Oblivious Transfer from Rerandomizable PKE

Shuaishuai Li[1,2], Cong Zhang[1,2], and Dongdai Lin[1,2(✉)]

[1] SKLOIS, Institute of Information Engineering, CAS
[2] School of Cyber Security, University of Chinese Academy of Sciences
{lishuaishuai,zhangcong,ddlin}@iie.ac.cn

Abstract. The relationship between oblivious transfer (OT) and public-key encryption (PKE) has been studied by Gertner et al. (FOCS 2000). They showed that OT can be constructed from special types of PKE, i.e., PKE with oblivious sampleability of public keys or ciphertexts. In this work, we give new black-box constructions of OT from PKE without any oblivious sampleability. Instead, we require that the PKE scheme is rerandomizable, meaning that one can use the public key to rerandomize a ciphertext into a fresh ciphertext. We give two different OT protocols with different efficiency features based on rerandomizable PKE. For 1-out-of-n OT, in our first OT protocol, the sender has sublinear (in n) cost, and in our second OT protocol, the cost of the receiver is independent of n. As a comparison, in the PKE-based OT protocols of Gertner et al., both the sender and receiver have linear cost.

Keywords: Oblivious Transfer · Public-Key Encryption · Rerandomizable

1 Introduction

Oblivious transfer (OT) [22] is a two-party cryptographic protocol that allows a sender to obliviously transfer a message to a receiver. OT is a fundamental building block in secure multiparty computation (MPC) and plays an important role in many famous MPC protocols, such as the gabled circuits [25] and GMW protocol [13]. In particular, the result of [13] implies that OT is complete for MPC. That is, any function can be securely computed with only OT in hand. For this reason, OT serves as one of the most important primitive in public-key cryptography.

In an OT protocol, there are two parties called the sender and receiver, where the sender takes two messages x_0, x_1 as inputs and the receiver takes a choice bit b as input. At the end of the protocol, the receiver obtains the message x_b while the sender receives nothing. This protocol is also known as 1-out-of-2 OT (the receiver obtains one message out of two messages). The work of [3] further generalized OT and introduced 1-out-of-n OT (under the name of all-or-nothing disclosure of secrets). In a 1-out-of-n OT protocol, the sender takes n messages $x_1, \ldots, x_n$ as inputs, and the receiver takes an index $i \in [n]$ as input. At the end of the protocol, the receiver outputs x_i.

Studying the relationships between cryptographic primitives is of great significance for understanding their powers and limitations. In the work of [12], Gertner et al. studied the relationship between OT and several other cryptographic primitives such as public-key encryption (PKE), key agreement and trapdoor permutations. They showed that OT and PKE are incomparable with respect to black-box reductions, which implies that one cannot obtain OT (resp. PKE) with only PKE (resp. OT) in hand. To further understand the relationship between OT and PKE, they considered the problem of constructing OT from PKE with special properties. In particular, they found that PKE with oblivious sampleability of public keys or ciphertexts is sufficient for OT, where oblivious sampleability of public keys (resp. ciphertexts) means that one can sample a public key (resp. ciphertext) without knowing the corresponding secret key (resp. plaintext). In this work, we follow this work and continue to study the relationship between OT and PKE. More precisely, we seek to present new constructions of OT from PKE without any oblivious sampleability. In fact, another sufficient property of PKE for constructing OT is homomorphism. PKE with homomorphism is also referred as homomorphic encryption (HE). For example, the work of [24] used additively HE (AHE) to design an efficient OT protocol, which tells us that AHE is sufficient for OT. A natural question is that whether we can base OT on weaker HE schemes than AHE, or to be formally, we ask the following question.

What is the minimum homomorphism required to construct OT?

1.1 Our Contribution

Main Result. In this work, we study the above question, and our main result is that OT could be based on rerandomizable PKE, which is a special PKE scheme with the property that one can rerandomize a ciphertext into a fresh ciphertext encrypting the same message. Note that rerandomization can be viewed as the minimum homomorphism in the sense that it only allows one to compute the identity function. We remark that many existing PKE schemes have the property of rerandomization, including the ElGamal scheme [10], the Paillier scheme [19], and the Regev scheme [23]. In particular, the work of [21] constructed a rerandomizable PKE scheme without any homomorphism other than rerandomization[3].

OT with New Efficiency Features. Our another contribution is that we design PKE-based OT protocols with new efficiency features. As we have said, Gertner et al. [12] showed that OT can be constructed from PKE with oblivious sampleability of public keys or ciphertexts. While their OT constructions are very efficient and general, the costs of both parties in their protocols are linear

[3] This scheme achieves a slightly weaker variant of IND-CCA security called replayable CCA (RCCA) security, which is introduced by [5]. As stated in [5], RCCA security is sufficient for many applications of IND-CCA secure PKE (authentication, key exchange, etc.).

in n. Consider the setting where the two parties have different data processing capability (e.g., the two parties use different machines with different computational powers, or they are in different network settings with different data transfer capability) and we want to minimize the running time of the protocol[4]. In such an unbalanced setting, if we can reduce the cost of the party with lower data processing capability than the other party, then the resulting protocol may have less running time even if the total cost remains the same or is even higher. Our OT constructions allow us to obtain OT protocols with new efficiency features which are more suitable for the aforementioned unbalanced setting. More precisely, we obtain the following two OT protocols based on rerandomizable PKE.

- A two-pass OT protocol where the costs of the sender and receiver are $O(n/\log n)$ and $O(n^{1+\varepsilon}/\log n)$ for any constant $\varepsilon > 0$, respectively.
- A three-pass OT protocol where the costs of the sender and receiver are $O(n)$ and $O(1)$, respectively.

In our first OT protocol, the sender has sublinear cost, hence it is more suitable for the setting where the sender has lower data processing capability than the receiver. In our second OT protocol, the receiver has cost independent of n, hence it is more suitable for the setting where the receiver has lower data processing capability than the sender.

1.2 Technical Overview

Now let us give an overview of the techniques used in our OT constructions. For simplicity, we focus on bit-OT where each message is a single bit (extending our results to string-OT is direct, and we will discuss this in Appendix A). Assume that the sender takes $x = (x_1, \dots, x_n) \in \{0,1\}^n$ as input and the receiver takes an index $i \in [n]$ as input. Our goal is to let the receiver obtain x_i.

OT with Sublinear Sender-Cost. Our starting point is that if we define the function $f_i(z) = z_i$ over $\{0,1\}^n$, then $f_i(x)$ is exactly x_i. We first design a toy protocol with exponential cost to let the receiver get the value of x_i.

