, h_3^0, h_3^1) is also a DH tuple. Therefore, R can firstly compute b so as to determine whether (x_0, x_1) are permutated or not, and then computes x_σ in the corresponding location. We claim that the correctness of protocol π_{PCCOT} is guaranteed based on the property of *RAND* function that if (w, x, y, z) is a DH tuple in which $x = w^a$ and $z = y^a$, then $u^a = v$ holds for $(u, v) \leftarrow RAND(w, x, y, z)$.

D. SECURITY PROOF

Intuitively speaking, the sender's security mainly relies on the assumption of DDH problem. Note that receiver's input reflects in the quadruples he sends to the sender, however the sender has no ability to distinguish a DH tuple and a non-DH tuple such that receiver's input is secret to the sender. Considering the sender's security, the property of *RAND* function ensures that the receiver can not obtain extra values beyond his choice. Furthermore, the ZKPOK guarantees that the receiver must send correct quadruples about his own input honestly.

The formal proof the security is shown in Theorem 1.

Theorem 1: Assuming that the DDH problem is hard in group $\mathbb{G}$, then protocol π_{PCCOT} securely computes $\mathcal{F}_{PCCOT}$ functionality in the presence of malicious adversaries according to the Definition 1.

Proof: The security of protocol π_{PCCOT} is proven in a hybrid model where ZKPOK is invoked using the ideal functionality $\mathcal{F}_{zkpok}$. The formal proof is separated for either S or R is controlled by the adversary.

1) THE ADVERSARY CONTROLS R

Assuming that $\mathcal{A}$ is a real-world adversary controlling R, an ideal-world simulator $\mathcal{S}_R$ is constructed for $\mathcal{A}$ so as to simulate the ideal process. The simulator invokes $\mathcal{A}$ internally and plays the roles of the honest R and the trusted party computing the $\mathcal{F}_{zkpok}$ functionality. Besides, $\mathcal{S}_R$ interacts externally with the trusted third party of $\mathcal{F}_{PCCOT}$ functionality. $\mathcal{S}_R$ simulates protocol π_{PCCOT} according to the following steps:

1) After receiving the values $(g_1, h_1^0, h_1^1, h_2^0, h_2^1, h_3^0, h_3^1)$ from the adversary $\mathcal{A}$, the simulator $\mathcal{S}_R$ verifies whether the ZKPOK process holds or not.
   - If ZKPOK fails, $\mathcal{S}_R$ sends $\perp$ to the trusted party of $\mathcal{F}_{PCCOT}$ which means that the simulation halts.
   - If ZKPOK holds, $\mathcal{S}_R$ runs the extractor of ZKPOK to extract a witness w. Afterwards $\mathcal{S}_R$ computes $(h_1^0)^w, (h_2^0)^w, (h_3^0)^w$ and compares them respectively with (h_1^1, h_2^1, h_1^3). Then $\mathcal{S}_R$ sets the actual input σ and j of $\mathcal{A}$ according to the comparison results:

a) if $h_3^1 \neq (h_3^0)^w$, $\mathcal{S}_R$ knows that $j = 0$ and then sets σ to be 0 or 1 randomly;
b) if $h_1^3 = (h_3^0)^w$, $\mathcal{S}_R$ knows that $j = 1$ and then sets $\sigma = 0$ when $h_1^i = (h_1^0)^w$ or $\sigma = 1$ when $h_1^2 = (h_2^0)^w$.

2) $\mathcal{S}_R$ sends (σ, j) obtained in step 1 to the trusted party computing $\mathcal{F}_{PCCOT}$ and receives output as follows:
- if $j = 0$, $\mathcal{S}_R$ receives (x_0, x_1) in random order;
- if $j = 1$, $\mathcal{S}_R$ receives x_σ and b.

3) Afterwards, $\mathcal{S}_R$ computes the *RAND* function like an honest sender in the following ways:
- if $j = 0$, $\mathcal{S}_R$ computes (u_1, w_1) and (u_2, w_2) using the values (x_0, x_1) respectively like the honest sender. However, $\mathcal{S}_R$ chooses random elements in $\mathbb{G}$ as the values (u_3, w_3). Then $\mathcal{S}_R$ sends these three pairs of values in the identical order receiving from the trusted party ($\mathcal{S}_R$ receives (x_0, x_1) in a random order, therefore it must send the *RAND* values in the same order). We only consider the order of (u_1, w_1) and (u_2, w_2), and the values (u_3, w_3) are still in the third location.
- if $j = 1$, $\mathcal{S}_R$ chooses randomly $p_b \in \{0, 1\}^n$ with last bit b and sets $lo = b \bigoplus \sigma$ (lo denotes the location of x_σ in the value pair). Then, $\mathcal{S}_R$ computes (u_{1+lo}, w_{1+lo}) and (u_3, w_3) using the values (x_σ, p_b) respectively like an honest sender. However, $\mathcal{S}_R$ sets (u_{2-lo}, w_{2-lo}) to be random elements chosen from the group $\mathbb{G}$.

