Practical Single-Round Secure Wildcard Pattern Matching

Jun Xu[1], Shengnan Zhao[2,3], Chuan Zhao[2(✉)], Zhenxiang Chen[1], Zhe Liu[4,5], and Liming Fang[4]

[1] University of Jinan, Jinan, China
[2] Quan Cheng Laboratory, Jinan, China
ise_zhaoc@ujn.edu.cn
[3] Shandong University, Jinan, China
[4] Nanjing University of Aeronautics and Astronaut, Nanjing, China
[5] Zhejiang Lab, Hangzhou, China

Abstract. Secure pattern matching allows a client who holds a substring (pattern) to find all the substring's locations appearing in the long string (text) stored in a server. Meanwhile, the server should not learn any information about the pattern or the matching results. *Wildcard pattern matching* (WPM) problem, a specific variant with more realistic significance, defines that the pattern contains wildcards that can match any character in the text.

Previous studies introduce various approaches for the WPM problem but requires at least a two-round protocol or computation cost linear to input length. Oriented to applications in the client-server mode, however, existing solutions are not practical and efficient enough. Therefore we focus on the round and computation complexity of the WPM. In this paper, under the semi-honest model, we propose a single-round secure WPM protocol based on oblivious transfer (OT) and secret sharing schemes. The insight of our proposed protocol is the reduction from the WPM to the process of secret sharing and reconstruction in a novel way. We provide a customized OT construction and apply the OT extension technique to the protocol, where the client and the server need merely a constant number of public key operations in a round of communication. In addition, we prove the security of the protocol in the ideal/real simulation paradigm and evaluate the performance. Compared to existing secure WPM protocols, both theoretical and experimental results show that our protocol is more practical.

Keywords: wildcard pattern matching · oblivious transfer · secret sharing

1 Introduction

Secure multi-party computation (MPC) is an important subfield of cryptography and is first introduced by Andrew Yao [1] in the early 1980s. The goal of MPC

J. Xu and S. Zhao—Contributed equally to this work.

Published by Springer Nature Switzerland AG 2024
N. Meyer and A. Grocholewska-Czuryło (Eds.): SEC 2023, IFIP AICT 679, pp. 87–101, 2024.
https://doi.org/10.1007/978-3-031-56326-3_7

is to enable a set of independent mutually untrusted parties to jointly compute a function f on their private inputs, during which any additional information except for the output of that function cannot be revealed. Traditionally, two security models are mainly taken into consideration in MPC, *i.e., semi-honset* model and *malicious* model. In the semi-honest adversarial model, the adversary follows the protocol instruction but tries to learn anything about the other party's input. In contrast, the adversary in the malicious model can follow any arbitrary polynomial-time strategy to deviate from the protocol.

Secure pattern matching (PM) problem is becoming a hot topic in the research field of MPC [2]. In this problem, the goal is to find all the locations of the pattern in a text for the party holding a pattern p, while the other learns nothing about the pattern. The PM problem can be formally defined as follows: given a finite alphabet Σ, a server holds a text $T \in \Sigma^n$ and a client holds a pattern $p \in \Sigma^m$ ($m<n$). The client wants to learn where its pattern is a substring of the server's text. Meanwhile, the server cannot learn any information about the pattern or the matching results. Considering a hospital holding patient genomic data, a researcher with a specific DNA sequence wishes to know the frequency and positions of the gene occurrences in the database and analyze the structure and properties of this sequence. However, the researcher does not intend to reveal its DNA sequence to the hospital. The hospital needs to prevent miscellaneous researchers from pirating the records of its genomic database on the other hand.

Wildcard pattern matching (WPM) problem, a specific variant with more realistic significance, defines that the pattern contains wildcards that can match any character of the alphabet in the text. We mainly focus on the WPM problem in this paper. In general, the wildcard character is denoted by $\star$, which could be any character of the alphabet. In addition, the server is not allowed to learn the locations of wildcard characters in the pattern, either. Same to the PM problem without wildcards, the client can only obtain information where the occurrence of the pattern in the server's text. Pattern matching is widely applied in text retrieval, computational biology, DNA analysis [3], intrusion detection systems [4], and other fields. Prior studies [5–8] introduce various approaches for the WPM problem but requires at least a two-round protocol or expensive computation cost (linear to input length).

Our Contributions. We design an efficient protocol that addresses the WPM problem in the presence of semi-honest adversaries. The proposed protocol is extremely competitive for lightweight devices in many scenarios such as Information Processing Systems. Our contributions can be summed up as follows:

- We provide a novel combination between the secret share scheme and a customized oblivious transfer protocol, which would be building blocks for the pattern matching problem.
- We propose an efficient single-round protocol with semi-honest security for wildcard pattern matching, which requires $O(\kappa)$ exponentiation computation and $O(mn)$ communication, where κ is the security parameter and independent of the input length m and n.

- We evaluate the performance and apply the precomputation technique and proxy OT to our contribution. We prove the security of the protocol in the ideal/real simulation paradigm.