1. The receiver samples a key pair and uses the public key to encrypt $f_i(z)$ to c_z for each $z \in \{0,1\}^n$. Then it sends the public key and all the ciphertexts to the sender.
2. The sender uses the public key to rerandomize c_x and sends the resulting ciphertext to the receiver.
3. Finally, the receiver decrypts the received ciphertext to get x_i.

The correctness of the above OT protocol is easy to verify. As for the security, note that the sender does not know the secret key, so it knows nothing about i from the received ciphertexts. Moreover, the receiver only receives a ciphertext c

[4] In the works of [7,6,9], the authors studied private set intersection (PSI) in a similar setting where one party may limited resources for computation and storage.

which is a rerandomization of a ciphertext encrypting x_i, and by the property of rerandomizable PKE, c is indistinguishable from a fresh encryption of x_i, which means that the receiver only obtains x_i from c.

In the toy protocol, although the cost of the sender is independent of n, the cost of the receiver is exponential in n (the receiver must compute and send 2^n ciphertexts). Namely, the toy protocol is an "inefficient" OT protocol. Now we show how to optimize the toy protocol such that the cost of the receiver is polynomial in n while the cost of the sender remains sublinear.

To achieve our goal, we use a reduction from long OT to short OT. More precisely, for any $n = t(m-1)$, a 1-out-of-n OT protocol can be constructed using t calls to a 1-out-of-m OT protocol. Moreover, the cost of this 1-out-of-n OT protocol is about t times that of the 1-out-of-m OT protocol. We refer to Section 3.2 for more details about this reduction. If we use the aforementioned "inefficient" OT protocol as the underlying 1-out-of-m OT protocol, then in the resulting 1-out-of-n OT protocol, the costs of the sender and receiver will be $O(t)$ and $O(t2^m)$, respectively. By setting $m = \varepsilon \log n$ for some positive constant ε, we can obtain an OT protocol where the costs of the sender and receiver are $O(n/\log n)$ and $O(n^{1+\varepsilon}/\log n)$, respectively. This is the desired OT protocol.

OT with Constant Receiver-Cost. Our sender-efficient OT protocol could be cast into the following framework: the receiver sends a set of ciphertexts containing an encryption of x_i; the sender uses x to select the encryption of x_i. Now we swap the roles of the sender and the receiver. Concretely, we let the sender send (a set of ciphertexts containing an encryption of x_i) first and then the receiver select (the encryption of x_i). Our starting point is that if we define the function $g_x(j) = x_j$ over $[n]$, then $g_x(i)$ is exactly x_i. We first describe the following "insecure" protocol.

1. The sender samples a key pair and uses the public key to encrypt $g_x(j)$ to c_j for each $j \in [n]$. Then it sends the public key and all the ciphertexts to the receiver.
2. The receiver uses the public key to rerandomize c_i and sends the resulting ciphertext to the sender.
3. Finally, the sender decrypts the received ciphertext to get x_i and sends x_i to the receiver.

The above protocol is insecure due to that the sender knows the value of x_i, which leaks information about i. To solve this problem, we let the receiver randomize the underlying plaintext of c_i. Fortunately, this can be done using rerandomization. Concretely, in the first step, we let the sender send encryptions of x_j and $x_j \oplus 1$ (instead of just x_j). In this way, the receiver has encryptions of x_i and $x_i \oplus 1$ (written as $c_{i,0}$ and $c_{i,1}$, respectively). Now, the receiver can randomize the underlying plaintext of c_i: it samples a random bit r, and then it computes a ciphertext e as a rerandomization of $c_{i,0}$ if $r = 0$ and $c_{i,1}$ otherwise. It is easy to verify that e is an encryption of $x_i \oplus r$. Now we can describe our secure OT protocol.

1. The sender samples a key pair and uses the public key to encrypt x_j to $c_{j,0}$ and $x_j \oplus 1$ to $c_{j,1}$ for each $j \in [n]$. Then it sends the public key and all the ciphertexts to the receiver.
2. The receiver samples a random bit r, and then it computes a ciphertext e as a rerandomization of $c_{i,0}$ if $r = 0$ and $c_{i,1}$ otherwise. Then it sends e to the sender.
3. The sender decrypts the received ciphertext to get s and sends s to the receiver.
4. Finally, the receiver computes $x_i = s \oplus r$.

The correctness of the above OT protocol is easy to verify. As for the security, note that s is a random bit due to r is random, hence the sender cannot get any information about i. Moreover, the receiver only obtains x_i as it does not know the secret key. Finally, it is easy to see that in our protocol, the cost of the sender is linear in n, and the cost of the receiver is independent of n.

1.3 Related Primitives

The most relevant primitives for OT are probably private information retrieval (PIR) [8] and symmetrically PIR (SPIR) [11], where SPIR is a stronger variant of PIR. If only security requirements are considered, SPIR is in fact equivalent to OT. However, PIR and SPIR typically are used in a different context where n is large (e.g., $n = 2^{20}$), and they additionally require that the total communication cost is sublinear in n. To date, the state-of-the-art PIR [16,2,20,18,1,17,14] and SPIR [15] protocols are based on fully-homomorphic encryption (FHE).

2 Preliminaries

Notations. Let κ be the security parameter. For any two integers i, j, we denote $[i, j]$ the set $\{i, \cdots, j\}$ and abbreviate $[1, j]$ by $[j]$. If $i > j$, $[i, j]$ represents the empty set $\varnothing$. For any two distributions $\mathcal{X}$ and $\mathcal{Y}$, we say that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable, denoted as $\mathcal{X} \approx_c \mathcal{Y}$, if no PPT algorithm can distinguish these two distributions. We say that $\mathcal{X}$ and $\mathcal{Y}$ are statistically indistinguishable, denoted as $\mathcal{X} \approx_s \mathcal{Y}$, if their statistical distance is negligible. For any set A, we use $a \leftarrow A$ to represent that we sample a random element a from A in a uniform way.

2.1 Oblivious Transfer

In this work, we prove the security of our protocols in the universally composable (UC) framework, and we refer to [4] for more detail about this framework. Now we describe the ideal functionality for OT.