4) Finally, $\mathcal{S}_R$ sends these three pairs of values (u_{1+lo}, w_{1+lo}), (u_{2-lo}, w_{2-lo}) and (u_3, w_3) to $\mathcal{A}$ in right order according to the subscript indices, and outputs whatever $\mathcal{A}$ outputs.

We wish to prove that the joint output distribution of the ideal simulation executed by $\mathcal{S}_R$ and honest S is identical to that of a real protocol executed by $\mathcal{A}$ and honest S. Formally speaking, the following equation should holds

$$\{IDEAL_{\mathcal{F}_{PCCOT},\mathcal{S}_R(z),R}((x_0, x_1, b), (\sigma, j), n)\} \stackrel{c}{\equiv} \{HYBRID^{ZKPOK}_{\pi_{PCCOT},\mathcal{A}(z),R}((x_0, x_1, b), (\sigma, j), n)\}.$$

Observe that the simulator extracts the witness in the zero-knowledge proof such that $\mathcal{S}_R$ can deduce the actual input of the real-world adversary $\mathcal{A}$. It's obvious that the ideal-world simulation and the real-world execution differ only in the ways some unknown values are generated. Concretely, in the case $j = 0$, $\mathcal{S}_R$ computes (u_1, w_1), (u_2, w_2) using (x_0, x_1) honestly, however sets (u_3, w_3) to be random elements chosen from the group $\mathbb{G}$. Besides, in the case $j = 1$, $\mathcal{S}_R$ computes (u_{1+lo}, w_{1+lo}) and (u_3, w_3) using (x_σ, b) respectively like an honest sender, however sets (u_{2-lo}, w_{2-lo}) to be random elements chosen from the group $\mathbb{G}$. The reason this behavior successes is that the corresponding tuples for these values randomly chosen are non-DH tuples. In addition, the property of *RAND* function implies that the output of the *RAND* with a non-DH tuple yields a uniform distribution from an element chosen randomly from the group $\mathbb{G}$. We conclude that the distribution of the values generated by $\mathcal{S}_R$ using (x_0, x_1) or (x_σ, b) is identical to the corresponding distribution of real protocol. Finally, we claim that sender obtains nothing in functionality $\mathcal{F}_{PCCOT}$, such that it is unnecessary to compare the joint output distribution. In conclusion, the proof when R is corrupted mainly relies on the property of the *RAND* function in non-DH tuple and the probability that the simulator fails in extracting the witness in the zero-knowledge proof when the adversary successfully proves is negligible.

2) THE ADVERSARY CONTROLS R

Assuming that $\mathcal{A}$ is a real-world adversary controlling S, an ideal-world simulator $\mathcal{S}_S$ is constructed for $\mathcal{A}$ so as to simulate the ideal process. The simulator invokes $\mathcal{A}$ internally and plays the roles of the honest S and the trusted party computing the $\mathcal{F}_{zkpok}$ functionality. Besides, $\mathcal{S}_S$ interacts externally with the trusted third party of $\mathcal{F}_{PCCOT}$ functionality. $\mathcal{S}_S$ simulates protocol π_{PCCOT} according to the following steps:

1) $\mathcal{S}_S$ chooses randomly $w, \alpha_1, \alpha_2, \alpha_3 \leftarrow \mathbb{Z}_q$ and computes $g_1 = (g_0)^w$ and

$$\begin{bmatrix} h_1^0 = (g_0)^{\alpha_1}, & h_1^1 = (g_1)^{\alpha_1} \\ h_2^0 = (g_0)^{\alpha_2}, & h_2^1 = (g_1)^{\alpha_2} \\ h_3^0 = (g_0)^{\alpha_3}, & h_3^1 = (g_1)^{\alpha_3} \end{bmatrix}$$

Afterwards, $\mathcal{S}_{SEN}$ sends $(g_1, h_1^0, h_1^1, h_2^0, h_2^1, h_3^0, h_3^1)$ to the adversary $\mathcal{A}$.

2) $\mathcal{S}_S$ emulates the ideal ZKPOK between R and S and sends 1 to $\mathcal{A}$ as if it comes from the $\mathcal{F}_{zkpok}$ functionality. We emphasize that the value 1 means R computes the above values in an honest way, however in fact the simulator $\mathcal{S}_S$ cheats in this step by generating $(h_1^0, h_1^1, h_2^0, h_2^1, h_3^0, h_3^1)$ according to its own strategy.

3) After receiving back $(c_1, \tilde{c}_1)$, $(c_2, \tilde{c}_2)$ and $(c_3, \tilde{c}_3)$ from the adversary $\mathcal{A}$, $\mathcal{S}_S$ computes $\frac{\tilde{c}_i}{(c_i)^w}$ for each $i = 1, \cdots, 3$ and determines the input (x_0, x_1, b) choosen actually by $\mathcal{A}$ in the real protocol.