2 Related Works

There are three main approaches to constructing secure pattern matching protocols: oblivious automaton evaluation, homomorphic encryption (HE), and Yao's garbled circuit. To our knowledge, the protocols based on oblivious automaton evaluation are often used to solve the approximate/exact pattern matching problem [9–11]. Yao's garbled circuits is a generic approach for secure computation, which can be used to evaluate arbitrary functions, given a description of the function as a fixed-size circuit. In 2010, Katz and Malka [12] showed how to modify Yao's garbled circuits to obtain a secure pattern matching protocol where the size of the circuit is linear in the number of matched locations. Later, Kolesnikov *et al.* [7] believed that the protocol [12] can be extended to solve wildcard pattern matching, while there may be a requirement that it should provide a priori bound on the number of matches for the circuit construction. Secure wildcard pattern matching protocols based on homomorphic encryption schemes have been extensively studied in the past decade.

The first work of secure wildcard pattern matching was considered by Hazay and Toft [5] in 2010. In their scheme, the client is required to encrypt the wildcard locations using an additively homomorphic variation of ElGamal encryption and then supply the ciphertext to the server. In the meanwhile, the substrings of the server's text must be modified to match the pattern at those positions. Baron *et al.* [13] suggested an efficient pattern matching protocol entitled 5PM. The core idea of their work is to reduce the problem of pattern matching with single-character wildcards to a sequence of linear operations, which can be efficiently computed in the malicious model using additively homomorphic encryption schemes. Therefore, they employed homomorphic encryption in an insecure pattern matching algorithm to support basic linear operations. Yasuda *et al.* [14] adopted a packing method and somewhat homomorphic encryption technique to address both approximate and wildcard pattern matching for non-binary inputs, where the encryption scheme supports a limited number of polynomial additions and multiplications on encrypted data. Their proposed packing method is applied to compute multiple Hammming distance values between the pattern and text in encrypted form.

Recently, Zarezadeh et al. [6] firstly resolved the parameterized pattern matching problem in the semi-honest and malicious setting where there exists a renaming bijection on the alphabet such that a pattern can be transformed into a substring of the text. Their proposed protocol supports wildcard and approximate pattern matching. Subsequently, they extended their construction to the multi-pattern matching scenario [15] that the pattern owner can find the matching locations in multiple texts and presented an efficient solution for parameterized matching of multiple patterns in the semi-honest adversary model.

In addition to the above methods, several works based on oblivious transfer for wildcard pattern matching were studied in [7,8,16]. The scheme of Kolesnikov *et al.* [7] called SWiM is a simple and fast protocol for wildcard pattern matching in a semi-honest setting, which converts the problem of wildcard pattern matching into the problem of secure equality test of strings. On the basis of this idea, Qin *et al.* [16] also presented a pattern matching protocol by combining oblivious transfer with secret sharing. However, these two works require an additional secure string equality test protocol.

3 Preliminaries

Throughout the paper, we use the following notation: The length of the text T is n, while the length of the pattern p is m. The notation $\boldsymbol{t}_i$ denotes the m-bit substring of the text T from the i-th location. The wildcard is denoted by $\star$. We use $[m]$ to denote a set $\{1, \cdots, m\}$. We denote vectors in bold, and matrices in capitals. For a vector $\boldsymbol{x}$, we let $\boldsymbol{x}[k]$ denote the k-element of vector $\boldsymbol{x}$, $x_{i,j}$ denote the j-th share in the i-th secret sharing. For a matrix A, we let $\boldsymbol{a}_i$ denote the i-th row of A, $\boldsymbol{a}^j$ denote the j-th column of A. We use the notation $\vec{\boldsymbol{k}}$ to denote a tuple which contains three vectors.

3.1 Oblivious Transfer

Oblivious Transfer [17] is an essential cryptographic primitive that is used as a fundamental building block for MPC protocols. The standard definition of 1-out-of-2 OT involves two participants, a sender (denoted by $\mathcal{S}$) holding two inputs (m_0, m_1) and a receiver (denoted by $\mathcal{R}$) holding a choice bit $b \in \{0, 1\}$. After the transfer is completed, $\mathcal{R}$ learns m_b without learning anything about the other input m_{1-b}, while $\mathcal{S}$ has no output and learns nothing about b. The efficient 1-out-of-2 OT extension technique (also known as IKNP OT) is introduced by Ishai *et al.* [18], which can achieve an arbitrarily large number of OTs by executing a small (relying on security parameter) number of OT instances, and a number of symmetric key primitives. In 2013, Kolesnikov and Kumaresan [19] presented an optimization and generalization of IKNP OT extension protocol, which offers the sublinear communication/computation cost in the security parameter.

In this paper, our building block of independent interest is 1-out-of-3 OT, which is used to design a secure wildcard pattern matching protocol.

3.2 Secret Sharing

Secret sharing is a fundamental primitive, that is at the core of many MPC protocols. The idea of secret sharing was introduced by Shamir [20] and Blakey [21], and they constructed specific threshold secret sharing schemes based on Lagrange's interpolation theorem and projective geometry theory, respectively. Informally speaking, in a (t, n)-secret sharing scheme ($t \leq n$), the secrets s

can be split into n shares, which would be distributed among several parties, such that any $t-1$ of the shares cannot leak anything about s, while any t shares allow complete reconstruction of the secret s. In secret-sharing-based MPC protocol, the target is then to obtain a secret-shared representation of the inputs to the computation, such that any possible set of adversarial parties reveals no information about the underlying secret. A (t, n)-secret sharing scheme needs to satisfy these two properties: *correctness* (meaning that any $k \geq t$ shares can completely determine the secret) and *privacy* (meaning that any set of shares of size less than t does not leak anything about the secret).