Definition 1 (Ideal Oblivious Transfer Functionality $\mathcal{F}_{\mathsf{OT}}$). *The ideal OT functionality $\mathcal{F}_{\mathsf{OT}}$ is a two-party functionality which receives n bits $x_1, \ldots, x_n$ from a party P_0 called the sender and an index $i \in [n]$ from the other party P_1 called the receiver. $\mathcal{F}_{\mathsf{OT}}$ returns x_i to P_1.*

2.2 Rerandomizable Public-Key Encryption

Our OT protocols make use of a rerandomizable PKE scheme, and we recall the definition of rerandomizable PKE in this section.

Definition 2 (Rerandomizable Public-Key Encryption). *A rerandomizable public-key encryption (PKE) consists of four algorithms* $\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}$, *and* Rand. *Let* $\mathcal{M}$ *be the plaintext space,* $\mathcal{C}$ *be the ciphertext space,* $\mathcal{PK}$ *be the public key space, and* $\mathcal{SK}$ *be the secret key space. These four algorithms are defined as follows.*

- $\mathsf{Keygen}(1^\kappa)$*: on input the security parameter* κ*, output a key pair* $(pk, sk) \in \mathcal{PK} \times \mathcal{SK}$.
- $\mathsf{Enc}(p, pk)$*: on input a plaintext* $p \in \mathcal{M}$ *and a public key* $pk \in \mathcal{PK}$*, outputs a ciphertext* $c \in \mathcal{C}$.
- $\mathsf{Dec}(c, sk)$*: on input a ciphertext* $c \in \mathcal{C}$ *and a secret key* $sk \in \mathcal{SK}$*, outputs a plaintext* $p \in \mathcal{M}$.
- $\mathsf{Rand}(c, pk)$*: on input a ciphertext* $c \in \mathcal{C}$ *and a public key* $pk \in \mathcal{PK}$*, outputs a ciphertext* $c' \in \mathcal{C}$.

Rerandomizable PKE requires the following properties.

- ***Correctness.*** *For any plaintext* $p \in \mathcal{M}$*, it holds that*

$$\Pr[\mathsf{Dec}(\mathsf{Enc}(p, pk), sk) = p] \geq 1 - \mathsf{neg}(\kappa)$$

  where $(pk, sk) \leftarrow \mathsf{Keygen}(1^\kappa)$.
- ***IND-CPA Security.*** *For any two plaintexts* p_0, p_1*, we have*

$$\mathsf{Enc}(p_0, pk) \approx_c \mathsf{Enc}(p_1, pk)$$

  where $(pk, sk) \leftarrow \mathsf{Keygen}(1^\kappa)$.
- ***Ciphertext Rerandomizable.*** *For any plaintext* $p \in \mathcal{M}$*, it holds that*

$$\mathsf{Rand}(c, pk) \approx_s \mathsf{Enc}(p, pk)$$

  where $(pk, sk) \leftarrow \mathsf{Keygen}(1^\kappa)$ *and* $c \leftarrow \mathsf{Enc}(p, pk)$.

2.3 Reviewing the Previous PKE-Based OT Protocols

In this section, we review the OT protocols of Gertner et al. [12]. Their protocols are based on PKE with oblivious sampleability of public keys or ciphertexts. We first define such special PKE.

Definition 3 (PKE with Oblivious Sampleability of Public Keys). *We say that a PKE scheme* $(\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ *has oblivious sampleability of public keys if there exists a PPT algorithm* OsPk *satisfying that*

$$\{pk | pk \leftarrow \mathsf{OsPk}(1^\kappa)\} \approx_s \{pk | (pk, sk) \leftarrow \mathsf{Keygen}(1^\kappa)\}.$$

Definition 4 (PKE with Oblivious Sampleability of Ciphertexts). *We say that a PKE scheme* ($\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}$) *has oblivious sampleability of ciphertexts if there exists a PPT algorithm* OsCt *satisfying that*

$$\{c | c \leftarrow \mathsf{OsCt}(pk)\} \approx_s \{c | p \leftarrow \mathcal{M}, c \leftarrow \mathsf{Enc}(p, pk)\}$$

where $(pk, sk) \leftarrow \mathsf{Keygen}(1^\kappa)$*.*

Now let us describe the OT protocols of [12]. The first is based on PKE with oblivious sampleability of public keys, and its description is in the following.

Protocol $\mathsf{OT}_{\mathsf{ospk}}$

Input: Let ($\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{OsPk}$) be a PKE scheme with oblivious sampleability of public keys. The sender P_0 takes n bits $x_1, \ldots, x_n$ as inputs, and the receiver P_1 takes an index $i \in [n]$ as input.
Output: P_1 gets x_i as output.

1. P_1 samples a key pair $(pk_i, sk_i) \leftarrow \mathsf{Keygen}(1^\kappa)$ and $pk_j \leftarrow \mathsf{OsPk}(1^\kappa)$ for each $j \in [n] \backslash \{i\}$. Then, P_1 sends $\{pk_j\}_{j \in [n]}$ to P_0.
2. P_0 computes a ciphertext $e_j = \mathsf{Enc}(x_j, pk_j)$ for each $j \in [n]$ and sends $\{e_j\}_{j \in [n]}$ to the receiver.
3. P_1 decrypts $u \leftarrow \mathsf{Dec}(e_i, sk_i)$ and returns u.

Security Analysis of $\mathsf{OT}_{\mathsf{ospk}}$**.** The correctness is easy to verify. Let us discuss the privacy. Firstly, note that pk_i is indistinguishable from each other pk_j, hence the sender P_0 cannot obtain any information about i. Secondly, for each $j \neq i$, since receiver P_1 does not know the decryption key of pk_j (pk_j is obliviously sampled), it cannot decrypt the ciphertext e_j.

Complexity of $\mathsf{OT}_{\mathsf{ospk}}$**.** In the protocol, both the sender and receiver have cost linear in n. Moreover, the protocol only takes two passes.

The second OT protocol of [12] is based on PKE with oblivious sampleability of ciphertexts, and its description is in the following.

Protocol $\mathsf{OT}_{\mathsf{osct}}$

Input: Let ($\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{OsCt}$) be a PKE scheme with oblivious sampleability of ciphertexts. The sender P_0 takes n bits $x_1, \ldots, x_n$ as inputs, and the receiver P_1 takes an index $i \in [n]$ as input.
Output: P_1 gets x_i as output.