4) Finally, $\mathcal{S}_S$ sends the values x_0, x_1 and b to the trusted party of the functionality $\mathcal{F}_{PCCOT}$. Then it outputs whatever the real-world adversary $\mathcal{A}$ outputs and halts.

We wish to prove that the joint output distribution of the ideal simulation executed by $\mathcal{S}_S$ and honest R is identical to that of a real protocol executed by $\mathcal{A}$ and honest R. Formally speaking, the following equation should hold

$$\{IDEAL_{\mathcal{F}_{PCCOT},\mathcal{S}_S(z),S}((x_0, x_1, b), (\sigma, j), n)\} \stackrel{c}{\equiv} \{HYBRID^{ZKPOK}_{\pi_{PCCOT},\mathcal{A}(z),S}((x_0, x_1, b), (\sigma, j), n)\}.$$

Two main observations must be considered in the above simulation. First, the ideal-world simulation and the real-world execution differ only in that $\mathcal{S}_S$ generates $(g_1, h_1^0, h_1^1, h_2^0, h_2^1, h_3^0, h_3^1)$ satisfying that all the tuples (g_0, g_1, h_1^0, h_1^1), (g_0, g_1, h_2^0, h_2^1) and (g_0, g_1, h_3^0, h_3^1) are DH tuples. Because of this cheating behavior, $\mathcal{S}_S$ learns all the actual (x_0, x_1, b) values of $\mathcal{A}$. The reason this behavior can not been caught by the adversary $\mathcal{A}$ is that $\mathcal{S}_S$

plays the role of simulator for the ZKPOK and claims to $\mathcal{A}$ that proof is correct. Second, the indistinguishability of the above two distributions is based on the DDH hard problem assumption. If there exists a probabilistic polynomial-time algorithm D can be used to distinguish the above two distributions with a non-negligible probability, then there also exists another probabilistic polynomial-time algorithm for distinguishing a DH tuple from a non-DH tuple with some non-negligible probability. However, the DDH problem is hard for polynomial-time algorithm, such that the above assumption is false. The reduction process to the DDH hard problem is straightforward and we omit for simplicity. We conclude that the ideal simulation and real execution are computationally indistinguishable when sender is controlled by an adversary, as required.

In conclusion, we accomplish formal proof of theorem 1.

E. EXACT EFFICIENCY

We analyse the exact efficiency of the protocol π_{PCCOT} in the following three aspects:

- **Rounds:** The protocol requires total 6 rounds of interactive, where the zero-knowledge proof protocol has 5 rounds and S transfers the ''encrypted'' values in the last additional round.
- **Computation cost:** In the setup phase, the receiver R computes 7 exponentiations for the values $(g_1, h_1^0, h_1^1, h_2^0, h_2^1, h_3^0, h_3^1)$. Besides, the zero-knowledge protocol requires 9 exponentiations such that total exponentiations in setup phase are 16. Besides, S computes 12 exponentiations when computing *RAND* functions in the transfer phase, however the involved double exponentiation $x^a \cdot y^b$ costs only 1.25 the cost of the standard exponentiations such that the total number is 7.5. Finally, the output phase requires the receiver R to compute 2 exponentiations for its own output. In summary, the whole protocol requires 25.5 exponentiations.
- **Communication cost:** In the setup phase, R sends 7 elements $(g_1, h_1^0, h_1^1, h_2^0, h_2^1, h_3^0, h_3^1)$ to the sender. Besides, the ZKPOK protocol involves the exchange of 8 group elements. Besides, the sender S sends 6 elements $(c_1, \tilde{c}_1), (c_2, \tilde{c}_2)$ and $(c_3, \tilde{c}_3)$ to R in the transfer phase. In summary, the protocol exchanges 21 group elements overall.

Considering the simple form of functionality $\mathcal{F}_{CCOT}$, the corresponding protocol in [5] exchanges 17 group elements in which 7 from R to S, 8 involved in ZKPOK and 4 from S to R. Besides, R only computes 5 exponentiations in the first step, S computes 8 exponentiations for *RAND* functions (it requires only 5 for the reason of double exponentiation). In total, the protocol requires $5 + 5 + 9 = 19$ exponentiations. Because of the ZKPOK, the protocol also interacts in 6 rounds. In conclusion, for the reason that our proposed PCCOT primitive generalizes the original CCOT functionality, some extra computation and communication cost is required (4 more group elements exchanged and 6.5 exponentiations computed).

Furthermore, we can remove ZKPOK and easily obtain semi-honest PCCOT protocol with only 2 rounds and better computation and commination cost, therefore we omit it.

IV. SECURE WILDCARD PATTERN MATCHING PROTOCOL VIA PCCOT

A. PROTOCOL DESCRIPTION

In this section, we apply PCCOT primitive to the construction of secure wildcard pattern matching protocol with security against semi-honest adversaries. Two main basic primitives zero-sharing and PCCOT are involved in our construction.

