Blazing Fast OT for Three-Round UC OT Extension

Ran Canetti[1], Pratik Sarkar[1], and Xiao Wang[2]

[1] Boston University, Boston, USA
{canetti,pratik93}@bu.edu
[2] Northwestern University, Evanston, USA
wangxiao@cs.northwestern.edu

Abstract. Oblivious Transfer (OT) is an important building block for multi-party computation (MPC). Since OT requires expensive public-key operations, efficiency-conscious MPC protocols use an OT extension (OTE) mechanism [Beaver 96, Ishai et al. 03] to provide the functionality of *many independent* OT instances with the same sender and receiver, using only symmetric-key operations plus *few* instances of some *base OT* protocol. Consequently there is significant interest in constructing *OTE friendly* protocols, namely protocols that, when used as base-OT for OTE, result in extended OT that are both round-efficient and cost-efficient. We present the most efficient OTE-friendly protocol to date. Specifically:

- Our base protocol incurs only 3 exponentiations per instance.
- Our base protocol results in a 3 round extended OT protocol.
- The extended protocol is UC secure in the Observable Random Oracle Model (ROM) under the CDH assumption.

For comparison, the state of the art for base OTs that result in 3-round OTE are proven only in the programmable ROM, and require 4 exponentiations under Interactive DDH or 6 exponentiations under DDH [Masney-Rindal 19]. We also implement our protocol and benchmark it against the Simplest OT protocol [Chou and Orlandi, Latincrypt 2015], which is the most efficient and widely used OT protocol but not known to suffice for OTE. The computation cost is roughly the same in both cases. Interestingly, our base OT is also 3 rounds. However, we slightly modify the extension mechanism (which normally adds a round) so as to preserve the number of rounds in our case.

1 Introduction

Oblivious Transfer (OT) is a fundamental primitive for multi-party computation (MPC). It has been shown to be complete [GMW87, Kil88] and has become the most widely used building block in both the two-party setting [Yao86, NNOB12] and the multi-party setting [BLO16, WRK17, HSS17]. However, oblivious transfer is expensive since it requires public key operations [IR89]. This limitation is mitigated by the seminal concept of OT extension [Bea96, IKNP03], which allows the parties to compute $m = \text{poly}(\kappa)$ number of OTs using only κ "base

OTs" and $O(m)$ symmetric-key operations, where κ is the computational security parameter. This yields a large number of OTs at the cost of $\mathcal{O}(1)$ symmetric key operations.

The state-of-the-art protocol for malicious OT extension [KOS15] can compute more than ten million OTs per second in a high bandwidth network setting. As such, it appears that the problem of constructing efficient OT extension has been resolved. However, some challenges remain. First, we note that the cost of the base OTs remains a significant consideration when m is only moderately larger than κ and security against all-but-one corruption is needed. For instance, Wang et al. [WRK17] reported that in their implemention of a malicious 128-party computation tolerating 127-party corruption in the WAN setting, it takes about 140 seconds to securely evaluate an AES circuit, where 80 seconds (more than 55% of the total cost!) are spent on computing base OTs.

Another challenge is the number of rounds. Ideally, we would like to obtain extended OT with only two rounds. However, here we have only two known solutions: The original OT extension pf Beaver [Bea96] which is highly inefficient due to non-black-box use of the underlying symmetric-key primitives, and the Boyle et al. [BCG+19] two-round OT extension, based on the Learning Parity with Noise (LPN) assumption, whose performance is better than IKNP-like OT extension only when the network bandwidth is low (≈100 Mbps).

The other approach taken in the literature is to apply a black-box OT extension (such as that of [KOS15]) to some base OT. This method, however results in an additional round. In fact, recent result by Garg et al. [GMMM18] shows that this is inevitable, namely $(n + 1)$ rounds for OT extension are necessary if an n-round base OT is used. Thus, this approach seems to result in extended OT protocols with three or more rounds. Furthermore, the state-of-the-art two-round OT protocols are much slower than the best three-round OT protocols. For example, the two-round OT by Peikert et al. [PVW08] requires 11 exponentiations. More recently, [MR19] proposed an OT that requires 6 exponentiations under standard DDH assumption or 4 exponentiations under non-standard IDDH assumption. This means that even three-round extended OT protocols, obtained in this way, are less than optimally efficient.

Another set of challenges revolves around the level of security obtained and the assumptions used. Chou and Orlandi [CO15] proposed a base-OT protocol with malicious security (dubbed as CO-OT). The work of [HL17] proposed a similar protocol. However, it has been shown [BPRS17, GIR17, LM18] that this protocol and [HL17] cannot be proven secure with simulation-based security because a simulator cannot extract a corrupt receiver's choice bit. There have been some works [BPRS17, DKLs18] trying to fix this issue, but all of them require either much more computation or higher round complexity. Masny and Rindal [MR19] recently proposed a UC-secure OT in the programmable random oracle model (ROM). Their performance is slightly worse than CO-OT under non-standard notion of interactive version of the Decisional Diffie Hellman (IDDH) assumption and much worse under Decisional Diffie Hellman (DDH) assumption.

Protocol	#Exponentiations per base OT	#Rounds of OT Extension	Computational Assumption	Trusted Setup
[PVW08]	11	3	DDH	CRS
[CO15][1]	3	4	CDH	PRO
[BPRS17]	11	3	DDH	PRO
[HL17][1]	5	4	CDH	PRO
[DKLs18][2]	3	6	CDH	ORO
[MR19]	4	3	IDDH	PRO
[MR19]	6	3	DDH	PRO
This work	3	3	CDH	ORO

Table 1. Comparison to related protocols. "#Rounds of OT extension" is the round complexity of the best OT extension with selected base OT protocol. IDDH refers to interactive DDH, not known to be reducible to DDH. PRO refers to programmable RO; ORO refers to observable RO.
[1]Do not provide simulation-based security.
[2]Incur a one-time computation cost of one NIZKPoK.

1.1 Our Contributions

In this paper, we construct an OT protocol tailored to be base OT for the [KOS15] OT extension. Our protocol is highly efficient, and results in a 3 round extended OT that is UC secure in the observable ROM assuming only CDH. See Table 1 for comparison with the state of the art.

The key idea underlying our construction is to design a three-round base OT protocol that circumvents the lower bound proved by Garg et al. [GMMM18]. This is achieved by considering a slight modification of the KOS extension, that is specific to our base OT protocol: The parties use the inputs for the base OT protocol to compute the OT extension messages in parallel to the execution of the base-OT computation. This yields a round-preserving three-round OT extension protocol. To preserve efficiency, we only use some specific property of the base OT protocol (and thus non-black-box to base OT), but avoid non-black-box use of any underlying primitives or computational assumptions. We observe that our protocol is compatible with OT extension protocols [ALSZ15, PSS17, OOS17] in the IKNP domain. It also works for state-of-the-art 1-out-of-N OT extension protocols [PSS17, OOS17]. See Sec. 3 for more discussion. Our detailed contributions are listed as follows:

- **Weaker base-OT Functionality.** To securely realize OT extension efficiently, we consider a UC-secure base-OT functionality that allows selective failure attack by a corrupt sender. We further relax the UC-security requirement to only sender-sided simulation-based security; on the receiver-side, we demonstrate that indistinguishability based security suffices for KOS, provided the receiver's input can be extracted.
- **Weaker Assumptions.** Our protocol is secure assuming CDH in the observable random oracle (ORO) model. Our assumptions and trusted setup

are weaker and far more well-studied than other protocols with comparable efficiency. When used in the OT extension, the OT extension protocol becomes UC-secure.
- **Best efficiency.** Our protocol requires three exponentiations per OT and is as efficient as the CO-OT [CO15]. This is also experimentally verified based on implementation. Since CO-OT (which is insecure) is the most efficient among all existing OT protocols, our new OT with provable security is also the most efficient.
- **Round Preserving.** Our OT protocol requires three rounds, one round more than necessary; however, one unique feature of our protocol is that its *last two rounds of messages* can be securely sent in parallel with the OT-extension messages and thus resulting in a three-round OT extension protocol.
- **Empirical Comparison.** Finally, we implement our protocol and demonstrate its high performance. In detail, our protocol is as efficient as the OT by Chou and Orlandi (which cannot be proven UC secure). When used in the OT extension, our protocol results in a even better performance due to reduced round complexity.

We note that the original KOS paper had an interactive coin tossing subprotocol. It resulted in a 5 rounds protocol and it relied on Correlation Robust Function (CRF). The subprotocol was made non-interactive by the work of [DKLs18] using the Fiat-Shamir transform by relying on a non-programmable random oracle. This reduced the round complexity to 3. We consider this round optimized variant of KOS in the RO model since we already require the RO for our OT protocol.

1.2 More Discussion on Related Works

Here we highlight the protocols from prior OT literature that are relevant to our work. A comparison can be found in Table 1.

- The two-round UC-secure OT protocol by [PVW08] is a candidate for the base OTs in KOS. Its optimized variant computes 11 exponentiations and is proven secure in the common reference string model under DDH assumption.
- The Simplest OT (or CO-OT) was proposed by Chou and Orlandi [CO15]. It computes 3 exponentiations in the programmable random oracle (PRO) model assuming CDH. It requires 2 rounds to compute a random OT, but their OT messages cannot be parallelized with the OT extension messages, thus resulting in a 4 round OT extension. It has been shown [LM18] that this protocol cannot be proven secure with simulation-based security because a simulator cannot extract a corrupt receiver's choice bit. For proving UC-security of the OT extension protocol, the inputs of the receiver (of the base OT) has to be extracted.
- The work of [DKLs18] proposed a 5 round OT protocol, with selective failure, for the base OTs in the ORO model. They compute 3 exponentiations per

OT and incur a one-time computation of a non-interactive zero-knowledge proof of knowledge (NIZKPoK) for Discrete Log. The high round complexity of the base OTs leads to a 6 round OT extension since their last OT message cannot be parallelized with the last message of the OT extension.
– A recent work by [MR19] proposed non-interactive OTs from non-interactive Key Exchange. The resulting OT extension would still require 3 rounds. Their optimized variant requires 4 exponentiations under IDDH assumption and their unoptimized variant requires 6 exponentiations in the PRO model. However, IDDH is not known to be reducible to the standard DDH assumption.
– Silent OT extension [BCG+19] does not follow the IKNP-style extension. Instead, it can be viewed as a special case of vector OLE [BCGI18] and requires an LPN assumption. The resulting protocol can be more efficient than KOS under low network bandwidth.

Roadmap. In the next section, we introduce some notations and important concepts used in this paper. In Sec. 3, we present the key intuitions behind our protocols. This is followed by our weakened OT functionality in Sec. 4. Then, we show that our weakened OT functionality suffices to obtain the KOS OT extension in Sec. 5. We instantiate κ instances of our OT functionality in Sec. 6. Finally, we present our implementation details and compare it with the CO-OT in Sec. 7.

2 Preliminaries

Notations. We denote by $a \leftarrow D$ a uniform sampling of an element a from a distribution D. The set of elements $\{1, \ldots, n\}$ is represented by $[n]$. A function $\mathsf{neg}(()\cdot)$ is said to be negligible, if for every polynomial $p(\cdot)$, there exists a constant c, such that for all $n > c$, it holds that $\mathsf{neg}(()n) < \frac{1}{p(n)}$. We denote a probabilistic polynomial time algorithm as PPT. We denote the computational security parameter by κ and statistical security parameter by μ respectively. Let $\mathbb{Z}_q$ denote the field of order q, where $q = \frac{p-1}{2}$ and p are primes. Let $\mathbb{G}$ be the multiplicative group corresponding to $\mathbb{Z}_p^*$ with generator g, where CDH assumption holds. We denote a field of size 2^κ as $\mathbb{F}$. Our security proofs are in the Universal Composability (UC) framework of [Can01]. We refer to the original paper for details. For a bit $b \in \{0, 1\}$, we denote $1 - b$ by $\bar{b}$. We denote a matrix as $\mathbf{M}$ where $\mathbf{M}^i$ refers to the ith column and $\mathbf{M}_j$ as the jth row of $\mathbf{M}$ respectively. Given a field element $x \in \mathbb{F}$ and a bit vector $\mathbf{a} = (a_1, a_2, \ldots, a_\kappa)$ we denote component-wise multiplication as $x \cdot \mathbf{a} = (a_1 \cdot x, a_2 \cdot x, \ldots, a_\kappa \cdot x)$. In our OT extension protocol, the sender is denoted as $\mathsf{S}_{\mathsf{Ext}}$ and the receiver is denoted as $\mathsf{R}_{\mathsf{Ext}}$ respectively.