In our discussion, we will use (n, n)-secret sharing schemes, where all n shares are required and sufficient to reconstruct the secret.

Functionality $\mathcal{F}_{\text{WPM}}$

Parameters:

- A text of length n, a pattern of length m, and two parties: sender $\mathcal{S}$ and receiver $\mathcal{R}$.

Input:

- Wait for $\mathcal{S}$ to input a text $T \in \{0,1\}^n$.
- Wait for $\mathcal{R}$ to input a pattern $p \in \{0,1,\star\}^m$.

Output:

- $\mathcal{S}$ has no output.
- $\mathcal{R}$ outputs i if p matches the substring $T[i \cdots i + |p| - 1]$.

Fig. 1. Wildcard Pattern Matching Functionality

4 Secure Wildcard Pattern Matching Scheme

In this work, we present a secure two-party protocol for wildcard pattern matching based on two cryptographic tools: oblivious transfer and secret sharing. The rationale behind our secure wildcard pattern matching scheme is described in the following. Then, we explain how to construct a secure scheme for semi-honest parties.

4.1 Overview of Techniques

Before discussing our contributions and technical approach, we review the privacy-preserving wildcard pattern matching protocol in [16]. They transformed the secure wildcard pattern matching problem into reconstruction of a shared secret and presented a secure two-party wildcard pattern matching protocol

based on oblivious transfer and secret sharing. As usual, the sender $\mathcal{S}$ has a text $T \in \{0,1\}^*$ and the receiver $\mathcal{R}$ holds a pattern $p \in \{0,1,\star\}^*$. The intuition idea behind their protocol is that, (1) the pattern p is represented by the shares of secret s; (2) the two parties with reverse roles invoke the standard 1-out-of-2 OT protocol, where $\mathcal{R}$ as the sender inputs these shares and some random shares and $\mathcal{S}$ as the receiver inputs the text T. (3) $\mathcal{S}$ receives shares corresponding to its own input T from OT protocol and reconstructs the secret s'. If $s' = s$, this means $\mathcal{S}$ receives all the valid shares. Therefore, the pattern p matches the text T. However, their proposed protocol is inapplicable to many application scenarios due to the limited computing power of the receiver (for example, if it is a mobile device). Furthermore, the sender and receiver in their protocol are required to perform a string equality test protocol relying on OT, which causes additional overhead. We modify their protocol and use 1-out-of-3 OT protocol to achieve wildcard pattern matching, where the two parties always maintain a single role during the whole execution of our protocol. In addition, we transfer most of the computational cost to the sender $\mathcal{S}$ and achieve better efficiency by dispensing with the string equality test technique.

Our method can be depicted as follows. Suppose the sender $\mathcal{S}$ holds a text $T \in \{0,1\}^*$ and the receiver $\mathcal{R}$ holds a pattern $p \in \{0,1,\star\}^*$. To achieve secure pattern matching, the key is how to judge the i-th bit of p matches the i-th bit of T securely. As a very simple warm up, consider the case that $|T| = |p| = 1$. $\mathcal{R}$ wants to know whether the pattern p matches the text T or not and $\mathcal{S}$ can not obtain any information about p. Initially, $\mathcal{S}$ will first choose a public string $\boldsymbol{r}$ of length κ as a substitute of T and another string $\boldsymbol{s}$ of the same length as $\boldsymbol{r}$ randomly. According to the value of T, $\mathcal{S}$ sets a tuple $\vec{\boldsymbol{k}}$ as follows: if $T = 0$, $\vec{\boldsymbol{k}} = (\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{r})$; Otherwise, $\vec{\boldsymbol{k}} = (\boldsymbol{s}, \boldsymbol{r}, \boldsymbol{r})$. Then $\mathcal{S}$ and $\mathcal{R}$ jointly execute a 1-out-of-3 OT protocol, where $\mathcal{S}$ gives input $\vec{\boldsymbol{k}}$ and $\mathcal{R}$ gives input $c \in \{0,1,2\}$. After that, $\mathcal{R}$ determines if p and T are matched based on whether its output from OT protocol is $\boldsymbol{r}$. Note if $p = \star$, then $\mathcal{R}$ sets $c = 2$ and always can obtain $\boldsymbol{r}$. In this process, the only new information that $\mathcal{R}$ obtains is output $\boldsymbol{r}$ or $\boldsymbol{s}$, which leaks no information about the text T of $\mathcal{S}$.

Next, we extend this approach to the case $|T| = |p| = m$ by combining secret sharing. Specifically, $\mathcal{S}$ represents its text T on shares of $\boldsymbol{r}$ using a secret sharing scheme, *i.e.,* each bit of T is denoted by a secret share r_j. In the meantime, a random share s_j is selected for every secret share of $\boldsymbol{r}$. After doing so, $\mathcal{S}$ will set a tuple $\vec{\boldsymbol{k}}_j$ for each bit of T using s_j and r_j in the same way as above. Then $\mathcal{S}$ and $\mathcal{R}$ invoke 1-out-of-3 OT protocol m times, where $\mathcal{R}$ takes as input $\boldsymbol{c}[j]$ for every time. T matches p if and only if $\mathcal{R}$ obtains all shares of $\boldsymbol{r}$ from OT protocol and reconstructs $\boldsymbol{r}$. An example is given in Fig. 2. During the whole execution, $\mathcal{R}$ either receive all the valid shares of $\boldsymbol{r}$ or at least a random share. In the latter case, $\mathcal{R}$ can not reconstruct the publicly shared string $\boldsymbol{r}$. The security of the protocol is obvious in the semi-honest setting: $\mathcal{R}$ can not obtain any information about the text T of $\mathcal{S}$ but know whether the pattern p matches T.