1. P_0 samples a key pair (pk, sk) and sends pk to P_1.
2. P_1 samples a random plaintext r and computes $c_i = \mathsf{Enc}(r, pk)$. Then, it obliviously samples $c_j \leftarrow \mathsf{OsCt}(pk)$ for each $j \in [n] \backslash \{i\}$. Finally, P_1 sends $\{c_j\}_{j \in [n]}$ to P_0.
3. P_0 decrypts $r_j = \mathsf{Dec}(c_j, sk)$ and computes $u_j = x_j \oplus r_j$ for each $j \in [n]$. Then, it sends $\{u_j\}_{j \in [n]}$ to P_1.

4. P_1 computes $z = u_i \oplus r$ and returns z.

Security Analysis of $\mathsf{OT}_{\mathsf{osct}}$. The correctness is easy to verify. Let us discuss the privacy. Firstly, note that each c_i is statistically indistinguishable from each other c_j, hence the sender P_0 cannot obtain any information about i. Secondly, for each $j \neq i$, since the receiver P_1 does not know the decryption key sk, it knows nothing about r_j, which implies that it knows nothing about x_j even with u_j in hand.

Complexity of $\mathsf{OT}_{\mathsf{osct}}$. In the protocol, both the sender and receiver have cost linear in n. Moreover, the protocol takes three passes (the first pass can be executed once for all).

3 Sender-Friendly Oblivious Transfer

In this section, we present our first OT protocol, which is sender-friendly, meaning that the cost of the sender is sublinear in n. Throughout this section, let $(\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rand})$ be a rerandomizable PKE scheme.

3.1 First Attempt: OT with Constant Sender-Cost and Exponential Receiver-Cost

We first give an "inefficient" OT protocol where the cost of the receiver P_1 is exponential in n (the cost of the sender P_0 is independent of n). This protocol proceeds as follows.

1. P_1 samples a key pair and uses the public key to encrypt z_i to c_z for each $z \in \{0,1\}^n$. Then it sends the public key and all the ciphertexts to P_0.
2. P_0 uses the public key to rerandomize c_x and sends the resulting ciphertext to P_1.
3. Finally, P_1 decrypts the received ciphertext to get x_i.

We introduce an optimization to the above protocol. Notice that if all the bits $x_1, \ldots, x_n$ are the same, then c_x is always an encryption of x_1. Therefore, the receiver just needs to send encryptions of the elements in $\{0,1\}^n \backslash \{0^n, 1^n\}$. This optimization has a small improvement to the protocol when n is large. However, if n is small, then this optimization is remarkable and for $n = 2$, it will halve the cost of the receiver. Now we describe the protocol with our optimization.

Protocol $\mathsf{OT}^{\mathsf{sen}}_{\mathsf{rpke}}$

Input: The sender P_0 takes n bits $x_1, \ldots, x_n$ as inputs, and the receiver P_1 takes an index $i \in [n]$ as input. Let $I = \{0^n, 1^n\}$.
Output: P_1 gets x_i as output.

1. P_1 samples a pair of keys $(pk, sk) \leftarrow \mathsf{Keygen}(1^\kappa)$. Then, for each $z = (z_1, \ldots, z_n) \in \{0,1\}^n \backslash I$, P_1 computes $c_z \leftarrow \mathsf{Enc}(z_i, pk)$. Finally, P_1 sends $(pk, \{c_z\}_{z \in \{0,1\}^n \backslash I})$ to P_0.
2. P_0 computes $e \leftarrow \mathsf{Enc}(x_1, pk)$ if $x \in I$ and $e \leftarrow \mathsf{Rand}(c_x, pk)$ otherwise. Then, it sends e to P_1.
3. P_1 computes $u \leftarrow \mathsf{Dec}(e, sk)$ and outputs u.

Complexity of $\mathsf{OT}^{\mathsf{sen}}_{\mathsf{rpke}}$. The sender needs to compute and send a single ciphertext, its cost is independent of n. The receiver needs to compute and send $2^n - 2$ ciphertexts, hence its cost is exponential in n.

Security of $\mathsf{OT}^{\mathsf{sen}}_{\mathsf{rpke}}$. We state the security of $\mathsf{OT}^{\mathsf{sen}}_{\mathsf{rpke}}$ by proving the following theorem.

Theorem 5. *For any $n = O(\log \kappa)$, the protocol $\mathsf{OT}^{\mathsf{sen}}_{\mathsf{rpke}}$ securely realizes the functionality $\mathcal{F}_{\mathsf{OT}}$ in the UC framework.*

Proof. If both the sender and receiver are honest, it is easy to verify that the receiver will obtain the bit x_i. If some party is corrupt, there are two cases to be considered.

`Sender is Corrupt.` In this case, we construct a simulator $\mathcal{S}$ as follows.

- $\mathcal{S}$ samples a key pair (pk, sk) and computes $c_z \leftarrow \mathsf{Enc}(0, pk)$ for each $z \in \{0,1\}^n \backslash I$. Then $\mathcal{S}$ simulates the receiver sending $(pk, \{c_z\}_{z \in \{0,1\}^n \backslash I})$ to the sender.

It remains to show that the environment cannot distinguish the simulated and real executions. We first define Hybrid_j for each $j \in [2^n - 1]$.

- Hybrid_j: $\mathcal{S}_j$ samples a key pair (pk, sk). Then, it computes $c_z \leftarrow \mathsf{Enc}(z_i, pk)$ for each $z \in [1, j-1]$ and $c_z \leftarrow \mathsf{Enc}(0, pk)$ for each $z \in [j, 2^n - 2]$. Finally, $\mathcal{S}_i$ simulates the receiver sending $(pk, \{c_z\}_{z \in \{0,1\}^n \backslash I})$ to the sender.

Note that Hybrid_1 is exactly the simulated execution, and Hybrid_{2^n-1} is the real execution. Now we proceed to show that each two consecutive hybrids are indistinguishable, which will imply that Hybrid_1 and Hybrid_{2^n-1} are indistinguishable because there are total $2^n - 1 = \mathsf{poly}(\kappa)$ hybrids.

For each $j \in [2^n - 2]$, the two hybrids Hybrid_j and Hybrid_{j+1} only differ in the generation of c_j, which is an encryption of 0 in Hybrid_j and an encryption of j_i in Hybrid_{j+1}[5]. By the IND-CPA security of the underlying PKE scheme, we know that an encryption of j_i is indistinguishable from an encryption of 0. Therefore, Hybrid_j and Hybrid_{j+1} are indistinguishable.

`Receiver is Corrupt.` In this case, we construct a simulator $\mathcal{S}$ as follows.

[5] Each j is viewed as a bitstring and j_i is the i-th bit of j.