Now we give an overview of our full protocol. For simplicity, we consider a text t with the same length as the pattern p, that is $t \in \{0,1\}^m$ and $p \in \{0,1,*\}^m$ with τ wildcard characters. We firstly show how to represent each bit $t_i \in t$ using zero-sharing scheme. Similar to the technique involved in [27], [28], each bit $t_i \in t$ is represented using a pair of two values s_i and r_i satisfying that $\bigoplus_i s_i = 0$ and the r_is are chosen randomly (both s_i and r_i are of length k which is security parameter). The security parameter k is essential in zero-sharing scheme so as to avoid the XOR collision. In other words, there exists a probability that some values which are not in the chosen set $\bigoplus_i s_i = 0$ also satisfy this relationship. In order to reduce this probability to be negligible, we choose these values with the length of security parameter k (i.e. $k = 128$). Concretely, the pair is in order (s_i, r_i) if t_i=0 and is (r_i, s_i) if t_i=1 (the order must guarantee that s_i is in the location corresponding to the bit t_i).

Afterwards, the two parties execute m PCCOT instances where Alice plays the role as sender with these ordered value pairs and Bob plays the role as receiver with input p. It is worth mentioning that a cut-and-choose index bit j is involved so as to differentiate the wildcard character and the ordinary character. Concretely, for each wildcard character the value j is set to be 0 such that Bob obtains both the values in each corresponding pair. Differently, for each ordinary character we set $j = 1$ and require that Bob can only obtain one value from each pair according to the corresponding bit of the pattern p. In summary, Bob obtains different types of values for the wildcard character and the ordinary character, and this is also the core idea and innovation we use permutable cut-and-choose OT primitive.

Finally, Bob computes XOR of the values received in PCCOT instances and then determines the actual output. If the pattern with wildcards matches successfully with a text substring, the fact is actually that Bob receives s_i for those ordinary character and both (s_i, r_i) in random order for those wildcard characters. There must exist 0 in all the XOR values that Bob computes in binary order using all the s_i and (s_i, r_i). We emphasize that the random order of (s_i, r_i) for wildcard character guarantees that Bob can not know the wildcard character of Alice's text substring. Considering the case that match fails, there must exist at least one ordinary character that Bob does not obtain the value s_i such that no 0 occurs in all the XOR values.

The full and detailed description of our secure WPM protocol is shown as follows.

Secure WPM protocol π_{WPM} via PCCOT

- **Inputs:** Alice inputs a text $t \in \{0,1\}^n$, Bob inputs a pattern $p \in \{0,1,*\}^m$ with τ wildcards.
- **Auxiliary input:** Alice and Bob both have a security parameter 1^k.
- **Input representation:**
  1) Alice chooses randomly $s_{i,j}$ with length k satisfying that $\bigoplus_j s_{i,j} = 0$, where $1 \le i \le n-m+1$ and $1 \le j \le m$.
  2) Alice chooses a random $\widetilde{s}_{i,j}$ (with the same length as $s_{i,j}$) for each $s_{i,j}$ and represents each bit of the m-bit substring according to the actual bit value:
     - if it equals 0, the order is $(s_{i,j}, \widetilde{s}_{i,j})$;
     - if it equals 1, the order is $(\widetilde{s}_{i,j}, s_{i,j})$;

     These above values are denoted as the input of Alice in the permutable cut-and-choose OT protocol. Without loss of generality, each pair is denoted as $(k^0_{i,j}, k^1_{i,j})$.
- **PCCOT instances:**
  1) The two parties run a secure PCCOT instance for each $1 \le i \le n$-m+1 and $1 \le j \le m$, where Alice (exactly sender) inputs $(k^0_{i,j}, k^1_{i,j})$ and a random permutable p_b (Alice can choose randomly in the protocol process), and Bob acts as the receiver with inputs p_j and σ_j (denoted as cut-and-choose bit). Concretely, if p_j is in the wildcard location then $\sigma_j = 0$ and otherwise $\sigma_j = 1$;
  2) Bob obtains the outputs during the PCCOT protocols as follows:
     - for each wildcard bit, Bob obtains both $(k^0_{i,j}, k^1_{i,j})$, however in a random order;
     - for each non-wildcard bit, Bob obtains $k^{p_j}_{i,j}$ where p_j equals 0 or 1.

  All the $m(n$-m+1) PCCOT instances can be executed in batch so as to obtain better efficiency.
- **Output:**
  1) Bob computes XOR of the values received from PCCOT protocols for each $1 \le i \le n$-m+1 as follows:
     - for τ wildcard bits, Bob computes 2^τ XOR values of all the τ pairs $(k^0_{i,j}, k^1_{i,j})$ in binary order;
     - for other m-τ non-wildcard bits, Bob computes the XOR value of all the $m-\tau$ $k^{p_j}_{i,j}$;

     Finally, Bob combines these values and obtains 2^τ XOR values in binary order for each $1 \le i \le n$-m+1.
  2) Bob adds i to his output set if the corresponding 2^τ XOR values related to location i contain 0, which means that the substring starting at location i matches the pattern with wildcards.