The idea of the general case of wildcard pattern matching with $|T| > |p|$ is natural: simply perform the above method on each substring $T[i \cdots i + |p| - 1]$

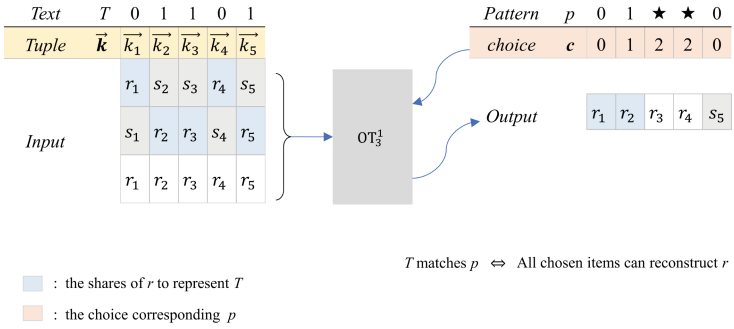

Fig. 2. Illustration of the core idea of our protocol

and p. Besides, note that in every matching of substring $T[i \cdots i + |p| - 1]$ and pattern p, $\mathcal{R}$'s OT choice is always the same selection integer vector $\boldsymbol{c}$. Hence instead of $|p|(|T| - |p| + 1)$ instances of string-OT where the string is κ bits long, we can use $|p|$ instances of string-OT, with string of length $\kappa(|T| - |p| + 1)$ to reduce computation cost.

4.2 Secure Wildcard Pattern Matching Protocol

In this section, we present a new secure wildcard pattern matching protocol Π_{WPM} based on 1-out-of-3 OT first and then give an analysis of correctness and a formal proof of security. The wildcard pattern matching functionality, denoted by $\mathcal{F}_{WPM}$, is formally defined in Fig. 1. Recall that the discussion in Sect. 4.1, T matches p if and only if $\mathcal{R}$ obtain all shares of $\boldsymbol{r}$ from OT protocol. The detailed protocol is presented in Fig. 3 and is proven secure in the presence of semi-honest adversaries.

Correctness. Our goal here is to prove that in an honest execution of the protocol the output of $\mathcal{R}$ is indeed the locations in the text T where the pattern p appears. Note that in the OT phase $\mathcal{S}$ takes as input shares of $\boldsymbol{r}$ and corresponding random shares. The pattern p matches the substring $\boldsymbol{t}_i$ if only if $\mathcal{R}$ can get all the valid shares of $\boldsymbol{r}$ and reconstruct $\boldsymbol{r}$. For non-wildcard bits in pattern p, the relevant valid shares would be received by $\mathcal{R}$ when these bits are equal to the values at the corresponding locations. In the case of wildcard bits in p, $\mathcal{R}$ can always obtain the corresponding valid shares from OT protocol. Therefore, $\mathcal{R}$ can reconstruct the publicly shared string $\boldsymbol{r}$ based on these valid shares and obtain the matching location. And if the match between $\boldsymbol{t}_i$ and p is unsuccessful, there is at least one random share will be outputted to $\mathcal{R}$. Under the circumstance, Bob cannot reconstruct the string $\boldsymbol{r}$ because of the property of the secret sharing scheme.

Security. We are now ready to prove the security of our protocol Π_{WPM} in the presence of semi-honest adversaries.

Wildcard pattern matching protocol Π_{WPM}

Parameters:

- Two parties: *sender* $\mathcal{S}$ and *receiver* $\mathcal{R}$.
- Text length n, pattern length m. Define $n' = n - m + 1$.
- Both parties share a public random string $\boldsymbol{r} \leftarrow \{0,1\}^{\kappa}$ as auxiliary input.
- Ideal functionality OT-functionality $\mathcal{F}_{\mathrm{OT}_3^1}$.

Protocol:

1. [**secret sharing**] For each $i \in [n']$, $\mathcal{S}$ invokes the secret sharing scheme to share the public random string $\boldsymbol{r}$ into $r_{i,j}$, where $j = 1, \cdots, m$ and $\boldsymbol{r} = \bigoplus_{j=1}^{m} r_{i,j}$.
2. $\mathcal{S}$ chooses values $s_{i,j} \leftarrow \{0,1\}^{\kappa}$ at random for $i \in [n']$ and $j \in [m]$.
3. Let $\boldsymbol{t}_i$ denote the m-bit substring of $\mathcal{S}$'s text T for $i = 1, \cdots, n'$. Then $\mathcal{S}$ generates the values tuple for each $j \in [m]$ as follows:
   (1) if $\boldsymbol{t}_i[j] = 0$, the values tuple is in the order of $(r_{i,j}, s_{i,j}, r_{i,j})$;
   (2) if $\boldsymbol{t}_i[j] = 1$, the values tuple is in the order of $(s_{i,j}, r_{i,j}, r_{i,j})$;
   Without loss of generality, let $\vec{\boldsymbol{k}}_{i,j} = (k_{i,j}^0, k_{i,j}^1, k_{i,j}^2)$ denote each tuple. Then $\mathcal{S}$ forms $n' \times m$ matrix U blow using these tuples.

