From Privacy-Only to Simulatable OT: Black-Box, Round-Optimal, Information-Theoretic

Varun Madathil ✉
North Carolina State University, Raleigh, NC, USA

Chris Orsini ✉
North Carolina State University, Raleigh, NC, USA

Alessandra Scafuro ✉
North Carolina State University, Raleigh, NC, USA

Daniele Venturi ✉
Sapienza University of Rome, Italy

Abstract

We present an information-theoretic transformation from any 2-round OT protocol with only *game-based* security in the presence of malicious adversaries into a 4-round (which is known to be optimal) OT protocol with simulation-based security in the presence of malicious adversaries.

Our transform is the first satisfying all of the following properties at the same time:

- It is in the *plain model*, without requiring any setup assumption.
- It only makes *black-box* usage of the underlying OT protocol.
- It is *information-theoretic*, as it does not require any further cryptographic assumption (besides the existence of the underlying OT protocol).

Additionally, our transform yields a cubic improvement in communication complexity over the best previously known transformation.

2012 ACM Subject Classification Theory of computation → Cryptographic protocols; Security and privacy → Information-theoretic techniques

Keywords and phrases Oblivious Transfer, Black-Box compiler, Malicious Security, Plain Model

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.5

Related Version *Full Version*: https://eprint.iacr.org/2022/034.pdf

Funding *Varun Madathil*: Protocol Labs, NSF grants #1718074,#1764025.
Alessandra Scafuro: Protocol Labs, NSF grants #1718074,#1764025.

1 Introduction

Oblivious Transfer (OT), introduced by Rabin [41], is a core primitive in cryptography. Intuitively, an OT protocol considers a sender with input a pair of strings (s_0, s_1) and a receiver with input a choice bit b. At the end of the protocol, the receiver should learn the string s_b (and nothing more), without the sender obtaining any information about b.

Privacy-only OT. Perhaps, the most basic way to define security of OT is to require that the receiver's messages are computationally indistinguishable when $b = 0$ and when $b = 1$, while the sender's messages computationally hide s_{1-b}. An OT protocol satisfying this property in the presence of *malicious* adversaries is sometimes referred to as *privacy-only* [22]. Privacy-only OT can be constructed from both generic assumptions such as the existence of trapdoor permutations, additively homomorphic encryption and public-key encryption with oblivious public key generation (see, e.g., [2, 15]), and concrete assumptions such as Decisional

Diffie-Hellman [36, 1], Quadratic Residuosity and Decisional Composite Residuosity [21], and Learning with Errors [5]. In particular, *2-round* privacy-only OT protocols against *malicious* adversaries are known under all such assumptions in the *plain model*.

Simulatable OT. Kilian [30] shows that an *ideal* "OT oracle" is sufficient to securely compute *any* cryptographic task. This seminal result has been extended in many different ways [27, 26, 4, 14], thus making OT a central tool in cryptography. Unfortunately, privacy-only OT is not sufficient to instantiate an ideal OT oracle, which instead requires a flavour of security known as *simulatability*. A simulatable OT protocol admits a polynomial-time algorithm, called *simulator*, that is able to fake transcripts of the real protocol without knowing the inputs of the honest parties, and by only having access to the ideal OT oracle.

To make an OT protocol simulatable, intuitively, one needs to augment the protocol with mechanisms that allow a simulator to extract the inputs of the malicious party from the protocol messages, as well as to correctly compute the messages from the honest parties without knowing their inputs. When a setup assumption, such as a Common Reference String (CRS) is available, it is possible to transform privacy-only OT to simulatable OT by embedding special trapdoors in the CRS: the simulator can use these trapdoors to "decrypt"/"equivocate" the protocol messages of the malicious/honest party. Indeed, a rich line of work shows that two rounds are sufficient in order to obtain simulatable OT in the CRS model [13, 28, 40, 9, 11]. Alternatively, 2-round simulatable OT can be obtained in the Random Oracle Model (ROM) [3, 7, 6] where parties have oracle access to a truly-random hash function.