Random Oracle Model. A random oracle (RO) functionality is parametrized by a domain and a range and it is as $\mathcal{F}_{\mathsf{RO}}$ in Fig. 2. A random oracle query on message m is denoted by $\mathcal{F}_{\mathsf{RO}}(m)$. The random oracle functionality can be

Fig. 1. The ideal functionality $\mathcal{F}_{\mathsf{OT}}$ for Oblivious Transfer

$\mathcal{F}_{\mathsf{OT}}$

$\mathcal{F}_{\mathsf{OT}}$ interacts with a sender S and a receiver R:

- On input $(\textsc{Choose}, \mathsf{rec}, \mathsf{sid}, b)$ from R where $b \in \{0,1\}$; if no message of the form $(\mathsf{rec}, \mathsf{sid}, b)$ has been recorded in the memory, store $(\mathsf{rec}, \mathsf{sid}, b)$ and send $(\mathsf{rec}, \mathsf{sid})$ to S.
- On input $(\textsc{Transfer}, \mathsf{sen}, \mathsf{sid}, (a_0, a_1))$ from S with $a_0, a_1 \in \{0,1\}^n$, if no message of the form $(\mathsf{sen}, \mathsf{sid}, (a_0, a_1))$ is recorded and a message of the form $(\mathsf{rec}, \mathsf{sid}, b)$ is stored, send $(\mathsf{sent}, \mathsf{sid}, a_b)$ to R and $(\mathsf{sent}, \mathsf{sid})$ to S. Ignore future messages with the same sid.

broadly classified [CDG+18] into three categories based on its features- plain RO, observable RO and programmable RO. A plain RO returns a random string, from its range, upon being queried on a message m, from its domain. An observable RO inherits the properties of the plain RO but in addition it grants the simulator to observe the queries made, to $\mathcal{F}_{\mathsf{RO}}$, by the adversary. Our proofs hold in the global RO (GRO) model of [CJS14] where the observable RO is replaced by the GRO.

Tweakable Correlation Robust Hash. OT extension requires a correlation robust hash function. We adopted the definition proposed by Guo et al. [GKW+19], where a tweak is explicitly included in the hash function too. Given a function $\mathrm{CRF} : \mathcal{T} \times \{0,1\}^\kappa \to \{0,1\}^\kappa$, define $\mathcal{O}_\Delta(t, w) \stackrel{\text{def}}{=} \mathrm{CRF}(t, w \oplus \Delta)$, where $t \in \mathcal{T}$. Let Func denote the set of functions from $\mathcal{T} \times \{0,1\}^\kappa$ to $\{0,1\}^\kappa$.

Definition 1. *Given a function* $\mathrm{CRF} : \mathcal{T} \times \{0,1\}^\kappa \to \{0,1\}^\kappa$*, a uniform distribution on* $\{0,1\}^\kappa$ *namely* U_κ*, we say that* CRF *is tweakable correlation robust if for any PPT distinguisher* D*, if*

$$\left| \Pr_{\Delta \leftarrow U_\kappa} \left[D^{\Delta(\cdot)} = 1 \right] - \Pr_{f \leftarrow \mathsf{Func}} \left[D^{f(\cdot)} = 1 \right] \right| = \mathsf{negli}(\kappa).$$

Note that in our use of tweakable correlation robust hash, $\mathcal{T}$ is a tuple of values, one for sid and one for index i.

Oblivious Transfer. In a 1-out-of-2 OT, we have a sender (S) holding two inputs $a_0, a_1 \in \{0,1\}^n$ and a receiver (R) holding a choice bit b. The correctness of OT means that R will obtain a_b as the outcome of the protocol. At the same time, S should learn nothing about b, and R should learn nothing about the other input of S, namely $a_{\bar{b}}$. The ideal OT functionality $\mathcal{F}_{\mathsf{OT}}$ is shown in Figure 1.

3 Technical Overview

In this section we give an overview of our technical contributions. First, we recall the KOS OT extension from a high-level. We argue that the base OTs in KOS

Fig. 2. The ideal functionality $\mathcal{F}_{\mathsf{OT}}$ for Random Oracle

$\mathcal{F}_{\mathsf{RO}}$

$\mathcal{F}_{\mathsf{RO}}$ is parameterized by a domain D and range R and it proceeds as follows, running on security parameter k:

- $\mathcal{F}_{\mathsf{RO}}$ maintains a list L (which is initially empty) of pairs of values $(\hat{m}, \hat{h})$, s.t. $\hat{m} \in D$ and $\hat{h} \in R$.
- Upon receiving a value (sid, m) (where $m \in D$) perform the following: If there is a pair $(m, \hat{h})$, for some $\hat{h} \in R$, in the list L, set $h := \hat{h}$. If there is no such pair, sample $h \leftarrow_R R$ and store the pair (m, h) in L. Once h is set, reply to the activating machine with (sid, h).

do not require UC security. Building on this idea, we propose a weaker OT functionality and then we provide an efficient three-round OT protocol which would yield a three round OT extension.

3.1 Overview of KOS

In the KOS OT extension, the sender $\mathsf{S}_{\mathsf{Ext}}$ and receiver $\mathsf{R}_{\mathsf{Ext}}$ wants to generate m OTs using κ invocations to $\mathcal{F}_{\mathsf{OT}}$ (base OTs) and symmetric key operations. The sender $\mathsf{S}_{\mathsf{Ext}}$ plays the role of a receiver in the base OTs. He samples a random κ bit string $\mathbf{s}$ and invokes ith instance of $\mathcal{F}_{\mathsf{OT}}$ with ith bit of $\mathbf{s}$ for $i \in [\kappa]$. The receiver $\mathsf{R}_{\mathsf{Ext}}$ invokes $\mathcal{F}_{\mathsf{OT}}$ as sender with random pads $(\mathbf{k}_{i,0}, \mathbf{k}_{i,1})$. $\mathsf{S}_{\mathsf{Ext}}$ obtains $\mathbf{k}_{i,s_i}$ from the ith base OT. In addition, $\mathsf{R}_{\mathsf{Ext}}$ also sends a mapping $\mathbf{D}$ from his inputs to the $(\mathbf{k}_{i,0}, \mathbf{k}_{i,1})$ values. Upon obtaining this mapping $\mathbf{D}$ and the base-OT output, the sender computes his mapping $\mathbf{Q}$. He computes correlated pads for the extended OTs using $\mathbf{s}$ and $\mathbf{Q}$ as $\mathrm{CRF}(\mathsf{sid}, j, \mathbf{Q}_j)$ and $\mathrm{CRF}(\mathsf{sid}, j, \mathbf{Q}_j \oplus \mathbf{s})$ for $j \in [m]$, where CRF is a correlation robust function. If the receiver's input bit for j-th extended OT is 0, then he can compute $\mathbf{Q}_j$, else he can compute $\mathbf{Q}_j \oplus \mathbf{s}$. The other value remains hidden due to $\mathbf{s}$ and security of CRF. Using the correlated pads, $\mathsf{S}_{\mathsf{Ext}}$ encrypts his inputs for the extended OTs and sends it to $\mathsf{R}_{\mathsf{Ext}}$.

In addition, $\mathsf{S}_{\mathsf{Ext}}$ also performs a consistency check on matrix $\mathbf{D}$ is correctly formed by $\mathsf{R}_{\mathsf{Ext}}$, else a malformed $\mathbf{D}$ matrix would leak the bits of $\mathbf{s}$ rendering the protocol insecure. The original KOS paper had an interactive check phase. It was made non-interactive by the work of [DKLs18] using the Fiat-Shamir transform by relying on the observable random oracle. Our protocol also uses the same non-interactive check to obtain a 3-round OT extension protocol where the checks are run in the second OT extension message. The base OTs are run for 3 rounds and the last message of the OT extension is sent in parallel to the last message of the base OT. Next, we discuss our proposed relaxations in the base OT functionality.

3.2 Relaxation in the OT functionality

Firstly, it can be observed that the parties invoke the base OTs in KOS with random OTs. So, one can consider random OT functionality instead of full OT functionality. Next, we can allow selective failure in the base OTs. The work of KOS and [DKLs18] showed that the base OTs do not require full UC-security of an OT functionality. Instead, the functionality can allow a corrupt sender (i.e. $\mathsf{R}_{\mathsf{Ext}}$ in the OT extension) to launch a selective failure attack on the $\mathbf{s}$ bits of the receiver (i.e. $\mathsf{S}_{\mathsf{Ext}}$ in the OT extension). In such a case, the corrupt $\mathsf{R}^*_{\mathsf{Ext}}$ will still have a negligible advantage in breaking the security of the extended OTs. We claim that the OT functionality can be further relaxed based on the following observations in the KOS protocol.

1. **Delayed input extraction of receiver:** The inputs of the receiver ($\mathsf{S}_{\mathsf{Ext}}$) in the base OTs can be extracted after $\mathsf{S}_{\mathsf{Ext}}$ sends the last message of the OT Extension protocol. Recall, that the last message of the OT Extension protocol consists of the inputs of $\mathsf{S}_{\mathsf{Ext}}$ encrypted with the correlated pads. The simulator against a corrupt $\mathsf{S}^*_{\mathsf{Ext}}$ can simulate the second message of the OT extension without the knowledge of $\mathbf{s}$. Later, it can extract $\mathbf{s}$ from the base OTs and then extract $\mathsf{S}_{\mathsf{Ext}}$'s inputs from the last message.
2. **Corrupt receiver can abort after base OT:** A corrupt receiver ($\mathsf{S}_{\mathsf{Ext}}$) can abort after obtaining the results of base-OT protocol corresponding to $\mathbf{s}$. In such a case, the honest $\mathsf{R}_{\mathsf{Ext}}$ would just abort the protocol as the base OTs resulted in an abort. For each base OT, one of the input of $\mathsf{R}_{\mathsf{Ext}}$ remain hidden from $\mathsf{S}_{\mathsf{Ext}}$ due to the security of the OT; thus hiding $\mathsf{R}_{\mathsf{Ext}}$'s inputs.
3. **Batch of κ OT:** The OT extension protocol requires κ base OTs between the parties. So, the base OTs can be computed in a batch of κ OTs instead of κ independent instances of the OT protocol.

Based on the above observations we can consider the following relaxations to the OT functionality for a corrupt receiver.

1. **Indistinguishability based security against corrupt receiver with input extraction:** We can reduce the simulation based security for a corrupt receiver to an indistinguishability based security. We need an extractor algorithm Ext that can extract the input bit b of a corrupt receiver, given blackbox access to it. The corrupt receiver cannot distinguish its real world view from a view constructed with the sender's message corresponding to bit $\bar{b}$ set as 0^κ.
2. **Corrupt receiver can abort without input extraction:** A corrupt receiver can decide to abort in the OT functionality and in such a case the Ext does not need to extract his inputs.
3. **Corrupt receiver cannot compute both sender messages:** A corrupt receiver cannot compute both sender input messages from the OT transcript and his internal randomness, even if he aborts the protocol.