$$\begin{pmatrix} (k_{1,1}^0, k_{1,1}^1, k_{1,1}^2) & (k_{1,2}^0, k_{1,2}^1, k_{1,2}^2) & \cdots & (k_{1,m}^0, k_{1,m}^1, k_{1,m}^2) \\ (k_{2,1}^0, k_{2,1}^1, k_{2,1}^2) & (k_{2,2}^0, k_{2,2}^1, k_{2,2}^2) & \cdots & (k_{2,m}^0, k_{2,m}^1, k_{2,m}^2) \\ \vdots & \vdots & & \vdots \\ (k_{n',1}^0, k_{n',1}^1, k_{n',1}^2) & (k_{n',2}^0, k_{n',2}^1, k_{n',2}^2) & \cdots & (k_{n',m}^0, k_{n',m}^1, k_{n',m}^2) \end{pmatrix}$$

4. [**OT**] For each $j \in [m]$, $\mathcal{S}$ and $\mathcal{R}$ invoke 1-out-of 3 OT-functionality $\mathcal{F}_{\mathrm{OT}_3^1}$
   (1) $\mathcal{R}$ acts as *receiver* with input a selection integer $\boldsymbol{c}[j]$ corresponding to the j-th bit of p.
   (2) $\mathcal{S}$ acts as *sender* with input $\mathbf{u}^j$ as the j-th column of U
   (3) $\mathcal{R}$ receives output $\mathbf{v}^j = (k_{1,j}^{\boldsymbol{c}[j]}, k_{2,j}^{\boldsymbol{c}[j]}, \cdots, k_{n',j}^{\boldsymbol{c}[j]})$
5. [**secret reconstruction**] $\mathcal{R}$ forms $n' \times m$ matrix V using $\mathbf{v}^j$ as follows.

$$\begin{pmatrix} k_{1,1}^{\boldsymbol{c}[1]} & k_{1,2}^{\boldsymbol{c}[2]} & \cdots & k_{1,m}^{\boldsymbol{c}[m]} \\ k_{2,1}^{\boldsymbol{c}[1]} & k_{2,2}^{\boldsymbol{c}[2]} & \cdots & k_{2,m}^{\boldsymbol{c}[m]} \\ \vdots & \vdots & & \vdots \\ k_{n',1}^{\boldsymbol{c}[1]} & k_{n',2}^{\boldsymbol{c}[m]} & \cdots & k_{n',m}^{\boldsymbol{c}[m]} \end{pmatrix}$$

   For $i \in [n']$, $\mathcal{R}$ reconstructs the public string $\boldsymbol{r}$ using $\mathbf{v}_i$ and outputs $\{i \in [n'] |$ the values in $\mathbf{v}_i$ can reconstruct $\boldsymbol{r}.\}$

Fig. 3. Secure Wildcard Pattern Matching Protocol

Theorem 1. *Assuming that the 1-out-of-3 oblivious transfer is secure against semi-honest adversaries and the secret sharing scheme satisfies that all shares are necessary and sufficient to reconstruct the secret, the protocol Π_{WPM} securely computes the functionality $\mathcal{F}_{WPM}$ in the semi-honest setting.*

Proof. We prove Theorem 1 in a hybrid model where a trusted party is used to compute the oblivious transfer functionality $\mathcal{F}_{\mathrm{OT}_3^1}$. We separately prove the case that $\mathcal{S}$ is corrupted and the case that $\mathcal{R}$ is corrupted. The form proof is available on https://github.com/Cathysrm/Proof-of-security for the lack of space.

5 Performance Evaluation

5.1 Complexity Analysis

In this section, we analyze the efficiency of our scheme by comparing it with the most representative secure wildcard pattern matching protocol, SWiM [7], the recently proposed Zarezadeh *et al.*'s protocol [22] and [6].

The main cost of our protocol appears in all OTs in step 4. We use the notation $\mathrm{OT}_3^1\text{-}\binom{m}{l}$ to denote m instances of 1-out-of-3 string-OT where the string is l bits long. Consider the case of $|T| = |p| = m$, $\mathrm{OT}_3^1\text{-}\binom{m}{\kappa}$ are required in our protocol. According to the OT extension technique [19], any number of OTs can be obtained with communication proportional to the total size of parties' inputs and computation proportional to the size of the security parameter. Thus it is easy to see that the total communication cost of $\mathrm{OT}_3^1\text{-}\binom{m}{\kappa}$ is the communication cost of implementing "base OT" instances plus $2\kappa m$ bits transferred for symmetric-key operations between $\mathcal{S}$ and $\mathcal{R}$. These base OTs has $O(\kappa m)$ communication complexity. Therefore, we can conclude that the communication cost of $\mathrm{OT}_3^1\text{-}\binom{m}{\kappa}$ is $O(\kappa m)$ bits. The computation cost can be reduced to $O(\kappa)$ exponentiations. As to $|T| > |p|$, we make use of a batched version of 1-out-of-3 OT to transfer the string of length $\kappa(n-m+1)$ and thus avoid the requirement of $m(n-m+1)$ instances of OT_3^1 on κ-bit strings. Consequently, our proposed protocol in the semi-honest model has $O(\kappa)$ computation complexity, and the total cost of communication of executing m OTs each of length $\kappa(n-m+1)$ would be $O(\kappa mn)$ (the security parameter κ can be viewed as a constant).