Round-optimal simulatable OT in the plain model. While two rounds are sufficient to build simulatable OT in the CRS model and in the ROM, Katz and Ostrovsky [29] show that *four* rounds are *necessary* for simulatable OT in the plain model. The need for four rounds comes from the fact that, without assuming setup, there are no extraction trapdoors that the simulator can use. Hence, to extract the inputs of the malicious party, the simulator must use rewinding, which requires at least three rounds of communication.[1] In the same paper, Katz and Ostrovsky show that four rounds are also *sufficient*, by providing a simulatable OT protocol from certified trapdoor permutations. Their construction leverages 3-round witness indistinguishable (WI) arguments of knowledge (AoK) and 4-round zero-knowledge (ZK) AoK, to force parties to behave honestly and consistently with the OT protocol.

It is folklore[2] that a similar approach, based on adding general WI/ZK-AoK, could be used to transform any 2-round privacy-only OT to round-optimal simulatable OT. However, the latter would entail unrolling the computation of the underlying OT into a Boolean (or arithmetic) circuit, and using the OT circuit and the transcript as a statement for the WI/ZK proof of consistency; the secret inputs used to compute the transcript would instead be the witnesses for the proof. The AoK property of WI/ZK proofs enables a simulator to extract the witness for the proof by rewinding, whereas the soundness property ensures that the witnesses extracted by the simulator are consistent with the transcript of the underlying privacy-only OT protocol. While very general, such a non-black-box approach requires to unroll the circuit of the OT, hence the complexity of the compiler depends on the circuit complexity of the underlying OT (and not on the security parameter and inputs/outputs of the protocol).

[1] The lower bound of [29] works only for black-box simulators. However, note that even assuming a non-black-box simulator we are not aware of any technique allowing to extract inputs in less than three rounds (unless one assumes non-falsifiable assumptions, which are not considered plain model).

[2] Though we are not aware of any paper formally proving this.

Table 1 Comparing our work to existing compilers to fully simulatable oblivious transfer.

Work	OT from	Black-box Black-box	Plain Model	Optimal Optimal	Information theoretic
[29]	Certified TDPs	✗	✓	✓	✗
[20]	Semi-honest OT	✓	✓	✗	✗
[37]	Certified TDPs	✓	✓	✓	✗
[12]	Strongly-uniform OT	✓	✓	✓	✗
[10]	TDPs	✓	✓	✓	✗
This work	Defensible OT	✓	✓	✓	✓
This work	Semi-honest OT	✓	✗	✓	✓

A central question in cryptography is to understand whether a given task can be performed having only *black-box* access[3] to other cryptographic primitives. Thus, it is natural to ask whether one can provide a general black-box compiler from privacy-only OT to (possibly round-optimal) simulatable OT.

Black-box round-optimal simulatable OT in the plain model. When treating the underlying OT as a black-box the main challenge is to add an extraction/rewinding mechanism, from which the simulator can extract values that are *consistent* with the actual secret inputs played in the OT protocol, *and* such that the output of the honest parties is distributed identically in the real and ideal world.[4] In fact, Lindell and Pinkas [34] show that there are input-dependent attacks that emerge uniquely in the black-box approach. Haitner, Ishai, Kushilevitz, Lindell, and Petrank [19, 20] showed that input-extraction and input-consistency are possible to achieve via cut-and-choose techniques, but unfortunately their compiler requires at least 12 rounds (assuming some steps can be parallelized).

In a different work, Ostrovsky, Richelson and Scafuro [37] provided a 4-round simulatable OT from black-box use of certified trapdoor permutations. Very recently, Friolo, Masny and Venturi [12] greatly generalized the approach of [37] by exhibiting a compiler that transforms any *strongly-uniform* 2-round OT protocol into a 4-round simulatable OT. Strong uniformity means that the messages sent by the receiver appear computationally indistinguishable from random to a malicious sender.[5] Importantly, as shown in [12], strongly-uniform OT can be instantiated from the most common number-theoretic assumptions (e.g., DDH, CDH, LPN, Subset Sum, and LWE).