3.3 Usage in KOS OT Extension

The above relaxations in the base-OT functionality are justified since we do not require simulation based security in the OT extension protocol for the base OTs. This is because the base OTs are internal to the protocol; hence the input/output of the honest parties in the base OTs are inaccessible to the environment $\mathcal{Z}$ who tries to distinguish between real and ideal world executions of the OT extension protocol. Indistinguishability based security suffices for a corrupt receiver (i.e. a corrupt $\mathsf{S}_{\mathsf{Ext}}$) in the base OTs, if we are guaranteed that the following conditions hold :

1. If the base OTs succeed then the input $\mathbf{s}$ of the receiver (i.e. $\mathsf{S}_{\mathsf{Ext}}$) can be extracted after obtaining the OT extension last message as that is used by the simulator (for a corrupt $\mathsf{S}^*_{\mathsf{Ext}}$) to extract the input messages of $\mathsf{S}^*_{\mathsf{Ext}}$. This is guaranteed by the correctness of Ext algorithm when the base OT protocol succeeds.
2. In case the base OT aborts, then the Ext does not need to extract the inputs of $\mathsf{S}^*_{\mathsf{Ext}}$ since the OT extension protocol terminates with an abort too.
3. The corrupt $\mathsf{S}^*_{\mathsf{Ext}}$ should not be able to distinguish between the real world interaction with honest $\mathsf{R}_{\mathsf{Ext}}$, and ideal world interaction with the simulator. In the ideal world, the simulator runs with input for the extended OTs as all 0s string. $\mathsf{S}^*_{\mathsf{Ext}}$, playing the role of receiver in the base OT, cannot compute both sender messages of the base OT. Based on this property, we tweak our OT extension protocol by relying on the random oracle. The tweak ensures that one of the sender's messages in the each base OT will be hidden from $\mathsf{S}^*_{\mathsf{Ext}}$. Thus, he cannot distinguish the real world from the ideal world by relying on the security of the original KOS protocol. Our tweak incurs a minimal overhead of 2 RO queries for each base OT.

We also utilize the fact that the base OTs are computed in a batch of κ OTs. This allows us to efficiently implement a batch of κ instances of the above (weakened) OT functionality based on observable random oracle. Next, we discuss our OT protocol which implements a batch of κ instances of weak OT functionality, as discussed above.

3.4 Optimized OT Protocol in the Observable RO model

We consider an OT protocol in the observable random oracle model where the receiver R generates receiver OT parameter T by invoking a random oracle $\mathcal{F}_{\mathsf{RO1}}$ on a seed. He samples a random $\alpha \leftarrow \mathbb{Z}_q$ and computes his first message based on his input bit b as B, where

$$B = g^{\alpha} \cdot T^{b}$$

He sends B and seed to the sender S. The sender computes T from seed and samples a random $r \leftarrow \mathbb{Z}_q$. S computes sender OT parameters- $z = g^r$ and sends

z to R. S computes his random pads p_0 and p_1 by invoking a random oracle $\mathcal{F}_{\mathsf{RO2}}$ as follows:

$$p_0 = \mathcal{F}_{\mathsf{RO2}}(\mathsf{sid}, B^r)$$
$$p_1 = \mathcal{F}_{\mathsf{RO2}}(\mathsf{sid}, (\tfrac{B}{T})^r)$$

Upon obtaining z, R computes his output pad p_b as follows:

$$p_b = \mathcal{F}_{\mathsf{RO2}}(\mathsf{sid}, z^\alpha) = \mathcal{F}_{\mathsf{RO2}}(\mathsf{sid}, g^{r\alpha})$$

This protocol ensures that a corrupt receiver R^* cannot compute both random pads as that would require invoking $\mathcal{F}_{\mathsf{RO2}}$ on B^r and $(\frac{B}{T})^r = \frac{B^r}{T^r}$. Such a corrupt receiver could be used to break the CDH assumption where $(T, z) = (g^t, g^r)$ is the CDH challenge. The queries made by R^* to $\mathcal{F}_{\mathsf{RO2}}$ can be used to obtain T^r which is the answer to the CDH challenge. This OT protocol also perfectly hides the input b of an honest receiver from a corrupt receiver as α and $\alpha - t$ are valid receiver randomness for $b = 0$ and $b = 1$ respectively, where $B = g^\alpha$. However, this protocol doesn't allow extraction of receiver or sender's inputs from the OT messages in the observable RO model.

Adding Receiver Input Extraction. To add receiver's input extraction the sender adds a challenge in the second round of the OT protocol.

$$\mathsf{chall} = \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_0) \oplus \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_1).$$

The receiver has to respond to the challenge by computing his answer $\mathtt{Ans}$.

$$\mathtt{Ans} = \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_b) \oplus (b \cdot \mathsf{chall}) = \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_0).$$

R has to query $\mathcal{F}_{\mathsf{RO3}}$ to compute $\mathtt{Ans}$ and assists a simulator to extract a corrupt receiver's input. R sends $\mathtt{Ans}$ to the sender in a third OT message. This increases the round complexity to 3 but it ensures that the simulator can extract the receiver's input bit from the RO queries of $\mathcal{F}_{\mathsf{RO3}}$ if $\mathtt{Ans}$ is valid. This is similar to the challenge-response paradigm introduced in the work of [DKLs18]. However, a corrupt sender can compute the challenge in a malicious way such that he can find out the bits of receiver from the response and the receiver fails to identify such an attack. The work of [DKLs18] tackles this issue by making the sender open his randomness (for computing the challenge) to the receiver in a fourth OT message. We ensure correctness of the challenge by making the sender send a proof γ along with the challenge in the second OT message,

$$\gamma = \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_0)) = \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, \mathtt{Ans}).$$

After obtaining the second OT message, the receiver can compute $\mathtt{Ans}$ and verify the proof γ. If γ is valid then he sends $\mathtt{Ans}$ to sender else he aborts. This ensures input extraction of receiver as argued before but it adds selective failure attack by a corrupt sender S^*. S^* can try to guess the bits of receiver and based on that he can maliciously construct the challenge. However, our OT functionality accommodates selective failure attack over a batch of κ OTs and hence this challenge-prove-response paradigm would work in our case. Next, we show that this trick already provides extraction of a corrupt sender's input.

Adding Sender Input Extraction. Our challenge-prove-response paradigm allows us to construct a protocol where the sender's inputs can be extracted if a batch of $\ell > \mu$ OTs are run together. In KOS, κ OTs are run where $\kappa > \mu$. Now, we will explain the reason behind our assumption of $\ell > \mu$. In a batch of ℓ OTs, every OT uses the same T and $z = g^r$, i.e. the ith OT pad is of the form :

$$p_{i,0} = \mathcal{F}_{\mathsf{RO2}}(\mathsf{sid}, B_i^r),$$

$$p_{i,1} = \mathcal{F}_{\mathsf{RO2}}(\mathsf{sid}, (\tfrac{B_i}{T})^r),$$

where B_i is chosen by R based on his randomness α_i and input bit b_i for the i-th OT. In such a case, the sender sends a unique challenge chall_i for each OT as follows:

$$\mathsf{chall}_i = \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{i,0}) \oplus \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{i,1}).$$

Now, the receiver computes the answer resp to the challenge as:

$$\mathsf{resp}_i = \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_b) \oplus (b \cdot \mathsf{chall}) = \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_0).$$

The answer to the challenge is optimized to sending one string for the whole batch instead of κ strings, as follows:

$$\begin{aligned}\mathtt{Ans} &= \mathcal{F}_{\mathsf{RO4}}(\mathsf{sid}, \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, \mathsf{resp}_1), \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, \mathsf{resp}_2), \ldots \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, \mathsf{resp}_\kappa)) \\ &= \mathcal{F}_{\mathsf{RO4}}(\mathsf{sid}, \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{1,0}), \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{2,0}), \ldots \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{\kappa,0})).\end{aligned}$$

The proof sent by the sender is also modified accordingly as:

$$\begin{aligned}\gamma &= \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, \mathcal{F}_{\mathsf{RO4}}(\mathsf{sid}, \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{1,0}), \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{2,0}), \ldots \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{\kappa,0}))) \\ &= \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, \mathtt{Ans}).\end{aligned}$$

The receiver can check his computed answer with the proof and then respond with $\mathtt{Ans}$. This tweak allows us to extract a corrupt sender's input for $\ell > \mu$ OTs. The simulator can extract T^r by observing the queries- B_i^r and $(\frac{B_i}{T})^r$ made by S to $\mathcal{F}_{\mathsf{RO2}}$. Sender needs to query $\mathcal{F}_{\mathsf{RO2}}$ for computing $p_{i,0}$ and $p_{i,1}$ values, which are in turn used to compute the challenge and proof for $\ell > \mu$ OTs. Sender can avoid querying $\mathcal{F}_{\mathsf{RO2}}$ with both- B_i^r and $(\frac{B_i}{T})^r$. In that case, he has to either guess the corresponding RO query results, i.e. $p_{i,0}$ or $p_{i,1}$, or he launches a selective failure attack for every OT and he has to correctly guess receiver's input bit for every OT. This is because, the receiver's input is random and he will compute the $\mathtt{Ans}$ and it will not match with the γ sent by the sender, except with $2^{-\mu}$ probability, since γ and chall_i were computed without correctly computing the $p_{i,0}$ or $p_{i,1}$ values, for every $i \in [\ell]$. Thus, the simulator can observe $\mathcal{F}_{\mathsf{RO2}}$, compute the candidate $p_{i,0}, p_{i,1}$, match with chall_i values and γ and extract the correct T^r. Using T^r, he can extract the sender's input messages.

3.5 Circumventing the impossibility result of [GMMM18]

We circumvent the impossibility result of [GMMM18] by allowing the OT extension receiver $\mathsf{R}_{\mathsf{Ext}}$, i.e. base OT sender, to use the base OT output messages

to send his mapping $\mathbf{D}$ in the second round of the OT extension protocol. If the base OT receiver fails to answer the base OT challenge then $\mathsf{R}_{\mathsf{Ext}}$ aborts else he computes his OT extension output message. The base OT security ensures that a corrupt base OT receiver (i.e. $\mathsf{S}^*_{\mathsf{Ext}}$) cannot compute both messages of the base OT sender (i.e. $\mathsf{R}_{\mathsf{Ext}}$). This hides the input of $\mathsf{R}_{\mathsf{Ext}}$ in $\mathbf{D}$ even though we use the base OT messages before it has terminated. Such non-blackbox usage of the 3 round base-OT protocol allows us to obtain a 3 round OT extension protocol.

4 Weakening the Oblivious Transfer Functionality

In this section, we discuss the type of security that we require from the base OT protocols of the KOS OT extension. We demonstrate that by gradually relaxing the $\mathcal{F}_{\mathsf{OT}}$ functionality, where the parties choose their own input, to a random OT where the functionality provides random inputs to the parties. Next, we allow selective failure attack by a sender on receiver's inputs and define it as $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$. We also allow a corrupt receiver to abort the protocol. We relax the UC-security of this protocol to one-sided simulation. Finally, we formally define our notion of weakened OT which provides simulation based security for a corrupt sender and indistinguishability security against a corrupt receiver.

Random Oblivious Transfer. The OT functionality can be relaxed to consider random inputs, i.e. $\mathcal{F}_{\mathsf{rOT}}$. In this case, the inputs of an honest sender and an honest receiver are chosen randomly by the functionality. However, a malicious sender (also a malicious receiver) can choose his own inputs. The ideal random OT functionality has been presented in Fig. 3.

Random Oblivious Transfer with Selective Failure and Explicit Abort. We can further weaken the $\mathcal{F}_{\mathsf{rOT}}$ functionality to allow selective failure attacks by a corrupt sender. Here, the corrupt sender S^* can try to guess the random input of the receiver by setting its message, corresponding to bit 0 as $\perp$, whereas the message corresponding to bit 1 is set correctly. An honest receiver would abort if his input bit b is 0, else he continues with the protocol. This would leak b to S^*. We also allow a corrupt receiver R^* to explicitly abort the protocol after it obtains its input message a_b. In such a case, the functionality notifies the sender regarding the abort. Our $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ functionality has been modeled in Fig. 4.