- $\mathcal{S}$ sends the input of P_1 to $\mathcal{F}_{\mathsf{OT}}$ and receives the output x_i. Then it samples a key pair (pk, sk) and computes $e' \leftarrow \mathsf{Enc}(x_i, pk)$. Finally, it simulates the sender sending e' to the receiver.

Now we show that the simulated and real executions are indistinguishable, which implies that the environment cannot distinguish the simulated and real executions. In the simulated execution, the simulated ciphertext e' is a fresh encryption of x_i under a fresh public key. In the real execution, the ciphertext e is a rerandomization of an encryption of the output u. By the correctness of the protocol, we know that u is x_i. Moreover, the underlying rerandomizable PKE guarantees that a rerandomization of a ciphertext (of any plaintext p) is indistinguishable from a fresh encryption of p under the same public key. Therefore, the simulated ciphertext e' is indistinguishable from the ciphertext e in the real execution. $\square$

3.2 A Reduction from Long OT to Short OT

In this section, we give a reduction from long OT to short OT. Concretely, we can construct 1-out-of-n OT using t calls to 1-out-of-m OT where $n = t(m-1)$. The construction is quite simple, and its description is in the following.

1. The sender P_0 sets $X_j = (x_{(j-1)(m-1)+1}, \ldots, x_{j(m-1)}, 0)$ for each $j \in [t]$.
2. The receiver P_1 sets $i = (i_1 - 1)(m-1) + i_2$ with $i_1 \in [t]$ and $i_2 \in [m-1]$. Then for each $j \in [t]$, P_1 lets q_j be i_2 if $j = i_1$ and m otherwise.
3. P_0 and P_1 parallelly invoke a 1-out-of-m OT protocol t times, where in the j-th execution, P_0 takes X_j as input and P_1 takes q_j as input.
4. P_1 takes the output in the i_1-th execution as its final output.

Proof Sketch of the above OT Construction. Firstly, it is easy to verify that in the i_1-th execution, the output of P_1 will be x_i, which implies that the correctness holds. As for the security, note that in the j-th execution, the output of P_1 will be 0 if $j \neq i_1$. This implies that the simulator will be able to infer the output of each short OT instance from the output of the long OT. Also, the simulator can infer the input of each short OT instance from the input of the long OT. Therefore, to simulate the view of the corrupted party, the simulator just invokes the simulators of all the short OT instances using the inferred inputs and outputs.

3.3 Putting It All Together: OT with Sublinear Sender-Cost and Polynomial Receiver-Cost

We show how to design an OT protocol where the cost of the sender is sublinear in n and the cost of the receiver is polynomial in n. To achieve this goal, we take our OT protocol $\mathsf{OT}^{\mathsf{sen}}_{\mathsf{rpke}}$ as the underlying 1-out-of-m OT protocol in the OT construction described in Section 3.2. As a result, we can derive an OT protocol where the costs of the sender and receiver are $O(t)$ and $O(t2^m)$, respectively ($n = t(m-1)$). By setting $m = \varepsilon \log n$ for any constant $\varepsilon > 0$, we obtain the desired OT protocol where the costs of the sender and receiver are $O(n/\log n)$ and $O(n^{1+\varepsilon}/\log n)$, respectively.

4 Receiver-Friendly Oblivious Transfer

In this section, we present our second OT protocol, which is receiver-friendly, meaning that the cost of the receiver is sublinear in n. More precisely, in our protocol, the costs of the sender and receiver are $O(n)$ and $O(1)$, respectively. Throughout this section, let $(\mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rand})$ be a rerandomizable PKE scheme. Our protocol is described as follows.

Protocol $\mathsf{OT}^{\mathsf{rec}}_{\mathsf{rpke}}$

Input: The sender P_0 takes n bits $x_1, \ldots, x_n$ as inputs, and the receiver P_1 takes an index $i \in [n]$ as input.

Output: P_1 gets x_i as output.

1. P_0 samples a key pair $(pk, sk) \leftarrow \mathsf{Keygen}(1^\kappa)$. Then, for each $j \in [n]$, P_0 computes $c_{j,0} \leftarrow \mathsf{Enc}(x_j, pk)$ and $c_{j,1} \leftarrow \mathsf{Enc}(x_j \oplus 1, pk)$. Finally, P_0 sends $(pk, \{c_{j,0}, c_{j,1}\}_{j\in[n]})$ to P_1.
2. P_1 chooses a random bit r and computes $e \leftarrow \mathsf{Rand}(c_{i,0}, pk)$ if $r = 0$ and $e \leftarrow \mathsf{Rand}(c_{i,1}, pk)$ otherwise. Then, it sends e to P_0.
3. P_0 computes $u \leftarrow \mathsf{Dec}(e, sk)$ and sends u to P_1.
4. P_1 outputs $z = u \oplus r$.

Complexity of $\mathsf{OT}^{\mathsf{rec}}_{\mathsf{rpke}}$. The sender needs to compute and send $2n$ ciphertexts, hence its cost is linear in n. The receiver needs to compute and send a single ciphertext, hence its cost is independent of n.

Security of $\mathsf{OT}^{\mathsf{rec}}_{\mathsf{rpke}}$. We state the security by proving the following theorem.

Theorem 6. *For any $n = \mathsf{poly}(\kappa)$, the protocol $\mathsf{OT}^{\mathsf{rec}}_{\mathsf{rpke}}$ securely realizes the functionality $\mathcal{F}_{\mathsf{OT}}$ in the UC framework.*

Proof. If both the sender and receiver are honest, it is easy to verify that u is exactly $x_i \oplus r$, hence the final output is $z = u \oplus r = x_i$, which guarantees the correctness. Now we proceed to prove the privacy of our protocol. We need to consider two cases.

`Sender is Corrupt.` In this case, we construct a simulator $\mathcal{S}$ as follows.

- $\mathcal{S}$ samples a key pair (pk, sk) and computes $e' \leftarrow \mathsf{Enc}(r', pk)$ with r' being a random bit. Then $\mathcal{S}$ simulates the receiver sending e' to the sender.

Now we show that the simulated and real executions are indistinguishable. In the simulated execution, the simulated ciphertext e' is an fresh encryption of a random bit r'. In the real execution, the ciphertext e is a rerandomization of an encryption of $x_i \oplus r$. Note that $x_i \oplus r$ is random because r is a random bit. Moreover, the underlying rerandomizable PKE guarantees that a rerandomization of a ciphertext (of any plaintext p) is indistinguishable from a fresh encryption of p under the same public key. This implies that the simulated ciphertext e' is indistinguishable from the real ciphertext e. Therefore, the simulated and real executions are indistinguishable.