B. CORRECTNESS

The correctness of our protocol π_{WPM} mainly relies on the properties of the PCCOT and zero sharing primitives. For simplicity, we consider a text $t = 101011$ with the same length as the pattern $p = 1*10*1$ as shown in Figure 4. It's obvious that wildcards '*' exist in the second and fifth locations of p, and furthermore p matches successfully with t in wildcard manner. After executing PCCOT instances, the pattern owner obtains $s_1, (s_2, r_2), s_3, s_4, (r_5, s_5), s_6$ for each bit of p. We emphasize that in location 1,3,4 and 6, only the legal shares are obtained, however in location 2 and 5, both the legal and illegal shares are obtained in random order (the order in our example is not modified, but in real execution the order is randomly set).

Using these received values, the pattern owner can compute 4 XOR values in TABLE 2. We know that 0 exists in the four XOR values which means that the pattern p with two wildcard bits matches the target text t. Furthermore, for the reason that (s_2, r_2) and (r_5, s_5) are obtained in random order, the line of 0 exists in TABLE 2 is also random. This randomness prevents the pattern owner to know the actual bits of t in wildcards locations.

TABLE 2. All XOR values for p.

all shares for each bit of p						XOR value
s_1	r_2	s_3	s_4	r_5	s_6	random
s_1	r_2	s_3	s_4	s_5	s_6	random
s_1	s_2	s_3	s_4	r_5	s_6	random
s_1	s_2	s_3	s_4	s_5	s_6	0

C. PROOF OF SECURITY

Before giving the formal proof, we provide an intuitive security description of the protocol π_{WPM}. Considering the security of Alice, if the match process successes in one location, Bob can only obtain the value 0 which exists in random location of the four XOR values such that Bob does not know the actual wildcard information of Alice. On the other hand, if the match fails in one location, Bob only obtains four random XOR values which reveal no extra information about Alice's input because of the property of zero-sharing scheme. That is, Bob has no ability to distinguish a correct zero-sharing and a random chosen value. Therefore, the security of Alice is guaranteed. Besides, Bob's security mainly relies on the PCCOT primitive. It's obvious that Bob's input p occurs only in the PCCOT instances, the security of PCCOT guarantees that Alice can not get any information about the pattern p.

The formal proof the security of protocol π_{WPM} is shown as follows.

Theorem 2: Assuming that the PCCOT protocol has security against static semi-honest adversaries, then protocol π_{WPM} is secure for computing the $\mathcal{F}_{WPM}$ functionality in semi-honest model.

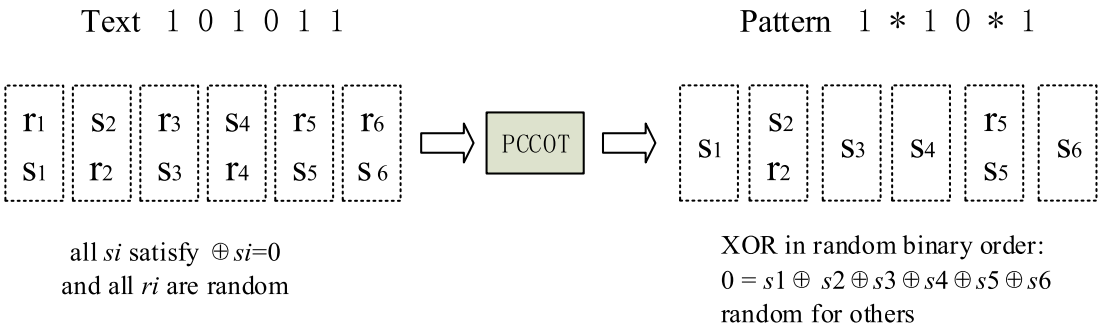


FIGURE 4. The overview of secure WPM protocol with a concrete example.

Proof: The security of protocol π_{WPM} is proven in a hybrid model where PCCOT is invoked using the ideal functionality $\mathcal{F}_{PCCOT}$. The formal proof is separated for either Alice or Bob is controlled by the adversary.

1) ALICE IS CORRUPTED

Let $\mathcal{A}$ be a real-world adversary controlling Alice in an execution of protocol in π_{WPM} where a trusted third party is used to compute the PCCOT functionality $\mathcal{F}_{PCCOT}$. We construct an ideal-world simulator $\mathcal{S}_A$ which invokes $\mathcal{A}$ on its input and plays the role as Bob so as to interact with $\mathcal{A}$ as follows:

1) $\mathcal{S}_A$ invokes $\mathcal{A}$ on a text $t \in \{0,1\}^n$ and an integer m.
2) $\mathcal{S}_A$ simulates the $\mathcal{F}_{PCCOT}$ functionality by receiving corresponding values from the adversary $\mathcal{A}$. Observe that in real protocol, the party Bob computes and sends these values according to its own input. However in the ideal simulation, $\mathcal{S}_A$ has no ability to obtain Bob's actual input such that it just chooses a pattern randomly with the same length and simulates the receiver (Bob) to send it to the $\mathcal{F}_{PCCOT}$ functionality.
3) $\mathcal{S}_A$ sends $\mathcal{A}$'s actual input t and m to the trusted party for computing functionality $\mathcal{F}_{WPM}$, outputs what $\mathcal{A}$ outputs and halts.