Oblivious Transfer for KOS. It was shown in the work of KOS and [DKLs18] that κ instances of $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ (without the ABORT option) suffices to instantiate the κ base OTs in the KOS OT extension protocol. However, this requires simulation based security against a corrupt R^*, where the simulator needs to extract the input of R^* and simulate the sender's message s.t. they open to the correct message, i.e. a_b, even if R^* aborts the protocol. However, such a strong requirement from $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ is an overkill for instantiating the base-OT protocols. We demonstrate that the security against a corrupt receiver can be reduced to indistinguishability based security and his input need not be extracted when he aborts. More precisely:

Fig. 3. The ideal functionality $\mathcal{F}_{\mathsf{rOT}}$ for random Oblivious Transfer

$\mathcal{F}_{\mathsf{rOT}}$

$\mathcal{F}_{\mathsf{OT}}$ interacts with a sender S and a receiver R:

- On input $(\textsc{Choose}, \mathsf{rec}, \mathsf{sid})$ from R; if no message of the form $(\mathsf{rec}, \mathsf{sid}, b)$ has been recorded in the memory, sample $b \leftarrow \{0,1\}$, store $(\mathsf{rec}, \mathsf{sid}, b)$ and send $(\mathsf{rec}, \mathsf{sid})$ to S and (b, sid) to R.
- On input $(\textsc{Choose}^*, \mathsf{rec}, \mathsf{sid}, b)$ from R^* where $b \in \{0,1\}$; if no message of the form $(\mathsf{rec}, \mathsf{sid}, b)$ has been recorded in the memory, store $(\mathsf{rec}, \mathsf{sid}, b)$ and send $(\mathsf{rec}, \mathsf{sid})$ to S and (b, sid) to R.
- On input $(\textsc{Transfer}, \mathsf{sen}, \mathsf{sid})$ from S, if no message of the form $(\mathsf{sen}, \mathsf{sid}, (a_0, a_1))$ is recorded and a message of the form $(\mathsf{rec}, \mathsf{sid}, b)$ is stored, sample $a_0, a_1 \leftarrow \{0,1\}^n$, store $(\mathsf{sen}, \mathsf{sid}, (a_0, a_1))$ in memory and send $(\mathsf{sent}, \mathsf{sid}, (b, a_b))$ to R and $(\mathsf{sent}, \mathsf{sid}, (a_0, a_1))$ to S. Ignore future messages with the same sid.
- On input $(\textsc{Transfer}^*, \mathsf{sen}, \mathsf{sid}, (a_0, a_1))$ from S^* with $a_0, a_1 \in \{0,1\}^n$, if no message of the form $(\mathsf{sen}, \mathsf{sid}, (a_0, a_1))$ is recorded and a message of the form $(\mathsf{rec}, \mathsf{sid}, b)$ is stored, send $(\mathsf{sent}, \mathsf{sid}, (b, a_b))$ to R and $(\mathsf{sent}, \mathsf{sid}, (a_0, a_1))$ to S. Ignore future messages with the same sid.

- **Indistinguishability against a Malicious R^* :** In the KOS OT extension, the invocations to the base OT functionality is internal to the OT extension protocol. The environment $\mathcal{Z}$ does not have access to it; hence it cannot choose inputs for the honest parties in the base OT functionality. This permits us to use an efficient OT protocol to emulate the functionality, s.t. it provides simulation based security against a corrupt S^* whereas for a corrupt R^*, it provides indistinguishability based security. Such a relaxation allows us to use observable random oracle instead of a programmable one in 3 rounds. Previous protocols, like [DKLs18] used observable RO but they required 5 rounds for the base OTs, where the last 2 rounds where spent in simulating the honest sender's messages, i.e. providing simulation based security against corrupt R^*. The work of [MR19] obtain a two-round OT but they require twice the amount of exponentiation as ours by extracting the receiver's input from the first OT message.
- **Input of R^* need not be extracted during Abort :** When the sender $\mathsf{S}^*_{\mathsf{Ext}}$ of the OT extension protocol (acting as the receiver R^* of the base OTs) misbehaves and causes an abort, the OT extension protocol leads to an abort. In such a case, the input messages (in the extended OTs) of $\mathsf{S}^*_{\mathsf{Ext}}$ need not be extracted. Hence, it is not necessary to extract the inputs of $\mathsf{S}^*_{\mathsf{Ext}}$(or R^*) in the base OTs. This allows us to push the extraction of the R^* input until the last round of the base OT (in our case also the last round of OT extension), where it can be extracted when he computes the base OTs correctly. In case he aborts then it is guaranteed that he cannot compute both sender messages. This allows us to save on the number of exponentiations.

Fig. 4. The ideal functionality $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ for Random Oblivious Transfer with Selective Failure

$\mathcal{F}_{\mathsf{SF\text{-}rOT}}$

$\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ interacts with a sender S and receiver R :

- On input $(\textsc{Choose}, \mathsf{rec}, \mathsf{sid})$ from R; if no message of the form $(\mathsf{rec}, \mathsf{sid}, b)$ has been recorded in the memory, sample $b \leftarrow \{0,1\}$, store $(\mathsf{rec}, \mathsf{sid}, b)$ and send $(\mathsf{rec}, \mathsf{sid})$ to S and (b, sid) to R. If a message of the form $(\mathsf{sen}, \mathsf{sid}, (a_0, a_1))$ is stored, send $(\mathsf{sent}, \mathsf{sid}, (b, a_b))$ to R and $(\mathsf{sent}, \mathsf{sid}, (a_0, a_1))$ to S and ignore future messages with the same sid.
- On input $(\textsc{Choose}^*, \mathsf{rec}, \mathsf{sid}, b)$ from R^* where $b \in \{0,1\}$; if no message of the form $(\mathsf{rec}, \mathsf{sid}, b)$ has been recorded in the memory, store $(\mathsf{rec}, \mathsf{sid}, b)$ and send $(\mathsf{rec}, \mathsf{sid})$ to S and (b, sid) to R. If a message of the form $(\mathsf{sen}, \mathsf{sid}, (a_0, a_1))$ is stored, send $(\mathsf{sent}, \mathsf{sid}, (b, a_b))$ to R and $(\mathsf{sent}, \mathsf{sid}, (a_0, a_1))$ to S and ignore future messages with the same sid.
- On input $(\textsc{Guess}^*, \mathsf{sen}, \mathsf{sid}, b')$ from S^*, if $(\mathsf{rec}, \mathsf{sid}, b)$ exists in memory, $b' \in \{0, 1, \perp\}$ and there does not exist $(\mathsf{sen}, \mathsf{sid}, (\textsc{Guess}, \cdot))$ in memory then store $(\mathsf{sen}, \mathsf{sid}, (\textsc{Guess}, b'))$ in memory and perform the following:
  - If $b' = \perp$, do nothing.
  - If $b' = \top$, send $(\textsc{Cheat-Detected}, \mathsf{S})$ to R and $(\textsc{Cheat-Detected})$ to S.
  - If $b' = b$, send $(\textsc{Cheat-Undetected})$ to S.
  - If $b' \neq b$, send $(\textsc{Cheat-Detected}, \mathsf{S})$ to R and $(\textsc{Cheat-Detected})$ to S.
- On input $(\textsc{Transfer}, \mathsf{sen}, \mathsf{sid})$ from S, if no message of the form $(\mathsf{sen}, \mathsf{sid}, (a_0, a_1))$ is stored; sample $a_0, a_1 \leftarrow \{0,1\}^\kappa$, store $(\mathsf{sen}, \mathsf{sid}, (a_0, a_1))$ in memory and send $(\textsc{Received}, \mathsf{sid})$ to R and S. If a message of the form $(\mathsf{rec}, \mathsf{sid}, b)$ is stored, send $(\mathsf{sent}, \mathsf{sid}, (b, a_b))$ to R and $(\mathsf{sent}, \mathsf{sid}, (a_0, a_1))$ to S and ignore future messages with the same sid.
- On input $(\textsc{Transfer}^*, \mathsf{sen}, \mathsf{sid}, (a_0, a_1))$ from S^*, if no message of the form $(\mathsf{sen}, \mathsf{sid}, (a_0, a_1))$ is stored then store $(\mathsf{sen}, \mathsf{sid}, (a_0, a_1))$ in memory and send $(\textsc{Received}, \mathsf{sid})$ to R and S. If a message of the form $(\mathsf{rec}, \mathsf{sid}, b)$ is stored, then send $(\mathsf{sent}, \mathsf{sid}, (b, a_b))$ to R and $(\mathsf{sent}, \mathsf{sid}, (a_0, a_1))$ to S and ignore future messages with the same sid.
- On input $(\textsc{Abort}, \mathsf{rec}, b, \mathsf{sid})$ from R^*, if messages of the form - $(\mathsf{sen}, \mathsf{sid}, (a_0, a_1))$ is stored; send $(\mathsf{sent}, \mathsf{sid}, (b, a_b))$ to R and $(\textsc{Abort}, \mathsf{sid}, (a_0, a_1))$ to S. Ignore future messages with the same sid.

- **Batch of $\kappa > \mu$ OTs :** We consider a batch of $\kappa > \mu$ invocations of OT protocol for the base OTs. This is necessary since a corrupt receiver of OT extension $\mathsf{R}^*_{\mathsf{Ext}}$ can launch a selective failure attack on the inputs of $\mathsf{S}_{\mathsf{Ext}}$ to the base OT. Since the inputs are random, $\mathsf{R}^*_{\mathsf{Ext}}$ can at most determine μ inputs bits of $\mathsf{S}_{\mathsf{Ext}}$'s randomness. However, that gives him a negligible advantage in breaking the security of the OT extension protocol due to the security of the underlying OT extension protocol.

We also assume there exists a PPT algorithm Ext that can extract the input of a malicious R^*, if he does not abort the protocol. We formally define our sender-simulatable $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ with the following security properties required against a corrupt receiver:

Definition 2. *Let $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ be the Oblivious Transfer functionality as shown in Fig. 4. We say that a protocol π_{OT} securely computes $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ with sender-sided simulation with input extractability of receiver if the following holds:*

1. *For every non-uniform PPT adversary S^* controlling the sender in the real model, there exists a non-uniform PPT adversary Sim for the ideal model, such that for any environment $\mathcal{Z}$,*

$$\text{IDEAL}_{\mathcal{F}_{\mathsf{SF\text{-}rOT}},\mathsf{Sim},\mathcal{Z}}((a_0,a_1),b,z)_{z\in\{0,1\}^*} \approx \text{REAL}_{\pi_{OT},\mathsf{S}^*,\mathcal{Z}}((a_0,a_1),b,z)_{z\in\{0,1\}^*}.$$

2. *For every non-uniform PPT adversary $\mathcal{A}$ controlling the receiver R^*, the following holds:*
   - *Property 1: If the sender did not abort, then there exists a PPT extractor algorithm Ext such that the following holds:*

$$\Pr\left[(a_0,a_1) \leftarrow \mathsf{S}^{\mathcal{F}_{\mathsf{SF\text{-}rOT}}}, (b,a_b) \leftarrow \mathcal{A}^{\mathcal{F}_{\mathsf{SF\text{-}rOT}}}, b' \leftarrow \mathsf{Ext}^{\mathcal{A}} : (b \neq b') \wedge (a_0 \neq \perp) \wedge (a_1 \neq \perp)\right] \leq \mathsf{negli}(\kappa)$$

   - *Property 2: If the sender did not abort, then the view of R^* is independent of $a_{\bar{b}}$. More formally the following condition holds:*

$$\mathcal{V}^{\mathsf{S}}_{\pi_{OT},\mathsf{R}^*(z)}((a_0,a_1),b,z)_{z\in\{0,1\}^*} \approx \mathcal{V}^{\mathsf{S}}_{\pi_{OT},\mathsf{R}^*(z)}((\widetilde{a_0},\widetilde{a_1}),b,z)_{z\in\{0,1\}^*},$$

   where Ext outputs b on interacting with R^*. $\mathcal{V}^{\mathsf{S}}_{\pi_{OT},\mathsf{R}^*(z)}$ denotes the view of adversarial R^* after a real execution of protocol π_{OT} with the honest sender S, with random sender inputs (a_0,a_1), and $(\widetilde{a_0},\widetilde{a_1})$ where $\widetilde{a_b} \leftarrow \{0,1\}^\kappa$ and $\widetilde{a_{\bar{b}}} = 0^\kappa$ respectively.
   - *Property 3: R^* cannot compute both sender messages except with negligible probability. More formally the following condition holds:*

$$\Pr\left[(a_0,a_1) \leftarrow \mathsf{S}^{\mathcal{F}_{\mathsf{SF\text{-}rOT}}}, (a_0',a_1') \leftarrow \mathcal{A}^{\mathcal{F}_{\mathsf{SF\text{-}rOT}}} : (a_0 = a_0') \wedge (a_1 = a_1')\right] \leq \mathsf{negli}(\kappa)$$

5 Oblivious Transfer Extension using $\pi^{\kappa}_{\mathsf{OT}}$

In this section we show that the base OTs (a.k.a seed OTs) in the KOS OT extension can be instantiated using κ invocations to the modified (according to Def. 2) $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$. This results in an efficient seed OT phase, where each base OT requires 3 exponentiations. We assume that there exists a protocol $\pi^{\kappa}_{\mathsf{OT}}$ which implements(according to Def. 2) κ instances of $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$. Then we use $\pi^{\kappa}_{\mathsf{OT}}$ to implement the base OTs. Our protocol has been presented in Fig. 5.