As can be seen in Table 1, we summarize the performance of the above three protocols and our scheme. Both the research [22] and [6] focus mainly on homomorphic encryption. Compared with our scheme, their protocol has lower communication costs. But in terms of computation, in addition to exponentiations, it usually involves complicated encryption operations and massive multiplications on encrypted data, which requires higher computing capacity for a client. The SWiM protocol is also based on OT and requires only a small number of public-key primitives plus some symmetric-key operations from OT extension. It involves m instances of 1-out-of-2 OT on $(n-m+1)$-bit strings in the OT phase where the communication cost is $O(mn)$ bits. However, we note that the SWiM protocol still needs to perform private equality tests (PEQT) of strings

behind the OT phase and thus this finally leads to two rounds of communication, while the proposed protocol requires only single-round of communication. The remarkable thing here is that we consider merely the communication rounds in the online phase. Furthermore, in the most basic PEQT protocol, $\mathcal{S}$ and $\mathcal{R}$ check whether their l-bit strings x and y are equal by executing random OT_2^1-$\binom{l}{\kappa}$, where $\mathcal{R}$ utilizes $y[i]$ as its choice and $\mathcal{S}$ acts sender with input random κ-bit strings (s_0^i, s_1^i). After that, $\mathcal{R}$ obtains $s_{y[i]}^i$ and then computes $str_{\mathrm{R}} = \oplus_{i=1}^{l} s_{y[i]}^i$. $\mathcal{S}$ computes $str_{\mathrm{S}} = \oplus_{i=1}^{l} s_{x[i]}^i$ where $s_{x[i]}^i$ corresponds to the i-bit of x and sends this value to $\mathcal{R}$. $\mathcal{R}$ determines that $x = y$ iff $str_{\mathrm{R}} = str_{\mathrm{S}}$. Therefore, for each PEQT instance of l-bit strings, it requires l instances of OT_2^1 on κ-bit strings and causes $O(\kappa l)$ communication costs. In the case of the SWiM protocol, there are $n-m+1$ instances of PEQT on m-bit string, so $O(\kappa m(n-m+1))$ communication costs are required. Overall, we have the same communication and computation complexity to the SWiM protocol but better communication rounds.

5.2 Experimental Performance

We analysis the efficiency of the proposed protocol through some experimental results in this section.

Since the main cost of our protocol comes from 1-out-of-3 OT, we performed the experiments on m instances of OT_3^1 on $\kappa(n-m+1)$-bit strings instead of implementing the whole protocol. Our implementation was done in libOTe library [23]. All runs have been taken on a virtual machine with 4GB RAM and 8 cores (the host machine is Intel Core i5-10210U 2.11 GHz with 20 GB RAM). The running time of our scheme compared with SWiM is shown in Table 2 and all running times are reported as the average over 20 trials.

As experimental results show, the proposed scheme is more significantly efficient than the protocols of SWiM in small-scale text/pattern, taking 0.034 seconds to conduct a wildcard pattern matching operation for text length $|T| = 10^3$ and pattern length $|p| = 10^2$. We see a 27.4× improvement in running time compared to SWiM. However, considering the larger values for size of text/pattern, our performance improvement is unsatisfactory or even worse in the extreme case where the length of the text is much greater than that of the pattern. Due to the limited computing power, the detailed results of our performance at scale can not be provided. We can conclude that the proposed scheme is optimized for the case where the size of the text is approximate to that of the pattern. For instance $|T| = |p| = 10^3$, our scheme needs to generate 1000 instances of OT_3^1 on κ-bit strings and finally takes 0.039 s. Using the same parameters, the SWiM protocol results in 1.019 seconds on executing 1000 instances of OT_2^1 on κ-bit strings and one PEQT instance of 1000-bit strings. This is a 26.1× improvement.

Table 1. Complexity comparison for text length = n, pattern = m, computation security parameter κ, a finite alphabet Σ

work	security	method	communication	round	computation		
					expo.	multi.	enc.
[7]	semi.	OT	$O(mn)$	2	$O(\kappa)$	–	–
[22]	mal.	HE	$O(m+n)\kappa)$	2	–	$O(mn)$	$O(m\|\Sigma\|)$
[6]	semi.	HE	$O(n)$	2	$O(n)$	$O(mn)$	$O(m(\|\Sigma\|+m))$
Ours	semi.	OT+SS	$O(mn)$	1	$O(\kappa)$	–	–

6 Optimizations

In this section, considering that the main computing and communication overhead in the WPM protocol comes from OT, we give two optimizations on securely computing functionality $\mathcal{F}_{\mathrm{OT}_3^1}$.