For the reason that $\mathcal{F}_{WPM}$ is a single-output functionality where only Bob obtains output, such that we have no need to consider the output of the two parties in the idea-world and real-world executions. We will just explain why the distributions of the simulation executed by $\mathcal{S}_A$ and Bob is computationally indistinguishable with the real protocol between the adversary $\mathcal{A}$ and an honest Bob. Formally, we should prove the following:

$$\{IDEAL_{\mathcal{F}_{WPM},\mathcal{S}_A(1^k),Alice}((t,m),(p,n))\} \overset{c}{\equiv} \{HYBRID^{PCCOT}_{\pi_{WPM},\mathcal{A}(1^k),Alice}((t,m),(p,n))\}.$$

The formal proof of the case Alice is corrupted relies on the security of PCCOT primitive. It's obvious that the ideal simulation and the real protocol execution differ only in the ways the pattern p is chosen. Bob uses its pattern p in real execution, however the simulator $\mathcal{S}_A$ chooses a random value with the same length to complete the simulation process. Considering the security of PCCOT protocol, the sender can not distinguish the choice of the receiver such that the adversary $\mathcal{S}_A$ also can not distinguish what the simulator chooses for the pattern. In other words, the distribution that $\mathcal{S}_A$ chooses a random pattern in ideal-world simulation process is computationally indistinguishable to that generated by an honest Bob with its own pattern. In summary, security holds when Alice is corrupted by an adversary.

2) BOB IS CORRUPTED

Assuming that $\mathcal{A}$ is a real-world adversary controlling Bob in protocol π_{WPM} where ideal functionality $\mathcal{F}_{PCCOT}$ is invoked. We construct an ideal-world simulator $\mathcal{S}_B$ which invokes $\mathcal{A}$ on its input and plays the role as Bob so as to interact with $\mathcal{A}$ as follows:

1) $\mathcal{S}_B$ invokes a pattern $p \in \{0,1,*\}^m$ with τ wildcards and an integer n which are $\mathcal{A}$'s input. Afterwards, $\mathcal{S}$ sends p and n to the trusted party for computing $\mathcal{F}_{WPM}$ functionality. Then it receives match result which contains all the locations of the matched substrings in the text. For simplicity, let the match result contains $i_1, \cdots, i_r$, where $1 \le i_1 \le \cdots \le i_r \le n-m+1$.
2) $\mathcal{S}_B$ chooses randomly $s_{i,j}$ which satisfy that $\bigoplus_j s_{i,j} = 0$ like an honest Alice, where $i = i_1, \cdots, i_r$ and $j = 1, \cdots, m$. Then $\mathcal{S}_B$ simulates the input representation phase by choosing another random $\widetilde{s}_{i,j}$ for each $s_{i,j}$ and generating value pairs according to the received pattern p. Concretely, for the non-wildcard locations, $\mathcal{S}_B$ sets the value pairs as $(s_{i,j}, \widetilde{s}_{i,j})$ for bit 0 or $(\widetilde{s}_{i,j}, s_{i,j})$ for bit 1. Differently, $\mathcal{S}_B$ sets the value pairs in arbitrarily order for the wildcard locations. However, for the other remaining locations, $\mathcal{S}_B$ generates all the value pairs by randomly choosing $s_{i,j}$ and $\widetilde{s}_{i,j}$ and setting them in random order.
3) Finally, $\mathcal{S}_B$ simulates the permutable cut-and-choose oblivious transfer protocol by sending the corresponding outputs to the adversary $\mathcal{A}$, and takes what $\mathcal{A}$ outputs as its own output.

We wish to prove the joint output distribution in the ideal simulation executed by the simulator $\mathcal{S}_B$ and Alice is computationally indistinguishable to that generated by the adversary $\mathcal{A}$ and Alice in real protocol. Formally, we should prove the following equation:

$$\{IDEAL_{\mathcal{F}_{WPM},\mathcal{S}_B(1^k),Bob}((t,m),(p,n))\} \overset{c}{\equiv} \{HYBRID^{PCCOT}_{\pi_{WPM},\mathcal{A}(1^k),Bob}((t,m),(p,n))\}.$$

TABLE 3. Comparison with related secure wildcard pattern matching protocols.