Fig. 5. KOS OT Extension using $\pi_{\mathsf{OT}}^{\kappa}$

π_{KOS}

– **Public Inputs:** Group $\mathbb{G}$, fields $\mathbb{Z}_q$ and $\mathbb{F}$, and generator g of group $\mathbb{G}$.
– **Private Inputs:** S has m pairs $\{a_{j,0}, a_{j,1}\}_{j\in[m]}$ of κ bit strings. R has m selection bit vector $\mathbf{r} = (r_1, \cdots, r_m)$ such that each $r_j \in \{0,1\}$.
– **Functionalities:** $\mathsf{PRG} : \{0,1\}^{\kappa} \to \{0,1\}^{m+\kappa}$, $\mathcal{F}_{\mathsf{RO1}} : \{0,1\}^{\kappa} \times \{0,1\}^{\kappa} \to \{0,1\}^{\kappa}$, $\mathcal{F}_{\mathsf{RO2}} : \{0,1\}^{\kappa} \times \{0,1\}^{(m+\kappa)\times\kappa} \to \mathbb{F}^{m+\kappa}$ and $\mathrm{CRF} : \{0,1\}^{\kappa} \times [m] \times \{0,1\}^{\kappa} \to \{0,1\}^{\kappa}$.
– **Notations:** $\pi_{\mathsf{OT}}^{\kappa}$ implements κ instances of $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$(according to Def. 2).

Seed OT Phase I:

1. For $i \in [\kappa]$, S invokes the ith instance of $\pi_{\mathsf{OT}}^{\kappa}$ with message $(\textsc{Choose}, \mathsf{rec}, \mathsf{sid})$ to obtain s_i. He forms $\mathbf{s} = \{s_1, s_2, \ldots, s_\kappa\} \in \{0,1\}^{\kappa}$.
2. For $i \in [\kappa]$, R invokes the ith instance of $\pi_{\mathsf{OT}}^{\kappa}$ with message $(\textsc{Transfer}, \mathsf{sen}, \mathsf{sid})$ to obtain $\mathbf{k}_{i,0}, \mathbf{k}_{i,1} \in \{0,1\}^{\kappa}$.

OT Extension Phase I:

1. R forms three $(m+\kappa)\times\kappa$ matrices $\mathbf{M}$, $\mathbf{R}$ and $\mathbf{D}$ in the following way and sends $\mathbf{D}$ to S:
   – Sets $\mathbf{M}^i = \mathsf{PRG}(\mathcal{F}_{\mathsf{RO1}}(\mathsf{sid}, \mathbf{k}_{i,0}))$.
   – Samples $\tau \leftarrow \{0,1\}^{\kappa}$ and sets $\mathbf{r}' = \mathbf{r}||\tau$.
   – Sets $\mathbf{R}_j = (r'_j, \ldots, r'_j)$. Clearly, $\mathbf{R}^i = \mathbf{r}'$.
   – Set $\mathbf{D}^i = \mathbf{M}^i \oplus \mathsf{PRG}(\mathcal{F}_{\mathsf{RO1}}(\mathsf{sid}, \mathbf{k}_{i,1})) \oplus \mathbf{R}^i$.

Consistency Check Phase I :

1. R computes challenge $\chi = \{\chi_1, \ldots, \chi_{m+\kappa}\}$ of consistency check as follows: $\widetilde{\chi} = \{\chi_1, \ldots, \chi_{m+\kappa}\} = \mathcal{F}_{\mathsf{RO2}}(\mathsf{sid}, \mathbf{D})$.
2. R computes $\mathbf{u} = \bigoplus_{j\in(m+\kappa)}(\chi_j \cdot \mathbf{M}_j)$ and $\mathbf{v} = \bigoplus_{j\in(m+\kappa)}(\chi_j \cdot \mathbf{R}_j)$. R sends $\mathbf{u}$ and $\mathbf{v}$ to S.

Seed OT Phase II:

1. If S receives a $\textsc{Cheat-Detected}$ message from $\pi_{\mathsf{OT}}^{\kappa}$ then he aborts.
2. S receives $\mathbf{k}_{i,\mathbf{s}_i}$ from the ith instance of $\pi_{\mathsf{OT}}^{\kappa}$ for $i \in [\kappa]$.

Consistency Check Phase II :

1. On receiving $\mathbf{D}$, S forms $(m+\kappa)\times\kappa$ matrix $\mathbf{Q}$ with the jth column of $\mathbf{Q}$ set as $\mathbf{Q}^i = \big(s_i \odot \mathbf{D}^i\big) \oplus \mathsf{PRG}(\mathcal{F}_{\mathsf{RO1}}(\mathsf{sid}, \mathbf{k}_{i,s_i}))$. Clearly, (i) $\mathbf{Q}^i = \big(\mathbf{M}^i \oplus (s_i \odot \mathbf{R}^i)\big)$ and (ii) $\mathbf{Q}_j = \big(\mathbf{M}_j \oplus (\mathbf{s} \odot \mathbf{R}_j)\big) = \big(\mathbf{M}_j \oplus (\mathbf{s} \odot r_j)\big)$.
2. S obtains $\widetilde{\chi}$ values from $\mathbf{D}$ and computes $\mathbf{w} = \bigoplus_{j\in(m+\kappa)}(\chi_j \cdot \mathbf{Q}_j)$. S checks that $\mathbf{w} = \mathbf{u} \oplus \mathbf{s} \cdot \mathbf{v}$.

OT Extension Phase II:

1. For every $j \in [m]$, S computes $\mathbf{y}_{j,0} = a_{j,0} \oplus \mathrm{CRF}(\mathsf{sid}, j, \mathbf{Q}_j)$ and $\mathbf{y}_{j,1} = a_{j,1} \oplus \mathrm{CRF}(\mathsf{sid}, j, \mathbf{Q}_j \oplus \mathbf{s})$. S sends $\{\mathbf{y}_{j,0}, \mathbf{y}_{j,1}\}_{j\in[m]}$ to R.
2. If R obtains $\textsc{Abort}$ message from $\pi_{\mathsf{OT}}^{\kappa}$, then he aborts.
3. For every $j \in [m]$, R recovers $a'_j = \mathbf{y}_{j,r_j} \oplus \mathrm{CRF}(\mathsf{sid}, j, \mathbf{M}_j)$. R outputs $\{a'_j\}_{j\in[m]}$.

5.1 Security Proof

We prove UC-security of our OT extension protocol π_{KOS} by relying on the security properties of $\pi^{\kappa}_{\mathsf{OT}}$, correlation robust function, PRG and RO. More precisely, we prove Thm. 1.

Theorem 1. *Assuming PRG is a secure pseudorandom generator,* CRF *is a tweakable correlation robust function, $\mathcal{F}_{RO2}$ is an observable random oracle and π^{κ}_{OT} implements (according to Def. 2) κ instances of $\mathcal{F}_{SF\text{-}rOT}$, then π_{KOS} UC-securely implements $m = poly(\kappa)$ instances of $\mathcal{F}_{OT}$ functionality.*

Proof. We will first argue security for a corrupt sender and then for a corrupt receiver. In both cases, we give a simulator algorithm and provide an indistinguishability argument.

The simulator for a statically corrupt S^* constructs the $\mathbf{M}, \mathbf{R}$ and $\mathbf{D}$ matrices using $\mathbf{r} = 0^m$ by following the honest receiver algorithm. S^* cannot obtain both $\mathbf{k}_{i,0}$ and $\mathbf{k}_{i,1}$ (due to the security of $\pi^{\kappa}_{\mathsf{OT}}$ against a corrupt receiver). Hence, $\mathbf{R}$ remains hidden due to the PRG security. The simulator will invoke the Ext algorithm of $\pi^{\kappa}_{\mathsf{OT}}$ to obtain the randomness of S^*, i.e. $\mathbf{s}$. Using $\mathbf{s}$, the simulator can compute back the sender's messages. Our simulator has been provided in Fig. 6. We argue indistinguishability between real and ideal world by providing hybrids and proving indistinguishability between each pair of consecutive hybrids as follows:

- **HYB_0:** Real world.
- **HYB_1:** Same as HYB_0, except the reduction constructs $\mathbf{M}, \mathbf{D}$ and $\mathbf{R}$ using $\mathbf{r} = 0^m$. Indistinguishability follows since the corrupt sender cannot compute both messages in the base OTs due to security of $\pi^{\kappa}_{\mathsf{OT}}$. Then one of $\mathcal{F}_{\mathsf{RO1}}(\mathsf{sid}, \mathbf{k}_{i,0})$ and $\mathcal{F}_{\mathsf{RO1}}(\mathsf{sid}, \mathbf{k}_{i,1})$ remains hidden for every $i \in [\kappa]$ due to the RO assumption of $\mathcal{F}_{\mathsf{RO1}}$.
- **HYB_2:** Same as HYB_1, except the reduction extracts $\mathbf{s}$ by invoking Ext algorithm of $\pi^{\kappa}_{\mathsf{OT}}$. Indistinguishability follows due to the correctness of Ext algorithm, which is guaranteed by the security of $\pi^{\kappa}_{\mathsf{OT}}$ when the OT extension sender does not abort.
- **HYB_3:** Same as HYB_2, except the reduction successfully extracts sender's input messages using $\mathbf{M}$ and $\mathbf{s}$. This HYB_3 is identical to HYB_2 due to correctness of the OT extension protocol.

Next, we discuss the simulation for a corrupt R^*. The simulator will extract the $(\mathbf{k}_{i,0}, \mathbf{k}_{i,1})$ values by invoking the simulator(for corrupt sender) of $\pi^{\kappa}_{\mathsf{OT}}$. A corrupt R^* can perform selective failure attack on the base OTs but the KOS protocol is resilient to such attacks. R^* can also construct the $\mathbf{R}$ matrix in such a way that it is not monochrome, i.e. some of the rows of $\mathbf{M}$ does not contain all 0s or all 1s. In such a case the consistency checks detect it with high probability and the sender aborts. If the checks pass, then R^* infers limited knowledge about the bits of $\mathbf{s}$. We refer to the original KOS [KOS15] paper for more details. Our simulator algorithm has been presented in Fig. 7. Next, we present our hybrids and indistinguishability argument:

Fig. 6. Simulation against a statically corrupt S^*

– **Functionalities:** $\mathsf{PRG} : \{0,1\}^\kappa \to \{0,1\}^{m+\mu}$, $\mathcal{F}_{\mathsf{RO1}} : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa$, $\mathcal{F}_{\mathsf{RO2}} : \{0,1\}^\kappa \times \{0,1\}^{(m+\kappa)\times\kappa} \to \mathbb{F}^{m+\kappa}$ and $\mathrm{CRF} : \{0,1\}^\kappa \times [m] \times \{0,1\}^\kappa \to \{0,1\}^\kappa$.