`Receiver is Corrupt.` In this case, we construct a simulator $\mathcal{S}$ as follows.

- $\mathcal{S}$ sends the input of P_1 to $\mathcal{F}_{\mathsf{OT}}$ and receives the output x_i. Then it samples a key pair (pk, sk) and a random bit r. $\mathcal{S}$ computes $c_{i,0} \leftarrow \mathsf{Enc}(x_i, pk)$ and $c_{i,1} \leftarrow \mathsf{Enc}(x_i \oplus 1, pk)$, and $c_{j,b} \leftarrow \mathsf{Enc}(b, pk)$ for each $j \in [n] \backslash \{i\}, b \in \{0,1\}$.
- Then, $\mathcal{S}$ simulates P_0 sending $(pk, \{c_{j,0}, c_{j,1}\}_{j\in[n]})$ to P_1.
- Finally, $\mathcal{S}$ simulates P_0 sending $u = x_i \oplus r$ to P_1.

It remains to show that the environment cannot distinguish the simulated and real executions. We first define Hybrid_j for each $j \in [n+1]$.

- Hybrid_j: $\mathcal{S}_j$ samples a key pair (pk, sk) and a random bit r_j. Next,
  - If $j \leq i$, then it computes $c_{k,b} \leftarrow \mathsf{Enc}(x_k \oplus b, pk)$ for each $k \in [1, j-1] \cup \{i\}, b \in \{0,1\}$, and $c_{k,b} \leftarrow \mathsf{Enc}(b, pk)$ for each $k \in [j, i-1] \cup [i+1, n], b \in \{0,1\}$.
  - If $j > i$, then it computes $c_{k,b} \leftarrow \mathsf{Enc}(x_k \oplus b, pk)$ for each $k \in [1, j-1], b \in \{0,1\}$, and $c_{k,b} \leftarrow \mathsf{Enc}(b, pk)$ for each $k \in [j, n], b \in \{0,1\}$.

  Then, $\mathcal{S}_j$ simulates P_0 sending $(pk, \{c_{k,0}, c_{k,1}\}_{k\in[n]})$ and $u = x_i \oplus r_j$ to P_1.

Note that Hybrid_1 is exactly the simulated execution, and Hybrid_{n+1} is the real execution. Now we proceed to show that each two consecutive hybrids are indistinguishable, which will imply that Hybrid_1 and Hybrid_{n+1} are indistinguishable because there are total $n + 1 = \mathsf{poly}(\kappa)$ hybrids.

It is easy to see that Hybrid_i and Hybrid_{i+1} are identical, so we only need to show that Hybrid_j and Hybrid_{j+1} for each $j \in [n] \backslash \{i\}$. The two hybrids Hybrid_j and Hybrid_{j+1} only differ in the generation of $(c_{j,0}, c_{j,1})$. In Hybrid_j, $c_{j,0}$ and $c_{j,1}$ are encryptions of 0 and 1, respectively. And in Hybrid_{j+1}, $c_{j,0}$ and $c_{j,1}$ are encryptions of x_j and $x_j \oplus 1$, respectively. By the IND-CPA security of the underlying PKE scheme, the encryptions of 0 and 1 are indistinguishable from that of x_j and $x_j \oplus 1$. Therefore, Hybrid_j and Hybrid_{j+1} are indistinguishable. $\square$

5 Comparision to the Previous OT Protocols Based on Special Types of PKE

In this section, we compare the concrete and asymptotic complexity of our OT protocols and the PKE-based OT protocols of [12]. For the comparison of concrete complexity, we consider 1-out-of-2 OT and use the protocol described in Section 3.1 as our sender-friendly OT protocol[6]. For the comparison of asymptotic complexity, we consider the 1-out-of-n OT and use the protocol described in Section 3.3 as our sender-friendly OT protocol. Moreover, we focus on the round complexity, communication cost of the protocols. In particular, we compare the communication cost on the sender side and the receiver side separately.

[6] Recall that though this protocol has exponential cost, it is still efficient for small n.

5.1 Comparison with Respect to 1-out-of-2 OT

When considering 1-out-of-2 OT, we directly use the protocol described in Section 3.1. The detailed comparison is shown in Table 1.

1-out-of-2 OT	Round Complexity	Sender Communication	Receiver Communication
$\mathsf{OT}_{\mathsf{ospk}}$ ([12])	2 passes	2 cts	2 pks
$\mathsf{OT}_{\mathsf{osct}}$ ([12])	3 passes	1 pk & 2 pts	2 cts
$\mathsf{OT}^{\mathsf{sen}}_{\mathsf{rpke}}$ (Section 3)	2 passes	1 ct	1 pk & 2 cts
$\mathsf{OT}^{\mathsf{rec}}_{\mathsf{rpke}}$ (Section 4)	3 passes	1 pk & 1 pt & 4 cts	1 ct

Table 1. A comparison of the PKE-based 1-out-of-2 OT protocols of [12] and our OT protocols regarding round complexity, communication costs of sender and the receiver. Note that we use pk (resp. pks), pt (resp. pts), and ct (resp. cts) to represent public key (resp. public keys), plaintext (resp. plaintexts), and ciphertext (resp. ciphertexts), respectively.

The comparison illustrates that in some scenarios our protocols may be a better choice. For example, if our goal is optimal round complexity and low sender-communication, then $\mathsf{OT}^{\mathsf{sen}}_{\mathsf{rpke}}$ is a better choice than $\mathsf{OT}_{\mathsf{ospk}}$.

5.2 Comparison with Respect to 1-out-of-n OT

To compare the asymptotic complexity, we consider 1-out-of-n OT. In particular, we use the protocol described in Section 3.3 as our sender-friendly OT protocol. The detailed comparison is shown in Table 2.

1-out-of-n OT	Round Complexity	Sender Communication	Receiver Communication
$\mathsf{OT}_{\mathsf{ospk}}$ ([12])	2 passes	$O(n)$	$O(n)$
$\mathsf{OT}_{\mathsf{osct}}$ ([12])	3 passes	$O(n)$	$O(n)$
$\mathsf{OT}^{\mathsf{sen}}_{\mathsf{rpke}}$ (Section 3)	2 passes	$O(n/\log n)$	$O(n^{1+\varepsilon}/\log n)$
$\mathsf{OT}^{\mathsf{rec}}_{\mathsf{rpke}}$ (Section 4)	3 passes	$O(n)$	$O(1)$

Table 2. A comparison of the PKE-based 1-out-of-n OT protocols of [12] and our OT protocols regarding round complexity, communication costs of sender and the receiver. Note that we focus on the asymptotic cost.