Protocol	Primitives-based	Communication	Computation	Rounds	Security
[27]	OT+PEQT	$O(nm)$	$O(k)$	2 (offline) | 2 (online)	semi-honest
ours	PCCOT	$O(nm)$	$O(nm)$	2	semi-honest

We emphasize that the ideal execution and the real protocol differ only in the ways that $\mathcal{S}_B$ generates the value pairs for those unmatched substrings and the wildcard bits of the matched substrings. Firstly, for the unmatched text substrings, $\mathcal{S}_B$ chooses random $s_{i,j}, \widetilde{s}_{i,j}$ and sets them in random order. This strategy successes for the reason that Bob obtains no 0 in all XOR values during the simulation execution, which is identical to the real protocol where Bob can also not obtain 0 in all XOR values for these unmatched text substrings. Second, for the wildcard bits of all the matched text substrings, the simulator $\mathcal{S}_B$ generates the value pairs $s_{i,j}$ and $\widetilde{s}_{i,j}$ honestly with the only difference that the order of the two values in each pair is set randomly. This makes no difference for the reason that in real protocol, Bob also obtains these values in random order and the "legal" shares $s_{i,j}$ guarantee that there exist corresponding 0s in all XOR values. In summary, we prove the security of protocol π_{WPM} when Bob is controlled by an adversary, as required.

In conclusion, we accomplish formal proof of theorem 2.

D. EFFICIENCY ANALYSIS AND COMPARISON

Our protocol mainly relies on PCCOT primitive which requires only 2 rounds, constant computation and computation cost in semi-honest model. Considering secure WPM protocol with input $t \in \{0, 1\}^n$ and $p \in \{0, 1, *\}^m$, it requires to invoke $m(n\text{-}m\text{+}1)$ 1-out-of-2 PCCOT protocol instances in total. As mentioned above, each instance requires constant computation and computation cost, such that the whole protocol π_{WPM} requires $O(nm)$ communication cost and $O(nm)$ computation cost. Besides, the protocol π_{WPM} requires only 2 rounds for that all PCCOT instances can be executed in batch way.

The comparison result of our proposed protocol π_{WPM} with another related and similar protocol presented in [27] is presented in Table 3. Their protocol is mainly based on OT extension, and additional private equality test (PEQT) is also required. However, our protocol only relies on the new PCCOT primitive. In terms of efficiency, our protocol has the same communication cost $O(nm)$ as the protocol in [27]. For the reason that the OT extension technique can not be directly used for the PCCOT primitive, our protocol has $O(nm)$ computation cost which is $O(k)$ in [27]. The security parameter k is the number the base OT in OT extension protocol which is less than nm. Besides, the protocol in [27] requires 4 rounds (2 for offline and 2 for online), however our work requires only 2 online rounds.

V. CONCLUSION AND FUTURE WORK

In this paper we present a new cryptographic primitive denoted as PCCOT, which is a general variant of the original CCOT functionality. We formalize the PCCOT functionality and construct an efficient protocol in presence of malicious adversaries using DDH hard problem assumption. Then we apply the new PCCOT primitive to construct secure WPM protocol in semi-honest model which reauires only 2 rounds, $O(nm)$ communication cost and computation cost, where n and m are separately the input length of the two parties.

In the future, we will mainly consider the malicious model and construct secure wildcard pattern matching protocol with better efficiency and higher security with malicious security. Furthermore, we will also pay attention to outsourced wildcard and approximate pattern matching so as to realize more and more applications in cloud computing model.

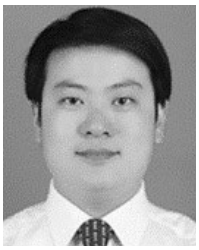
REFERENCES

[1] A. C. Yao, "Protocols for secure computations," in *Proc. 23rd Annu. Symp. Found. Comput. Sci. (FOCS)*, Nov. 1982, pp. 160–164.
[2] A. C. Yao, "How to generate and exchange secrets," in *Proc. 27th Annu. Symp. Found. Comput. Sci. (FOCS)*, Los Alamitos, CA, USA, Oct. 1986, pp. 162–167.
[3] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game—A completeness theorem for protocols with honest majority," in *Proc. 19th STOC*, Oct. 1987, pp. 218–229.
[4] O. Goldreich, *Foundations of Cryptography: Basic Applications*, vol. 2. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[5] C. Hazay and Y. Lindell, *Efficient Secure Two-Party Protocols: Techniques and Constructions*. Berlin, Germany: Springer, 2010.
[6] M. O. Rabin, "How to exchange secrets by oblivious transfer," Harvard Univ., Cambridge, MA, USA, Tech. Rep. TR-81, 1981.
[7] Y. Lindell and B. Pinkas, "Secure two-party computation via cut-and-choose oblivious transfer," in *Theory of Cryptography* (Lecture Notes in Computer Science), vol. 6597. Cham, Switzerland: Springer, 2011, pp. 329–346.
[8] Y. Lindell, "Fast cut-and-choose based protocols for malicious and covert adversaries," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 8043. Cham, Switzerland: Springer, 2013, pp. 1–17.
[9] Y. Huang, J. Katz, and D. Evans, "Efficient secure two-party computation using symmetric cut-and-choose," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 8043. Cham, Switzerland: Springer, 2013, pp. 18–35.
[10] P. Mohassel and B. Riva, "Garbled circuits checking garbled circuits: More efficient and secure two-party computation," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 8043. Cham, Switzerland: Springer, 2013, pp. 36–53.
[11] V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu, "Efficient batched oblivious PRF with applications to private set intersection," in *Proc. ACM CCS*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds., Oct. 2016, pp. 818–829.
[12] V. Kolesnikov, N. Matania, B. Pinkas, M. Rosulek, and N. Trieu, "Practical multi-party private set intersection from symmetric-key techniques," in *Proc. ACM CCS*, B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds., Oct./Nov. 2017, pp. 1257–1272.