Seed OT Phase I:

1. Sim invokes the ith instance of π^κ_{OT} with message $(\textsc{Transfer}, \mathsf{sen}, \mathsf{sid})$ to obtain $\mathbf{k}_{i,0}, \mathbf{k}_{i,1} \in \{0,1\}^\kappa$ for $i \in [\kappa]$.

OT Extension Phase I:

– Sim forms matrices $\mathbf{M}$, $\mathbf{R}$ and $\mathbf{D}$, following the honest receiver algorithm using $\mathbf{r} = 0^m$.
– Sim sends $\mathbf{D}^i = \mathbf{M}^i \oplus \mathsf{PRG}(\mathcal{F}_{\mathsf{RO1}}(\mathsf{sid}, \mathbf{k}_{i,1})) \oplus \mathbf{R}^i$.

Consistency Check Phase I :

Sim computes $\mathbf{u}$ and $\mathbf{v}$ using the honest receiver algorithm and sends it to S^*.

Seed OT Phase II:

S^* performs his own adversarial algorithm.

Consistency Check Phase II :

S^* performs his own adversarial algorithm.

OT Extension Phase II:

1. For every $j \in [m]$, S^* sends $\{\mathbf{y}_{j,0}, \mathbf{y}_{j,1}\}_{j\in[m]}$ to Sim.
2. If Sim obtains $\textsc{Abort}$ message from π^κ_{OT}, then he aborts.
3. Else, Sim invokes Ext algorithm to obtain $\mathbf{s}$.
4. For every $j \in [m]$, Sim recovers $a'_{j,0} = \mathbf{y}_{j,0} \oplus \mathrm{CRF}(\mathsf{sid}, j, \mathbf{M}_j)$ and $a'_{j,1} = \mathbf{y}_{j,1} \oplus \mathrm{CRF}(\mathsf{sid}, j, \mathbf{M}_j \oplus \mathbf{s})$.
5. For $j \in [m]$, Sim invokes jth instance of $\mathcal{F}_{\mathsf{OT}}$ with input $(a'_{j,0}, a'_{j,1})$.

– **HYB_0:** Real world.
– **HYB_1:** Same as HYB_0, except the reduction invokes the simulator (for corrupt sender) of $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ to obtain $(\mathbf{k}_{i,0}, \mathbf{k}_{i,1})$.
– **HYB_2:** Same as HYB_1, except the reduction aborts if $\mathbf{M}$ has more than μ non-monochromatic rows. The real world sender would abort due to the correctness of the consistency checks, which follow from the RO assumption. Thus, indistinguishability follows from the RO assumption.
– **HYB_3:** Same as HYB_2, except the simulator extracts R^* input and simulates the $\mathbf{y}_{j,0}$ and $\mathbf{y}_{j,1}$ according to the simulation algorithm. Indistinguishability follows due to the CRF assumption.

□

5.2 Efficiency

Our instantiation of the KOS protocol has a minimal overhead of two random oracle, per base OT, on top of the modified KOS protocol of [DKLs18]. The

Fig. 7. Simulation against a statically corrupt R^*

– **Functionalities:** $\mathsf{PRG} : \{0,1\}^\kappa \to \{0,1\}^{m+\mu}$, $\mathcal{F}_{\mathsf{RO1}} : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa$, $\mathcal{F}_{\mathsf{RO2}} : \{0,1\}^\kappa \times \{0,1\}^{(m+\kappa)\times\kappa} \to \mathbb{F}^{m+\kappa}$ and $\mathrm{CRF} : \{0,1\}^\kappa \times [m] \times \{0,1\}^\kappa \to \{0,1\}^\kappa$.

Seed OT Phase I:

For $i \in [\kappa]$, Sim invokes the ith instance of π^κ_{OT} with message $(\textsc{Choose}, \mathsf{rec}, \mathsf{sid})$ to obtain s_i. He forms $\mathbf{s} = \{s_1, s_2, \ldots, s_\kappa\} \in \{0,1\}^\kappa$.

OT Extension Phase I:

R^* sends $\mathbf{D}$ to Sim.

Consistency Check Phase I :

R^* sends $\mathbf{u}$ and $\mathbf{v}$ to Sim.

Seed OT Phase II:

1. If Sim receives a $\textsc{Cheat-Detected}$ message from π^κ_{OT} then he aborts.
2. Sim invokes the simulator (for corrupt sender) of $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ to obtain $(\mathbf{k}_{i,0}, \mathbf{k}_{i,1})$ values for $i \in [\kappa]$.

Consistency Check Phase II :

1. Sim computes $\mathbf{Q}$ matrix following the honest sender algorithm.
2. Sim computes $\mathbf{R}^i = \mathbf{D}^i \oplus \mathsf{PRG}(\mathcal{F}_{\mathsf{RO1}}(\mathsf{sid}, \mathbf{k}_{i,0})) \oplus \mathsf{PRG}(\mathcal{F}_{\mathsf{RO1}}(\mathsf{sid}, \mathbf{k}_{i,1}))$ for $i \in [\kappa]$.
3. Sim verifies $\mathbf{u}$ and $\mathbf{v}$ by following the honest sender algorithm and aborts if check fails.
4. Sim aborts if $\mathbf{M}$ has at least μ non-monochromatic rows.

OT Extension Phase II:

1. For every $j \in [m]$, Sim invokes jth instance of $\mathcal{F}_{\mathsf{OT}}$ with input r_j to obtain a_j.
2. For every $j \in [m]$, Sim sets $\mathbf{y}_{j,r_j} = a_j \oplus \mathrm{CRF}(\mathsf{sid}, j, \mathbf{Q}_j \oplus r_j \cdot \mathbf{s})$ and $\mathbf{y}_{j,\bar{r_j}} \leftarrow \{0,1\}^\kappa$.

original communication complexity remains preserved. Using Fiat-Shamir like transform of [DKLs18] the consistency checks have become non-interactive. In the original KOS protocol, the consistency checks required 2 extra rounds for coin-tossing protocol between the parties. In the next section we will provide an efficient protocol for π^κ_{OT} using 3 rounds. The base-OT messages can be sent in parallel to the OT extension messages. Thus, it would be round preserving and it will circumvent the impossibility result of [GMMM18] since we consider non-blackbox usage of the base-OT protocol.

6 Implementing κ instances of $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$

In this section we present our protocol π^ℓ_{OT} which implements (according to Def. 2) $\ell = \kappa$ instances of $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ assuming Observable Random Oracle, where $\kappa > \mu$. We refer to Sec. 3 for a detailed overview. Our protocol has been presented in Fig. 8.

Fig. 8. Protocol computing ℓ instances of $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ according to Def. 2

$$\pi^{\ell}_{\mathsf{OT}}$$

- **Public Inputs:** Group $\mathbb{G}$, field $\mathbb{Z}_q$ and generator g of group $\mathbb{G}$.
- **Private Inputs:** Sender S and receiver R do not possess any private inputs.
- **Random Oracles:** $\mathcal{F}_{\mathsf{RO1}} : \{0,1\}^{2\kappa} \to \mathbb{G}$, $\mathcal{F}_{\mathsf{RO2}} : \{0,1\}^{\kappa} \times \mathbb{G} \to \{0,1\}^{\kappa}$, $\mathcal{F}_{\mathsf{RO3}} : \{0,1\}^{2\kappa} \to \{0,1\}^{\kappa}$, $\mathcal{F}_{\mathsf{RO4}} : \{0,1\}^{(\ell+1)\kappa} \to \{0,1\}^{\kappa}$.

Choose:

- *Receiver Parameters:* R samples $\mathsf{seed} \leftarrow \{0,1\}^{\kappa}$ and computes $T \leftarrow \mathcal{F}_{\mathsf{RO1}}(\mathsf{sid}, \mathsf{seed})$. R sends seed as OT receiver parameters.
- *Receiver Message:* For $i \in [\ell]$, R computes its message for ith OT as follows :
  - R samples b_i as its random input for ith OT.
  - R samples $\alpha_i \leftarrow \mathbb{Z}_q$ and sets $B_i = g^{\alpha_i} T^{b_i}$
  - R sends B_i as ith OT message.

Transfer:

- *Sender Parameters:* S computes $T \leftarrow \mathcal{F}_{\mathsf{RO1}}(\mathsf{sid}, \mathsf{seed})$. S samples $r \leftarrow \mathbb{Z}_q$ and computes $z = g^r$. S sends z to R as OT sender parameters.
- *Sender Message:* For $i \in [\ell]$, S computes its message for ith OT as follows :
  - S computes $p_{i,0} = \mathcal{F}_{\mathsf{RO2}}(\mathsf{sid}, B_i^r)$ and $p_{i,1} = \mathcal{F}_{\mathsf{RO2}}(\mathsf{sid}, \left(\frac{B_i}{T}\right)^r)$.
  - S sets $(p_{i,0}, p_{i,1})$ as its random inputs messages for ith OT.
- *Challenge Computation:* S computes the challenge for ith OT as $\mathsf{chall}_i = \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{i,0}) \oplus \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{i,1})$ for $i \in [\ell]$. S sends $\mathtt{Chall} = (\mathsf{chall}_1, \mathsf{chall}_2, \ldots, \mathsf{chall}_\ell)$ to R.
- *Proof Computation:* S computes the answer to the challenge as follows :

  $$\mathtt{Ans} = \mathcal{F}_{\mathsf{RO4}}(\mathsf{sid}, \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{1,0}), \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{2,0}), \ldots, \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{\ell,0})).$$

  S computes the validity proof of challenge as $\gamma = \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, \mathtt{Ans})$. S sends the proof γ to R.

Response:

- *Message Decryption:* For $i \in [\ell]$, R computes $p_{i,b_i} = \mathcal{F}_{\mathsf{RO2}}(\mathsf{sid}, z^{\alpha_i})$.
- *Response Computation:* For $i \in [\ell]$, R computes $\mathsf{resp}_i = \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{i,b_i}) \oplus (b \cdot \mathsf{chall}_i)$. R computes response as $\mathtt{Ans}' = \mathcal{F}_{\mathsf{RO4}}(\mathsf{sid}, \mathsf{resp}_1, \mathsf{resp}_2, \ldots, \mathsf{resp}_\ell)$.
- *Challenge Verification:* R aborts if $\mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, \mathtt{Ans}') \neq \gamma$. Else, he sends $\mathtt{Ans}'$ to S and outputs $\{(b_i, p_{b_i})\}_{i \in [\ell]}$.

Verification:

- S aborts if $\mathtt{Ans} \neq \mathtt{Ans}'$. Else, he outputs $(p_{i,0}, p_{i,1})$ as his output for ith OT.

6.1 Security proof.

We prove security of our protocol π_{OT} by proving Theorem 2. In Sec. 5, we will show that such a relaxation in security suffices for KOS OT extension.

Theorem 2. *Assuming the Decisional Diffie-Hellman holds in group $\mathbb{G}$, then π^{κ}_{OT} (Fig. 8) UC-securely implements $\ell(> \mu)$ instances of $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ functionality (according to Def. 2) in the observable random oracle model with sender-sided simulation.*

Proof. We will first argue security for a corrupt sender and then for a corrupt receiver. In the first case we provide a simulator and in the latter part we provide an indistinguishability argument.