The comparison tells us that for relatively large n (e.g., $n = 1000$), our protocols may be better choices in some settings. For example, if the sender is in a low-speed network and we want the sender to have low communication, then $\mathsf{OT}^{\mathsf{sen}}_{\mathsf{rpke}}$ will be a better choice. Similar, if the receiver is in a low-speed network and we want the receiver to have low communication, then $\mathsf{OT}^{\mathsf{rec}}_{\mathsf{rpke}}$ will be a better choice.

6 Conclusion

This work takes the work of Gertner et al. [12] as the starting point and continue to study the relationship between OT and PKE. Our main result is that rerandomizable PKE implies OT. Since rerandomization can be viewed as the minimum homomorphism in the sense that it only allows one to compute the identity function, our result answers the question of what is the minimum homomorphism required to construct OT. Based on rerandomizable PKE, we give two OT protocols and compare its efficiency with previous PKE-based OT protocols. Our OT protocols have new efficiency features, and they are more suitable for the unbalanced setting where one party may have more data processing power than the other one.

Acknowledgement. We are grateful for the helpful comments from the anonymous reviewers. This work was supported by the National Key Research and Development Program of China (No. 2020YFB1805402) and the National Natural Science Foundation of China (Grants No. 61872359 and No. 61936008).

References

1. Ali, A., Lepoint, T., Patel, S., Raykova, M., Schoppmann, P., Seth, K., Yeo, K.: Communication-computation trade-offs in PIR. In: USENIX Security 2021 (2021), https://www.usenix.org/conference/usenixsecurity21/presentation/ali
2. Angel, S., Chen, H., Laine, K., Setty, S.T.V.: PIR with compressed queries and amortized query processing. In: 2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA. pp. 962–979. IEEE Computer Society (2018). https://doi.org/10.1109/SP.2018.00062
3. Brassard, G., Crépeau, C., Robert, J.: All-or-nothing disclosure of secrets. In: Odlyzko, A.M. (ed.) Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings. Lecture Notes in Computer Science, vol. 263, pp. 234–238. Springer (1986). https://doi.org/10.1007/3-540-47721-7_17
4. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA. pp. 136–145. IEEE Computer Society (2001). https://doi.org/10.1109/SFCS.2001.959888
5. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2729, pp. 565–582. Springer (2003). https://doi.org/10.1007/978-3-540-45146-4_33
6. Chen, H., Huang, Z., Laine, K., Rindal, P.: Labeled PSI from fully homomorphic encryption with malicious security. In: Lie, D., Mannan, M., Backes, M., Wang,

X. (eds.) Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018. pp. 1223–1237. ACM (2018). https://doi.org/10.1145/3243734.3243836

7. Chen, H., Laine, K., Rindal, P.: Fast private set intersection from homomorphic encryption. In: Thuraisingham, B., Evans, D., Malkin, T., Xu, D. (eds.) Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017. pp. 1243–1255. ACM (2017). https://doi.org/10.1145/3133956.3134061
8. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: 36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995. pp. 41–50. IEEE Computer Society (1995). https://doi.org/10.1109/SFCS.1995.492461
9. Cong, K., Moreno, R.C., da Gama, M.B., Dai, W., Iliashenko, I., Laine, K., Rosenberg, M.: Labeled PSI from homomorphic encryption with reduced computation and communication. In: Kim, Y., Kim, J., Vigna, G., Shi, E. (eds.) CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021. pp. 1135–1150. ACM (2021). https://doi.org/10.1145/3460120.3484760
10. Gamal, T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings. Lecture Notes in Computer Science, vol. 196, pp. 10–18. Springer (1984). https://doi.org/10.1007/3-540-39568-7_2
11. Gertner, Y., Ishai, Y., Kushilevitz, E., Malkin, T.: Protecting data privacy in private information retrieval schemes. In: Vitter, J.S. (ed.) Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998. pp. 151–160. ACM (1998). https://doi.org/10.1145/276698.276723
12. Gertner, Y., Kannan, S., Malkin, T., Reingold, O., Viswanathan, M.: The relationship between public key encryption and oblivious transfer. In: 41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA. pp. 325–335. IEEE Computer Society (2000). https://doi.org/10.1109/SFCS.2000.892121
13. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A.V. (ed.) Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA. pp. 218–229. ACM (1987). https://doi.org/10.1145/28395.28420
14. Henzinger, A., Hong, M.M., Corrigan-Gibbs, H., Meiklejohn, S., Vaikuntanathan, V.: One server for the price of two: Simple and fast single-server private information retrieval. IACR Cryptol. ePrint Arch. p. 949 (2022), https://eprint.iacr.org/2022/949
15. Lin, C., Liu, Z., Malkin, T.: XSPIR: efficient symmetrically private information retrieval from ring-lwe. In: Atluri, V., Pietro, R.D., Jensen, C.D., Meng, W. (eds.) Computer Security - ESORICS 2022 - 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26-30, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13554, pp. 217–236. Springer (2022). https://doi.org/10.1007/978-3-031-17140-6_11

16. Melchor, C.A., Barrier, J., Fousse, L., Killijian, M.: XPIR : Private information retrieval for everyone. Proc. Priv. Enhancing Technol. **2016**(2), 155–174 (2016). https://doi.org/10.1515/popets-2016-0010
17. Menon, S.J., Wu, D.J.: SPIRAL: fast, high-rate single-server PIR via FHE composition. In: SP 2022 (2022). https://doi.org/10.1109/SP46214.2022.9833700
18. Mughees, M.H., Chen, H., Ren, L.: Onionpir: Response efficient single-server PIR. In: CCS '21 (2021). https://doi.org/10.1145/3460120.3485381
19. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding. Lecture Notes in Computer Science, vol. 1592, pp. 223–238. Springer (1999). https://doi.org/10.1007/3-540-48910-X_16
20. Park, J., Tibouchi, M.: SHECS-PIR: somewhat homomorphic encryption-based compact and scalable private information retrieval. In: ESORICS 2020 (2020). https://doi.org/10.1007/978-3-030-59013-0_5
21. Prabhakaran, M., Rosulek, M.: Rerandomizable RCCA encryption. In: Menezes, A. (ed.) Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4622, pp. 517–534. Springer (2007). https://doi.org/10.1007/978-3-540-74143-5_29
22. Rabin, M.O.: How to exchange secrets with oblivious transfer (1981)
23. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005. pp. 84–93. ACM (2005). https://doi.org/10.1145/1060590.1060603
24. Stern, J.P.: A new efficient all-or-nothing disclosure of secrets protocol. In: Ohta, K., Pei, D. (eds.) Advances in Cryptology - ASIACRYPT '98, International Conference on the Theory and Applications of Cryptology and Information Security, Beijing, China, October 18-22, 1998, Proceedings. Lecture Notes in Computer Science, vol. 1514, pp. 357–371. Springer (1998). https://doi.org/10.1007/3-540-49649-1_28
25. Yao, A.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982. pp. 160–164. IEEE Computer Society (1982). https://doi.org/10.1109/SFCS.1982.38