[13] H. Chen, K. Laine, and P. Rindal, "Fast private set intersection from homomorphic encryption," in *Proc. ACM SIGSAC Conf. CCS*, 2017, pp. 1243–1255.
[14] B. Pinkas, T. Schneider, and M. Zohner:, "Scalable private set intersection based on OT extension," *ACM Trans. Privacy Secur.*, vol. 21, no. 2, pp. 7:1–7:35, 2018.
[15] Y. Ishai, J. Kilian, and K. Nissim, "Extending oblivious transfers efficiently," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2003, pp. 145–161.
[16] V. Kolesnikov and R. Kumaresan, "On cut-and-choose oblivious transfer and its variants," in *Advances in Cryptology—ASIACRYPT*. Berlin, Germany: Springer, 2015, pp. 386–412.
[17] C. Zhao, H. Jiang, X. Wei, Q. Xu, and M. Zhao, "Cut-and-choose bilateral oblivious transfer and its application," in *Proc. 14th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Aug. 2015, pp. 384–391.
[18] X. Wei, H. Jiang, and C. Zhao, "Fast cut-and-choose bilateral oblivious transfer for malicious adversaries," in *Proc. 15th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Aug. 2016, pp. 418–425.
[19] C. Peikert, V. Vaikuntanathan, and B. Waters, "A framework for efficient and composable oblivious transfer," in *Proc. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2008, pp. 554–571.
[20] H. Carter, B. Mood, P. Traynor, and K. Butler, "Secure outsourced garbled circuit evaluation for mobile devices," in *Proc. USENIX Secur. Symp.*, 2013 pp. 137–180.
[21] C. Hazay and T. Toft, "Computationally secure pattern matching in the presence of malicious adversaries," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur. (ASIACRYPT)*. Berlin, Germany: Springer, 2010, pp. 195–212.
[22] D. Vergnaud, "Efficient and secure generalized pattern matching via fast Fourier transform," in *Proc. Int. Conf. Cryptol. Africa*. Berlin, Germany: Springer, 2011, pp. 41–58.
[23] J. Baron, K. El Defrawy, and K. Minkovich, "5pm: Secure pattern matching," in *Proc. Int. Conf. Secur. Cryptogr. Netw.* Berlin, Germany: Springer, 2012, pp. 222–240.
[24] M. Yasuda, T. Shimoyama, and J. Kogure, "Privacy-preserving wildcards pattern matching using symmetric somewhat homomorphic encryption," in *Information Security and Privacy* (Lecture Notes in Computer Science), vol. 8544. Cham, Switzerland: Springer, 2014, pp. 338–353.
[25] M. Yasuda, T. Shimoyama, and J. Kogure, "Secure pattern matching using somewhat homomorphic encryption," in *Proc. ACM Workshop Cloud Comput. Secur. Workshop*. New York, NY, USA: ACM, 2013, pp. 65–76.
[26] T. K. Saha and T. Koshiba, *An Enhancement of Privacy-Preserving Wildcards Pattern Matching*. Cham, Switzerland: Springer, 2017, pp. 145–160.
[27] V. Kolesnikov, M. Rosulek, and N. Trieu. (Jul. 26, 2017). *SWiM: Secure Wildcard Pattern Matching From OT Extension*. [Online]. Available: https://eprint.iacr.org/2017/1150.pdf
[28] X. Wei, M. Zhao, and Q. Xu, "Efficient and secure outsourced approximate pattern matching protocol," *Soft Comput.*, vol. 22, no. 4, pp. 1175–1187, Feb. 2018.
[29] J. Darivandpour and M. J. Atallah, "Efficient and secure pattern matching with wildcards using lightweight cryptography," *Comput. Secur.*, vol. 77, pp. 666–674, Aug. 2018, doi: 10.1016/j.cose.2018.01.004.

XIAOCHAO WEI was born in 1990. He received the Ph.D. degree in computer science and technology from Shandong University. He is currently a Lecturer with Shandong Normal University. His main interests include secure multiparty computation, privacy preserving, and searchable encryption.

LIN XU was born in 1997. She is currently pursuing the master's degree with Shandong Normal University. Her main interests include secure multiparty computation protocol and privacy preserving.



HAO WANG was born in 1984. He received the Ph.D. degree in computer science and technology from Shandong University. He is currently an Associate Professor with Shandong Normal University. His main interests include public key cryptography, secure multiparty computation, and blockchain.

ZHIHUA ZHENG was born in 1962. She is currently an Associate Professor with Shandong Normal University. Her main interests include information security, cryptographic protocols, and blockchain.