The simulator for a statically corrupt sender S^* will use the observability of the ROs to extract the sender's input, since he cannot program $\mathcal{F}_{\mathsf{RO1}}$ on seed. He will compute the receiver message with random input bits. He can compute either $g^{\alpha_i r}$ by following the receiver algorithm and extract one of the sender's input message for ith OT. In order to extract the other sender's input message, he needs to find out T^r. Assume for sake of simplicity, that the simulator's input bit for all OTs are $b_i = 0$ and he can compute $p_{i,0}$ and $a_{i,0}$, for all $i \in [\kappa]$. He tries to compute $p_{i,1}$ values as follows:

1. Observe the queries made by S^* to $\mathcal{F}_{\mathsf{RO3}}$ for computing chall_i using $p_{i,1}$ for every $i \in [\kappa]$. Extract candidate $p_{i,1}$ values s.t. chall is well-formed using $p_{i,0}$ and $p_{i,1}$. It is guaranteed that from every chall_i, Sim can get at most one candidate $p_{i,1}$ value.
2. After obtaining candidate $p_{i,1}$ values for each ith OT, the simulator observes the queries made to $\mathcal{F}_{\mathsf{RO2}}$ to obtain $p_{i,1}$. If there aren't any such query, then the simulator sets adversary's guess for ith OT as 0 in the selective failure attack. Else, simulator will obtain an unique (guaranteed by RO assumption) query $\rho_{i,1}$, s.t. $\mathcal{F}_{\mathsf{RO2}}(\mathsf{sid}, \rho_{i,1}) = p_{i,1}$.
3. Simulator obtains all the candidate values of T^r from $\rho_{i,0}$(which he can compute locally) and $\rho_{i,1}$. Let $\mathbf{A} = \{A_1, A_2, \ldots, A_\ell\}$ denote the list of values obtained as $A_i = \frac{\rho_{i,0}}{\rho_{i,1}}$ for $i \in [\kappa]$ for $i \in [\kappa]$. Let A be the value which has been obtained the maximum number of times from the OTs. The simulator sets A as the supposed T^r value. If there are more than μ OTs whose A_i values are different from A, then simulator sends $(\textsc{Guess}, \mathsf{Sim}, \top, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ and aborts. In such a case, S^* can distinguish the real and ideal world, only if the honest receiver does not abort. To ensure that, S^* must correctly guess the random input bits of the honest R. Thus, S^* can distinguish with $2^{-\mu}$ probability if he miscomputed more than μ OTs. Otherwise, the simulator can extract $T^r = A$ value correctly.
4. Given the correct T^r value, simulator computes the correct $p_{i,1}$ values whose corresponding A_j values were different from A.

Next, the simulator needs to simulate the selective OT failure attacks. Sim simulates the selective failure attack by checking whether chall_i is correctly

formed or not, i.e. if $\mathsf{chall}_i = \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{i,0}) \oplus \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{i,1})$ then $\mathcal{A}$ is not performing a selective failure attack. Else, he has performed a selective failure attack on the receiver's input. Simulator needs to find out the bit b' which adversary has guessed and invoke $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ with $(\textsc{Guess}, \mathsf{Sim}, \mathsf{sid}, \top)$. Sim performs this by observing the queries made by $\mathcal{A}$ to $\mathcal{F}_{\mathsf{RO4}}$ and $\mathcal{F}_{\mathsf{RO3}}$.

1. Sim observes the queries made to $\mathcal{F}_{\mathsf{RO3}}$ and compares it with γ. If γ was formed without querying $\mathcal{F}_{\mathsf{RO3}}$, then Sim aborts as the real world sender would also abort irrespective of input, due to RO assumption. There can be only one such candidate query β s.t. $\mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, \beta) = \gamma$, due to RO assumption. β is the candidate for `Ans` that S must have obtained while computing γ.
2. Next, Sim searches for $y = (y_1 y_2 \dots y_\ell)$, s.t. $\mathcal{F}_{\mathsf{RO4}}(\mathsf{sid}, y) = \beta$, due to RO assumption.
3. Upon finding such a y tuple, it individually checks for y'_i values s.t. $y_i = \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, y'_i)$.
4. These y'_i values are then individually matched with the pads $(p_{i,0}, p_{i,1})$ and chall_i. If $p_{i,0} = y'_i$, then chall_i is correctly formed for $b' = 0$ and so Sim sets $b' = 0$. If $\mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{i,1}) \oplus \mathsf{chall}_i = y'_i$ then chall_i is correctly formed for $b' = 0$ and so Sim sets $b' = 1$. Else, the challenge is malformed for both $b' = 0$ and $b' = 1$; hence the simulator aborts as the honest sender would also abort.

If there are more than μ OTs where the $\mathcal{A}$ has launched a selective failure attack then the simulator aborts and sends $(\textsc{Guess}, \mathsf{Sim}, \mathsf{sid}, \top)$ to $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$. Finally, the simulator invokes $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ with input $(\textsc{Guess}, \mathsf{Sim}, b', \mathsf{sid})$. It should be noted that the input bit b_i remains perfectly hidden in $B_i = g^{\alpha_i}$ since α_i and $\alpha_i - t$ are valid randomness for $b_i = 0$ and $b_i = 1$ respectively. The real and ideal world are statistically indistinguishable except with $2^{-\mu}$ probability. The simulation algorithm has been provided in Fig. 9 and the formal hybrids and indistinguishability argument are as follows:

- **HYB_0:** Real world.
- **HYB_1:** Same as HYB_0, except the reduction except the reduction computes $(p_{i,0}, p_{i,1})$ by following the simulation strategy and computes $A = T^r$, or aborts if necessary. Indistinguishability follows due to RO assumption as discussed previously.
- **HYB_2:** Same as HYB_1, except the reduction invokes $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ with input $p_{i,0} = p_{i,1} = \perp$ and aborts if there are 0 or 2 (and more) candidate values for β. Indistinguishability follows from RO assumption.
- **HYB_3:** Same as HYB_2, except the reduction invokes $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ with input $p_{i,0} = p_{i,1} = \perp$ and aborts if there are 0 or 2 (and more) candidate values for y. Indistinguishability follows from RO assumption.
- **HYB_4:** Same as HYB_3, except the reduction invokes $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ with input $p_{i,0} = p_{i,1} = \perp$ and aborts if there are 0 or 2 (and more) candidate values for y'_i for each $i \in [\ell]$. Indistinguishability follows from RO assumption.
- **HYB_5:** Same as HYB_4, except the reduction aborts if there are ¿μ OTs where the sender has launched a selective failure attack. Indistinguishability follows statistically since the inputs of the honest sender are identically distributed

to the inputs of the reduction, and both are random. The probability that the reduction aborts and the sender doesn't abort in the real world is $2^{-\mu}$.

- **HYB_6:** Same as HYB_5, except the reduction simulates the selective failure attack following the simulation algorithm. Indistinguishability follows due to the RO assumption.
- **HYB_7:** Same as HYB_6, except Sim computes $(p_{i,0}, p_{i,1})$ following the simulation strategy and invokes $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ with it. This hybrid is identical to the previous hybrid.

This completes the security proof for a corrupt sender. Next, we discuss the case for a corrupt receiver. In this case, the simulator either extracts the input bit of a corrupt receiver or it aborts. In the real world, the honest sender would also abort. However, in both cases the receiver can obtain the message corresponding to his bit after the OT protocol results in an abort by the sender. More formally, the simulator for a corrupt receiver R^* will set $(c_{i,0}, c_{i,1})$ values randomly. Later, upon obtaining the second OT second message, the receiver can follow either of the two tactics:

- **`Resp` is valid** : The receiver can query $\mathcal{F}_{\mathsf{RO2}}$ to compute p_{i,b_i}, corresponding to his input bit b_i for ith OT. Then he can correctly compute the response to the challenge by running the honest receiver algorithm. This would result in the sender accepting the response to the challenge. The simulator can observe the queries made to the random oracles to extract b_i and invoke $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ with input b_i.
- **`Resp` is invalid** : On the other hand, the receiver can send a random response to the challenge and force the sender to abort. However, the receiver can decrypt the message corresponding to his input bit b_i after the protocol ends by running the honest receiver's algorithm. This would hamper simulation as the simulator cannot extract b_i during the protocol since R^* did not query the random oracles. Hence, the message decrypted (after the protocol aborted) by R^* in the simulated world will be distinguishable from the message decrypted (after the protocol aborted) by R^* in the simulated world. Based on a_{b_i}, the view of R^* can be distinguished by the environment $\mathcal{Z}$; hence simulation would fail.

Next, we show indistinguishability based security for a corrupt receiver. We demonstrate that there exists a PPT algorithm Ext who can extract the input choice bit of R^* if Ext has blackbox access to R^* for the protocol session. If R^* decides to forcefully abort the protocol, then it is guaranteed that he cannot compute both sender input messages as that would require solving the CDH problem. We present our Ext algorithm in Fig. 10 and we modularly discuss the details of our proof by arguing that each property in Def. 2 holds for our protocol.

- **Correctness of Ext algorithm:** The corrupt receiver has to compute a correct answer to the challenge. To do that, he has to query either B_i^r or $(B_i \cdot T^{-1})^r$ to $\mathcal{F}_{\mathsf{RO2}}$, to obtain $p_{i,0}$ or $p_{i,1}$ and construct the correct response

Fig. 9. Simulation against a statically corrupt S^*

– **Functionalities:** Random Oracles $\mathcal{F}_{\mathsf{RO1}} : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \mathbb{G}$, $\mathcal{F}_{\mathsf{RO2}} : \{0,1\}^\kappa \times \mathbb{G} \to \{0,1\}^\kappa$, $\mathcal{F}_{\mathsf{RO3}} : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa$, $\mathcal{F}_{\mathsf{RO4}} : \{0,1\}^\kappa \times \{0,1\}^{\ell\kappa} \to \{0,1\}^\kappa$.

Choose:

– *Receiver Parameters:* Sim runs the honest receiver algorithm.

– *Receiver Message:* Sim runs the honest receiver algorithm.

Transfer:

– S^* sends $(z, \mathtt{Chall}, \gamma)$.

Response:

– *Message Decryption:* Sim extracts and sender's messages as follows:
- Sim computes $p_{i,b_i} = \mathcal{F}_{\mathsf{RO2}}(\mathsf{sid}, z_i^\alpha)$.
- Sim extracts candidate $p_{i,\bar{b_i}}$ values for each $i \in [\kappa]$ by observing the queries made to $\mathcal{F}_{\mathsf{RO3}}$ for computing $\mathsf{chall}_i = \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{i,0}) \oplus \mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{i,1})$.
- Sim extracts $\rho_{i,\bar{b_i}}$ from the query list of $\mathcal{F}_{\mathsf{RO2}}$, s.t. $\mathcal{F}_{\mathsf{RO2}}(\mathsf{sid}, \rho_{i,\bar{b_i}}) = p_{i,\bar{b_i}}$.
- Sim computes $\mathbf{A} = \{A_i\}_{i\in[\kappa]} = \frac{\rho_{i,0}}{\rho_{i,1}}$. Sets A as the most frequent A_i value in $\mathbf{A}$. If there are at least μ A_i values s.t. $A_i \neq A$, then invoke $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ with message $(\textsc{Guess}, \mathsf{Sim}, \mathsf{sid}, \top)$ and abort. Else, consider $T^r = A$.
- Sim computes the correct value of $p_{i,\bar{b_i}} = \mathcal{F}_{\mathsf{RO2}}(\mathsf{sid}, z^{\alpha_i} \cdot A^{-1})$.

– *Challenge Verification and Response Computation:* Sim extracts values by observing RO queries as follows:
- Sim extracts β s.t. $\mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, \beta) = \gamma$. Set $\mathtt{Ans} = \beta$ by observing $\mathcal{F}_{\mathsf{RO3}}$. Sim observes $\mathcal{F}_{\mathsf{RO4}}$ to extract $y = (y_1, y_2, \ldots, y_\ell)$, s.t. $\mathcal{F}_{\mathsf{RO4}}(\mathsf{sid}, y) = \mathtt{Ans}$.
- For $i \in [\ell]$, Sim extracts y_i' s.t. $\mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, y_i') = y_i$ for each $i \in [\ell]$.