Yet, the compiler of [12] inherits the complexity and significant overhead of its predecessor. Indeed, as in [37], it entails two intermediate transformations: one for achieving simulatability against malicious receivers, and one for achieving simulatability against malicious senders. Each transformation is somewhat complex and requires the use of extractable commitments (to extract the inputs) and black-box commit-and-prove of equality (in particular, a modification of the one built by Kilian [30]) to enforce input consistency w.r.t. the underlying OT. In particular, the resulting communication complexity of this compiler is *quartic* in the security parameter (i.e., $O(|\mu_R|\lambda^4 + 2|\mu_S|\lambda)$ where μ_R is the message sent by the receiver and μ_S is the message sent by the sender in the underlying 2-round OT protocol).

[3] Black-box means that the underlying OT is treated as an oracle.
[4] Note that this consistency property comes for free when using the underlying OT protocol in a non-black-box way, since the protocol transcript is part of the statement of the WI/ZK proof.
[5] Specifically, they require an OT protocol that is strongly-uniform against malicious senders and simulatable in presence of semi-honest receivers.

More recently, Choudhuri, Ciampi, Goyal, Jain and Ostrovsky [10] constructed 4-round simulatable OT from black-box use of trapdoor permutations. Their construction still relies on the inefficient transformation from [37] to go from one-sided simulatable to fully simulatable OT.

Hence, the question:

Does there exist an efficient, information-theoretic, black-box transform from privacy-only OT to round-optimal simulatable OT in the plain model?

1.1 Our Contribution

In this work, we answer the above question in the positive by providing a simple, black-box, information-theoretic, transformation turning any privacy-only 2-round OT protocol into a round-optimal simulatable OT protocol. We elaborate on our contributions in more details below.

An information-theoretic transform. Our transformation does not require any additional cryptographic assumption besides the existence of privacy-only OT. As a result, our approach is much simpler than previous work and, in fact, yields an improved communication complexity of $O(|\mu_R|\lambda + |\mu_S|\lambda + \lambda^2)$. In particular, we prove the following theorem:

▶ **Theorem 1** (Main Theorem, informal). *There is a black-box information-theoretic transformation from any 2-round privacy-only OT protocol to a 4-round simulatable OT protocol in the plain model.*

Towards assuming semi-honest privacy only. From a theoretical perspective, the holy grail in this line of research would be to build a round-optimal simulatable OT protocol that makes black-box usage of any *semi-honest* privacy-only 2-round OT protocol. While we do not settle this question in the plain model, we do give a positive answer in the ROM as explained below.

First, we observe that our compiler only requires privacy against *defensible* receivers [20], which is a weaker flavour of privacy than privacy only OT, which is private against malicious receivers. Defensible here refers to the fact that, while a malicious receiver can cheat in the protocol and learn both inputs of the sender without being detected, it should be hard to later convince the sender that it behaved honestly. Second, we prove that any 2-round semi-honest privacy-only OT is necessarily private against malicious senders.

Putting together the above two observations, we can plug into our transform any 2-round OT protocol that is both: (i) private against defensible receivers, and (ii) private against semi-honest senders. Next, we show that in the ROM we can relax the security requirements for the underlying OT even further to just requiring semi-honest privacy against both the sender and the receiver. More in details, we exhibit a transformation turning any 2-round semi-honest privacy-only OT into one satisfying properties (i) and (ii) above. Our transform is round-preserving, and simply requires the receiver to use randomness derived from the output of the random oracle. The programmability of the random oracle is used only in the reduction.