6.1 Online/Offline OT

We briefly describe how the protocol can be modified so that most of the cost can be incurred in an offline phase before the parties' inputs are known. The idea of precomputing OT could trace back to Beaver's work [24] where a precomputing 1-out-of-2 OT construction is given, we apply this idea to a more general case. See Fig. 4 for a full description.

On the basis of the precomputing protocol, we are ready now to briefly describe how the protocol Π_{WPM} can be modified so that most of the cost can be incurred in an offline phase before the parties' inputs are known. In brief, we are able to run all OTs in Step 4 of the protocol by invoking our precomputing protocol. First, the receiver uses a random $\boldsymbol{\sigma}[j] \in \{0,1,2\}$ as its OT choice. The sender chooses a $n' \times m$ matrix W randomly and takes as OT input $\mathbf{w}^i$ denoting the i-th column of matrix W. The each element of this matrix $\mathbf{w}_{i,j}$ is denoted by a tuple $\vec{\boldsymbol{\gamma}}_{i,j} = (\gamma_{i,j}^0, \gamma_{i,j}^1, \gamma_{i,j}^2)$. Later, upon learning p, the receiver sends $\theta = \boldsymbol{c}[j] - \boldsymbol{\sigma}[j] \mod 3$ to the sender, where $\boldsymbol{c}[j]$ denotes the i-th bit of p. As the sender learns its input T, it prepares the value tuple $\vec{\boldsymbol{k}}_{i,j} = (k_{i,j}^0, k_{i,j}^1, k_{i,j}^2)$ and computes $\vec{\boldsymbol{z}}_{i,j} = \vec{\boldsymbol{k}}_{i,j} \oplus \vec{\boldsymbol{\gamma}}_{i,j}$ as follows: $z_{i,j}^a = k_{i,j}^a \oplus \gamma_{i,j}^b$ where $b = a - \theta \mod 3$. The receiver can compute $k_{i,j}^{\boldsymbol{c}[j]} = \gamma_{i,j}^{\boldsymbol{\sigma}[j]} \oplus z_{i,j}^a$ after receiving $\vec{\boldsymbol{z}}_{i,j}$ from the sender. By the precomputation technique, we are able to transfer most of the $O(nm)$ communication to the offline phase, and the resulting protocol is still secure.

6.2 Proxy OT

So far as we know, all known oblivious transfer protocol relies on a large number of asymmetric-key operations, which are typically implemented by modular

Table 2. Comparison of the total runtime (in seconds) for the text of length $|T| = n$, the pattern of length $|p| = m$

$\|p\|$	$\|T\|$	SWiM	Ours	Improved
10	10^3	0.846	0.025	33.8×
	10^4	0.953	0.031	30.7×
	10^5	1.073	0.098	10.9×
10^2	10^3	0.930	0.034	27.4×
	10^4	1.011	0.097	10.4×
	10^5	2.052	0.684	3.0×
10^3	10^3	1.019	0.039	26.1×
	10^4	1.122	0.627	1.8×
	10^5	2.227	–	–
2^8	2^{12}	0.993	0.098	10.1×
	2^{14}	1.139	0.334	3.4×
	2^{16}	1.771	1.563	1.1×
2^{10}	2^{12}	1.072	0.348	2.3×
	2^{14}	1.596	1.388	1.2×
	2^{16}	2.139	–	–

Parameters:

- Two parties: *Sender* $\mathcal{S}$ and *Receiver* $\mathcal{R}$.
- A security parameter κ.
- Ideal functionality $\mathcal{F}_{\mathrm{OT}_3^1}$ primitive.

Input of $\mathcal{S}$: Message set $M = \{m_0, m_1, m_2\}$.
Input of $\mathcal{R}$: Choice σ.
Protocol:

- Precomputing Phase:
  1. $\mathcal{S}$ generates random message set $M' = \{m'_0, m'_1, m'_2\}$ and sends $\mathcal{F}_{\mathrm{OT}_3^1}$ M'. Here $|m'_i| = |m_i| = l$.
  2. $\mathcal{R}$ sends $\mathcal{F}_{\mathrm{OT}_3^1}$ a random choice σ' and receives message $\hat{m} = m'_{\sigma'}$.
- Online Phase:
  1. $\mathcal{R}$ computes $\delta = \sigma - \sigma' \mod 3$ and sends δ to $\mathcal{S}$.
  2. After receiving δ, $\mathcal{S}$ computes $x_i = m_i \oplus m'_j$, here $j = i - \delta \mod 3$.
  3. $\mathcal{R}$ receives m_σ by computing $x_\sigma \oplus \hat{m}$.

Fig. 4. Generic Precomputing OT_3^1 Protocol

exponentiations, that are dense computational tasks. The computational overhead of oblivious transfer is usually more critical than that in communication. The construction of Ishai *et al.* [19] shows an efficient extension technique with

the additional symmetric-key operation to achieve massive effective OT, which reduces the number of asymmetric operations. However, in many certain scenarios, the receiver with limited computational resources *e.g.,* a handheld device can not undertake such intensive computational tasks even using OT extension technology. We wish to minimize the computational task of the receiver. Therefore, we use *proxy oblivious transfer*, a variant of oblivious transfer proposed in [25], to further reduce the computation overhead of the receiver.