If Sim either finds two or more matching queries, or he finds no matching query then he invokes $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ with input messages $p_{i,0} = p_{i,1} = \perp$ and aborts. For $i \in [\ell]$, Sim computes $\mathsf{chall}_i' = \mathcal{F}_{\mathsf{RO3}}(p_{i,0}) \oplus \mathcal{F}_{\mathsf{RO3}}(p_{i,1})$ and performs the following:
- If $\mathsf{chall}_i = \mathsf{chall}_i'$: If $y_i' = p_{i,0}$ and $p_{i,1}$ was queried to $\mathcal{F}_{\mathsf{RO3}}$, then invoke $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ with input $(\textsc{Guess}, \mathsf{Sim}, \mathsf{sid}, \perp)$ else abort.
- If $\mathsf{chall}_i \neq \mathsf{chall}_i'$: If $y_i' = p_{i,0}$ then set $b_i' = 0$ else if $\mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{i,0}) \oplus \mathsf{chall}_i = y_i'$ then set $b_i' = 1$. Invoke $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ with input $(\textsc{Guess}, \mathsf{Sim}, \mathsf{sid}, b_i')$.
- Else, Sim aborts in the simulated execution.

– If Sim receives $(\textsc{Cheat-Detected})$ from any $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ instance then he aborts.

– For $i \in [\ell]$, Sim computes $\mathtt{Ans}'$ following honest receiver algorithm using input $\{b_i\}_{i\in[\ell]}$. Sends $\mathtt{Ans}'$ to S^* or he aborts.

– Sim invokes ith instance of $\mathcal{F}_{\mathsf{SF\text{-}rOT}}$ with input $(\textsc{Transfer}^*, \mathsf{sen}, \mathsf{sid}, (p_{i,0}, p_{i,1}))$ for $i \in [\ell]$.

using $\mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{i,0})$ or $\mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{i,1})$. He can bypass querying the RO if he can correctly guess $p_{i,0}$ or $p_{i,1}$ or $\mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{i,0})$ or $\mathcal{F}_{\mathsf{RO3}}(\mathsf{sid}, p_{i,1})$. However, that occurs with negligible probability. Thus, the Ext algorithm succeeds if

Fig. 10. Extractor Algorithm Ext

– **Functionalities:** Random Oracles $\mathcal{F}_{\mathsf{RO1}} : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \mathbb{G}$, $\mathcal{F}_{\mathsf{RO2}} : \{0,1\}^\kappa \times \mathbb{G} \to \{0,1\}^\kappa$, $\mathcal{F}_{\mathsf{RO3}} : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa$, $\mathcal{F}_{\mathsf{RO4}} : \{0,1\}^\kappa \times \{0,1\}^{\ell\kappa} \to \{0,1\}^\kappa$.

Choose:

– *Receiver Parameters:* R^* sends seed as OT receiver parameters.
– *Receiver Message:* For $i \in [\ell]$, R^* sends B_i as ith OT message.

Transfer:
Ext follows honest sender algorithm.

Response:
R^* sends $\mathtt{Ans}'$.

Verification:

– If $\mathtt{Ans}'$ is not valid then set $b_i = \perp$ for $i \in [\ell]$ and abort.
– For $i \in [\ell]$, Ext extracts b_i as follows and performs the following -
  • If R^* queried both B_i^r and $(B_i \cdot T^{-1})^r$ to $\mathcal{F}_{\mathsf{RO3}}$ then set $b_i = \perp$.
  • If R^* queried $\rho_{i,b_i'} = (B_i \cdot T^{-b_i'})^r$ to $\mathcal{F}_{\mathsf{RO3}}$ to obtain $p_{i,b_i'}$ then set $b_i = b_i'$ else set $b_i = \perp$.
  • Output b_i.

R^* correctly responds to the challenge. In such a case, R^* queries $\mathcal{F}_{\mathsf{RO2}}$; hence Ext can correctly extract b_i.

– **R^* cannot compute both $p_{i,0}$ and $p_{i,1}$:** It can be observed that if R^* obtains both $p_{i,0}$ and $p_{i,1}$ by querying $\mathcal{F}_{\mathsf{RO2}}$ on $\rho_{i,0}$ and $\rho_{i,1}$ respectively, then he can be used to solve the CDH problem where the CDH challenge instance is $T = g^t$ and $z = g^r$. The solution to the CDH challenge would be $T^r = \frac{\rho_{i,0}}{\rho_{i,1}}$. Here, we can assume that the reduction programs $\mathcal{F}_{\mathsf{RO1}}$ on seed s.t. it returns the CDH challenge T. This is a reasonable assumption to make since we are programming the RO in the reduction. Such programming instances can be found out in the work of [DKLs18].

– **Indistinguishability of R^* views:** The real world view of the corrupt receiver R^* - $\mathcal{V}^{\mathsf{S}}_{\pi_{\mathsf{OT}},\mathsf{R}^*(z)}((p_{i,0},p_{i,1}),b,z)_{z\in\{0,1\}^*}$, after executing an OT protocol with S using random inputs is indistinguishable from the ideal world view of R^*- $\mathcal{V}^{\mathsf{S}}_{\pi_{\mathsf{OT}},\mathsf{R}^*(z)}((\widetilde{p_{i,0}},\widetilde{p_{i,1}}),b,z)_{z\in\{0,1\}^*}$, after executing an OT protocol with sender since the sender only sends z. This is because R^* cannot query $\rho_{i,\widetilde{b_i}}$ to $\mathcal{F}_{\mathsf{RO2}}$ (due to CDH assumption) and hence $p_{i,\overline{b_i}}$ and $\widetilde{p_{i,\overline{b_i}}} = 0^\kappa$ would look indistinguishable due to the RO assumption.

This completes our proof of Thm. 2. □

RTT	< 0.1 ms	50 ms	100 ms	200 ms
CO-OT [CO15]	21 ms	67 ms	117 ms	217 ms
This work	21 ms	67 ms	117 ms	217 ms

Table 2. Comparing the performance to compute $\kappa = 128$ base OTs using our protocol and CO-OT.

6.2 Efficiency

Overall the complexity of our protocol is similar to the CO-OT protocol. Our protocol requires the receiver to compute 2 exponentiations and the sender to compute 1 exponentiation for each OT. The sender needs to compute 5 RO queries and the receiver need to query the RO for 4 times, for each OT. The receiver needs to communicate one group element and one κ bit string for each OT. The sender needs to send 4κ bit strings for each OT.

In addition, the sender has a one-time computation of 1 exponentiation, one RO query and communication of one group element, which can be reused. The receiver has a one-time communication of κ bit string and one-time computation of one RO query.

7 Implementation and Evaluation

We will study the concrete performance of our OT protocol in this section. As we have analyzed in previous sections, our OT protocol is expected to be as fast as the CO-OT protocol by Chou and Orlandi [CO15], which is the most efficient OT protocol but not provably UC-secure and does not provide input extraction of a corrupt receiver. Since all state-of-the-art OT protocols are slower than CO-OT, the above is sufficient to demonstrate the efficiency of our protocol against all other alternatives [MR19, DKLs18].

We implement CO-OT and our protocol using relic-toolkit [AG] and test them on a machine with a 3 GHz Intel Xeon CPU. No multi-thread or assembly-level optimization is used. We throttle the network bandwidth to be 1 Gbps but with different network round-trip time (RTT, measured by `ping`). The performance is summarized in Table 2, where we can see that the performance of our protocol is identical to the CO-OT protocol for different network latency values. This is expected as both protocols have the same number of exponentiation operations. Note that prior works [CO15, MR19] reported performance of CO-OT with low-level hardware-dependent accelerations. Our protocol can benefit from them too, resulting in even higher performance. We further applied both protocols as the base OT to compute OT extension. We can see that due to the reduce round complexity of our protocol, we are to obtain a better efficiency with the overall running time improved by one RTT.

RTT	< 0.1 ms	50 ms	100 ms	200 ms
CO-OT [CO15]+KOS	1300 ms	1388 ms	1496 ms	1679 ms
This work +KOS	1300 ms	1327 ms	1391 ms	1485 ms

Table 3. Comparing the performance to compute 10^7 random OTs using KOS with base OT as our protocol and CO-OT.

Acknowledgements

This work was supported by the the IARPA ACHILLES project, the NSF MACS project and NSF grant CNS-1422965. The first author also thanks the Check Point Institute for Information Security.

References

AG. D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic.

ALSZ15. Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions with security for malicious adversaries. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 673–701. Springer, Heidelberg, April 2015.

BCG+19. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 291–308. ACM Press, November 2019.

BCGI18. Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 896–912. ACM Press, October 2018.

Bea96. Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th Annual ACM Symposium on Theory of Computing*, pages 479–488. ACM Press, May 1996.

BLO16. Aner Ben-Efraim, Yehuda Lindell, and Eran Omri. Optimizing semi-honest secure multiparty computation for the internet. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 578–590. ACM Press, October 2016.

BPRS17. Megha Byali, Arpita Patra, Divya Ravi, and Pratik Sarkar. Fast and universally-composable oblivious transfer and commitment scheme with adaptive security. Cryptology ePrint Archive, Report 2017/1165, 2017. https://eprint.iacr.org/2017/1165.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, October 2001.

CDG+18. Jan Camenisch, Manu Drijvers, Tommaso Gagliardoni, Anja Lehmann, and Gregory Neven. The wonderful world of global random oracles. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 280–312. Springer, Heidelberg, April / May 2018.

CJS14. Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014: 21st Conference on Computer and Communications Security*, pages 597–608. ACM Press, November 2014.

CO15. Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *Progress in Cryptology - LATINCRYPT 2015: 4th International Conference on Cryptology and Information Security in Latin America*, volume 9230 of *Lecture Notes in Computer Science*, pages 40–58. Springer, Heidelberg, August 2015.

DKLs18. Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. Secure two-party threshold ECDSA from ECDSA assumptions. In *2018 IEEE Symposium on Security and Privacy*, pages 980–997. IEEE Computer Society Press, May 2018.

GIR17. Ziya Alper Genç, Vincenzo Iovino, and Alfredo Rial. "The simplest protocol for oblivious transfer" revisited. Cryptology ePrint Archive, Report 2017/370, 2017. http://eprint.iacr.org/2017/370.

GKW+19. Chun Guo, Jonathan Katz, Xiao Wang, Chenkai Weng, and Yu Yu. Better concrete security for half-gates garbling (in the multi-instance setting). Cryptology ePrint Archive, Report 2019/1168, 2019. https://eprint.iacr.org/2019/1168.

GMMM18. Sanjam Garg, Mohammad Mahmoody, Daniel Masny, and Izaak Meckler. On the round complexity of OT extension. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 545–574. Springer, Heidelberg, August 2018.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM Press, May 1987.

HL17. Eduard Hauck and Julian Loss. Efficient and universally composable protocols for oblivious transfer from the CDH assumption. Cryptology ePrint Archive, Report 2017/1011, 2017. http://eprint.iacr.org/2017/1011.

HSS17. Carmit Hazay, Peter Scholl, and Eduardo Soria-Vazquez. Low cost constant round MPC combining BMR and oblivious transfer. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 598–628. Springer, Heidelberg, December 2017.

IKNP03. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161. Springer, Heidelberg, August 2003.

IR89. Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 44–61, 1989.

Kil88. Joe Kilian. Zero-knowledge with log-space verifiers. In *29th Annual Symposium on Foundations of Computer Science*, pages 25–35. IEEE Computer Society Press, October 1988.

KOS15. Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 724–741. Springer, Heidelberg, August 2015.

LM18. Baiyu Li and Daniele Micciancio. Equational security proofs of oblivious transfer protocols. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 527–553. Springer, Heidelberg, March 2018.

MR19. Daniel Masny and Peter Rindal. Endemic oblivious transfer. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 309–326. ACM Press, November 2019.

NNOB12. Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 681–700. Springer, Heidelberg, August 2012.

OOS17. Michele Orrù, Emmanuela Orsini, and Peter Scholl. Actively secure 1-out-of-N OT extension with application to private set intersection. In Helena Handschuh, editor, *Topics in Cryptology – CT-RSA 2017*, volume 10159 of *Lecture Notes in Computer Science*, pages 381–396. Springer, Heidelberg, February 2017.

PSS17. Arpita Patra, Pratik Sarkar, and Ajith Suresh. Fast actively secure OT extension for short secrets. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*, 2017.

PVW08. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, Heidelberg, August 2008.

WRK17. Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Global-scale secure multiparty computation. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 39–56. ACM Press, October / November 2017.

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society Press, October 1986.