As mentioned earlier, it is well known that if the simulator is allowed to both observe and program the random oracle, it is fairly easy to build a simulatable OT protocol. Yet, since in our transform the random oracle is used only to lift (game-based) privacy against semi-honest receivers to (game-based) privacy against defensible receivers, we believe the gap to fill towards a result in the plain model is much narrower. In other words, future work must only focus on finding a round-preserving transformation from OT with privacy against semi-honest receivers to OT with privacy against defensible receivers in the plain model.

1.2 Our Techniques

As mentioned above, in the black-box setting, the main challenge towards obtaining simulatable OT is to design extraction mechanisms which allow the simulator to extract inputs that are consistent with the ones played by the parties in the real world. The latter is particularly challenging when only 4 rounds of communication are available.

The compiler of Friolo, Masny and Venturi [12] achieves extractability and input consistency by adding black-box commit-and-prove proofs of consistency. In particular, they rely on such proofs for two reasons: (1) to force the receiver to sample one of the messages for the underlying OT protocol uniformly, and (2) to force the sender to create a valid secret sharing of its inputs (without opening the way to input-dependent abort attacks against the receiver), while allowing the simulator to successfully reconstruct both inputs.

In this work, we take a completely different approach. In particular, instead of adding mechanisms to *force* good behaviour, we only add *publicly-verifiable* checkpoints to *assess* good behavior. Public verifiability here means that the checkpoints are verifiable by looking only at the protocol transcript (without requiring access to secret inputs), which avoids attacks based on input-dependent aborts. As a result, we can enforce both extractability and indistinguishability of the simulation, by simply having parties justify some of their actions (when challenged). Thanks to this feature, our transform does not require any additional cryptographic primitives (e.g., commitments), and can be based just on privacy-only OT and threshold secret sharing.

Overview of our compiler. The sender and the receiver engage in m parallel *sessions* $(\mu_R^{(i)}, \mu_S^{(i)})$ of the underlying 2-round privacy-only OT protocol, using uniformly random inputs: the receiver uses random choice bits $b^{(1)}, \ldots, b^{(m)}$, whereas the sender uses pairs of random keys $(\kappa_0^{(1)}, \kappa_1^{(1)}), \ldots, (\kappa_0^{(m)}, \kappa_1^{(m)})$. This results in messages $\mu_R^{(i)}$ which are sent from the receiver to the sender in the first round of the compiled protocol.

In the second round, the sender responds to the messages received from the receiver with its own messages $\mu_S^{(i)}$ (computed via the underlying 2-round OT protocol), but also selects a random subset $\mathcal{A}$ of t_R indices in $[m]$ for cut-and-choose: In the third round, for each $i \in \mathcal{A}$, the receiver is asked to provide the randomness $\rho_R^{(i)}$ and the input $b^{(i)}$ (we call these a *defense*) that explain the message $\mu_R^{(i)}$ sent in the i-th session. Additionally, the receiver selects a random subset $\mathcal{B}$ of t_S indices in $[m] \setminus \mathcal{A}$ and forwards $\mathcal{B}$ and the defenses $(\delta_R^{(i)})_{i \in \mathcal{A}}$ along with a bit $d^{(i)} = b \oplus b^{(i)}$ for each of the $n = m - t_R - t_S$ sessions that were not selected for cut and choose (we call those the *alive sessions*). The bit $d^{(i)}$ allows to adjust the bit $b^{(i)}$ in the i-th session to the choice bit b of the receiver.