A From Bit-OT to String-OT

Our OT protocols are designed for bit-OT where each item is a bit. In this section, we show how to extend our protocols to string-OT where each item is a bitstring. Concretely, we use the idea of [8]. Let $x_1, \ldots, x_n \in \{0,1\}^l$ be the bitstrings held by the sender where each $x_j = (x_{j,1}, \ldots, x_{j,l})$, and let i be the index held by the receiver. The sender first defines $X_k = (x_{1,k}, \ldots, x_{n,k})$ for each $k \in [l]$, then a naive string-OT protocol is that the sender and receiver direct invoke a bit-OT protocol l times, where the sender uses X_k as its input in the k-th invocation. However, the authors in [8] observed that some messages of the receiver may be used for multiples invocations because the receiver has the same input in every invocation, which allows us to reduce the communication cost. For the sake of completeness, we present the detailed descriptions of our PKE-based string-OT protocols in this section. We note that the security proofs of our string-OT protocols will be much like the security proofs of our bit-OT protocols, and we omit the details about the security proofs.

A.1 Sender-Friendly 1-out-of-n String-OT

In this section, we give the description of our sender-friendly string-OT protocol. Similar to our bit-OT protocol, we first give an inefficient string-OT protocol.

Protocol $\mathsf{sOT}^{\mathsf{sen}}_{\mathsf{rpke}}$

Input: The sender P_0 takes n l-bit long bitstrings $x_1, \ldots, x_n$ as inputs where each $x_j = (x_{j,1}, \ldots, x_{j,l})$, and the receiver P_1 takes an index $i \in [n]$ as input. Let $I = \{0^n, 1^n\}$.
Output: P_1 gets x_i as output.

1. P_1 samples a pair of keys $(pk, sk) \leftarrow \mathsf{Keygen}(1^\kappa)$. Then, for each $z = (z_1, \ldots, z_n) \in \{0,1\}^n \backslash I$, P_1 computes $c_z \leftarrow \mathsf{Enc}(z_i, pk)$. Finally, P_1 sends $(pk, \{c_z\}_{z \in \{0,1\}^n \backslash I})$ to P_0.
2. P_0 defines $X_k = (x_{1,k}, \ldots, x_{n,k})$ for each $k \in [l]$. Then for each $k \in [l]$, P_0 computes $e_k \leftarrow \mathsf{Enc}(x_{1,k}, pk)$ if $X_k \in I$ and $e_k \leftarrow \mathsf{Rand}(c_{X_k}, pk)$ otherwise. Finally, it sends $\{e_k\}_{k \in [l]}$ to P_1.
3. For each $k \in [l]$, P_1 computes $u_k \leftarrow \mathsf{Dec}(e_k, sk)$. P_1 outputs $(u_1, \ldots, u_l)$.

Complexity of $\mathsf{sOT}^{\mathsf{sen}}_{\mathsf{rpke}}$. The protocol $\mathsf{sOT}^{\mathsf{sen}}_{\mathsf{rpke}}$ requires the sender to send l ciphertexts and the receiver to send $2^n - 2$ ciphertexts (and a public key). The reduction from long OT to short OT described in Section 3.2 also applies to string-OT. By a similar discussion in Section 3.3, we could obtain an efficient string-OT protocol where the costs of the sender and receiver are $O(ln/\log n)$ and $O(n^{1+\varepsilon}/\log n)$ for a positive constant ε, respectively.

A.2 Receiver-Friendly 1-out-of-n String-OT

This section presents the description of our receiver-friendly string-OT protocol.

Protocol $\mathsf{sOT}^{\mathsf{rec}}_{\mathsf{rpke}}$

Input: The sender P_0 takes n l-bit long bitstrings $x_1, \ldots, x_n$ as inputs where each $x_j = (x_{j,1}, \ldots, x_{j,l})$, and the receiver P_1 takes an index $i \in [n]$ as input.
Output: P_1 gets x_i as output.

1. P_0 samples a key pair $(pk, sk) \leftarrow \mathsf{Keygen}(1^\kappa)$. Then, for each $j \in [n]$ and $k \in [l]$, P_0 computes $c^k_{j,0} \leftarrow \mathsf{Enc}(x_j, pk)$ and $c^k_{j,1} \leftarrow \mathsf{Enc}(x_j \oplus 1, pk)$. Finally, P_0 sends $(pk, \{c^k_{j,0}, c^k_{j,1}\}_{j\in[n],k\in[l]})$ to P_1.
2. For each $k \in [l]$, P_1 chooses a random bit r_k, and computes $e_k \leftarrow \mathsf{Rand}(c^k_{i,0}, pk)$ if $r_k = 0$ and $e_k \leftarrow \mathsf{Rand}(c^k_{i,1}, pk)$ otherwise. Then, it sends $\{e_k\}_{k\in[l]}$ to P_0.
3. P_0 computes $u_k \leftarrow \mathsf{Dec}(e_k, sk)$ for each $k \in [l]$ and sends $\{u_k\}_{k\in[l]}$ to P_1.
4. P_1 computes $z_k = u_k \oplus r_k$ for each $k \in [l]$ and outputs $(z_1, \ldots, z_k)$.

Complexity of $\mathsf{sOT}^{\mathsf{rec}}_{\mathsf{rpke}}$. The protocol $\mathsf{sOT}^{\mathsf{rec}}_{\mathsf{rpke}}$ requires the sender to send $2ln$ ciphertexts and l plaintexts (and a public key) and the receiver to send l ciphertexts. Namely, the costs of the sender and receiver are $O(ln)$ and $O(l)$, respectively.