In *proxy oblivious transfer* protocol, there are three parties: A sender that holds two messages m_0 and m_1, and a receiver with a choice $\sigma \in \{0, 1\}$, as well as a third party, the proxy, which has no inputs and serves as the receiver's proxy to learn the chosen item. At the end of the protocol, the proxy receives the output m_σ without learning the choice σ, while the sender and receiver learn nothing.

Our protocol can be implemented on the basis of the 1-out-of-3 proxy OT. It is remarkable that this improvement is particularly useful for the receiver with low computational power, since most of the computational overhead is transferred to the proxy, and it can actually compute all the exponentiations in the preprocessing phase.

7 Conclusion

In this paper, we transformed the wildcard pattern matching problem into reconstruction of a shared secret by combining XOR-secret-sharing and 1-out-of-3 OT and presented an efficient protocol with security against semi-honest adversary. The proposed protocol has the same communication and computation complexity to the state-of-the-art solutions but better round complexity.

Acknowledgement. This work is supported by the Taishan Scholars Program, National Natural Science Foundation of China (No. 61702218, 62172258, 61672262), Shandong Provincial Natural Science Foundation (No. ZR2021LZH007, ZR2019LZH 015), Shandong Provincial Key Research and Development Project (No. 2019GGX10 1028, 2018CXGC0706, 2021SFG C0401), Project of Independent Cultivated Innovation Team of Jinan City (No. 2018GXRC002).

References

1. Yao, A.C.: Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science (SFCS 1982), pp. 160–164. IEEE (1982)
2. Zhao, C., Zhao, S., Zhao, M., Chen, Z., Gao, C.-Z., Li, H., Tan, Y.: Secure multiparty computation: theory, practice and applications. Inf. Sci. **476**, 357–372 (2019)
3. Xu, G., Li, H., Ren, H., Lin, X., Shen, X.S.: DNA similarity search with access control over encrypted cloud data. IEEE Trans. Cloud Comput. **10**, 1233–1252 (2020)
4. Namjoshi, K., Narlikar, G.: Robust and fast pattern matching for intrusion detection. In: 2010 Proceedings IEEE INFOCOM, pp. 1–9. IEEE (2010)

5. Hazay, C., Toft, T.: Computationally secure pattern matching in the presence of malicious adversaries. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 195–212. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_12
6. Zarezadeh, M., Mala, H., Ladani, B.T.: Secure parameterized pattern matching. Inf. Sci. **522**, 299–316 (2020)
7. Kolesnikov, V., Rosulek, M., Trieu, N.: SWiM: secure wildcard pattern matching from OT extension. In: Meiklejohn, S., Sako, K. (eds.) FC 2018. LNCS, vol. 10957, pp. 222–240. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-662-58387-6_12
8. Wei, X., Zhao, M., Xu, Q.: Efficient and secure outsourced approximate pattern matching protocol. Soft. Comput. **22**(4), 1175–1187 (2018)
9. Troncoso-Pastoriza, J.R., Katzenbeisser, S., Celik, M.: Privacy preserving error resilient DNA searching through oblivious automata. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 519–528 (2007)
10. Frikken, K.B.: Practical private DNA string searching and matching through efficient oblivious automata evaluation. In: Gudes, E., Vaidya, J. (eds.) DBSec 2009. LNCS, vol. 5645, pp. 81–94. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03007-9_6
11. Gennaro, R., Hazay, C., Sorensen, J.S.: Text search protocols with simulation based security. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 332–350. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13013-7_20
12. Katz, J., Malka, L.: Secure text processing with applications to private DNA matching. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, pp. 485–492 (2010)
13. Baron, J., El Defrawy, K., Minkovich, K., Ostrovsky, R., Tressler, E.: 5PM: secure pattern matching. J. Comput. Secur. **21**(5), 601–625 (2013)
14. Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., Koshiba, T.: Privacy-preserving wildcards pattern matching using symmetric somewhat homomorphic encryption. In: Susilo, W., Mu, Y. (eds.) ACISP 2014. LNCS, vol. 8544, pp. 338–353. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08344-5_22
15. Zarezadeh, M., Mala, H.: Secure parameterized multi-pattern matching in multi-text owner setting. In: 2021 18th International ISC Conference on Information Security and Cryptology (ISCISC), pp. 6–12. IEEE (2021)
16. Qin, H., Wang, H., Wei, X., Xue, L., Lei, W.: Privacy-preserving wildcards pattern matching protocol for IoT applications. IEEE Access **7**, 36094–36102 (2019)
17. Rabin, M.O.: How to exchange secrets with oblivious transfer. Cryptology ePrint Archive (2005)
18. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_9
19. Kolesnikov, V., Kumaresan, R.: Improved OT extension for transferring short secrets. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 54–70. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_4
20. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
21. Blakley, G.R.: Safeguarding cryptographic keys. In: International Workshop on Managing Requirements Knowledge, p. 313. IEEE Computer Society (1979)
22. Zarezadeh, M., Mala, H., Ladani, B.T.: Efficient secure pattern matching with malicious adversaries. IEEE Trans. Dependable Secure Comput. **19**, 1407–1419 (2020)

23. Rindal, P.: libOTe: an efficient, portable, and easy to use oblivious transfer library (2018)
24. Beaver, D.: Precomputing oblivious transfer. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 97–109. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-44750-4_8
25. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: Proceedings of the 1st ACM Conference on Electronic Commerce, pp. 129–139 (1999)