We note that, since the underlying OT satisfies privacy against malicious senders, the random bits $b^{(i)}$ used by the receiver are computationally hidden, which implies the adjusting bits $d^{(i)}$ are computationally close to uniform and hide the receiver's choice bit b to the eyes of a computationally-bounded malicious sender. Moreover, observe that at the end of the execution of the underlying 2-round OT sessions, the receiver might have already noticed that some of the messages $\mu_S^{(i)}$ played by the sender are "bad", in the sense that they yield an invalid output (we do not make any assumption about how the underlying OT protocol deals with bogus inputs). Hence, in such a case, it seems the receiver should just abort (and righteously so) instead of continuing with the protocol. Doing so, however, opens the door to attacks based on input-dependent abort. For instance, the sender could plant a single $\perp$ in, say, the j-th session, by playing with the inputs $(\kappa_0^{(j)}, \perp)$, and thus learning that $b^{(j)} = 0$ in case the receiver did not abort. Later, after observing $d^{(j)}$, the sender can compute $b = d^{(j)} \oplus 0$ which is a clear security breach. To prevent this type of attack, in our compiler we never let parties abort depending on their local view. (In fact, up to this point, in our protocol the parties eventually abort only after checking the responses to cut-and-choose challenges, which do not involve secret inputs.)

In the fourth (and last) round, for each $i \in \mathcal{B}$, the sender reveals the randomness $\rho_S^{(i)}$ and the inputs $(\kappa_0^{(i)}, \kappa_1^{(i)})$ that explain the message $\mu_S^{(i)}$ sent in the i-th session. Moreover, it uses the n pairs of keys $(\kappa_0^{(i)}, \kappa_1^{(i)})$ corresponding to each alive session to mask n shares $s_0^{(i)}$ and $s_1^{(i)}$ of the actual secret inputs s_0 and s_1, yielding ciphertexts $(\gamma_0^{(i)}, \gamma_1^{(i)})$; here, we make use of the bits $d^{(i)}$ sent by the receiver to align the shares $s_0^{(i)}$ and $s_1^{(i)}$ with the keys obtained by the receiver in the i-th session. Namely, the share $s_0^{(i)}$ (resp. $s_1^{(i)}$) is encrypted using key $\kappa_{d^{(i)}}^{(i)}$ (resp. $\kappa_{1\oplus d^{(i)}}^{(i)}$) to create the ciphertext $\gamma_{d^{(i)}}^{(i)}$ (resp. $\gamma_{1\oplus d^{(i)}}^{(i)}$).

After checking the defenses from the sender, for each alive session, the receiver decrypts the ciphertext $\gamma_{b^{(i)}}^{(i)}$ using the key $\kappa_{b^{(i)}}^{(i)}$ previously obtained as output in the i-th session, which yields the i-th share $s_b^{(i)}$ of s_b and thus allows to reconstruct s_b using any subset of t shares (where t is the minimum number of shares required for reconstruction in the underlying secret sharing scheme).

We will show how to choose the parameters m, t_R, t_S and t when discussing the simulator below. Note that the receiver does not check if different subsets of shares lead to different secrets, neither it aborts if in some of the alive sessions it retrieves a bogus string. The only other aborting case in the real world would be when the receiver gets less than t "valid keys", and thus shares, which however we bound to happen with negligible probability if the number of parallel sessions and the cut-and-choose parameters are set appropriately.

Simulator for malicious receivers. Let R^* be a malicious receiver. The simulator Sim starts by running R^* and thus receiving the messages $(\mu_R^{(i)})_{i\in[m]}$ that the receiver sends in the first round. Hence, it can perfectly simulate the second round $((\mu_S^{(i)})_{i\in\mathcal{A}}, \mathcal{A})$ of the protocol using pairs of random keys $(\kappa_0^{(i)}, \kappa_1^{(i)})$ for each of the sessions $i \in [m] \setminus \mathcal{A}$ (as the honest sender would do).

Next, R^* replies with $((\delta_R^{(i)})_{i\in\mathcal{A}}, d_{i\in\mathsf{Alive}}^{(i)}, \mathcal{B})$ where $\mathsf{Alive} = [m] \setminus (\mathcal{A} \cup \mathcal{B})$ contains all the indices corresponding to alive sessions. We call the execution up to this point the *main thread.* Now, after checking the defenses from the receiver are good, Sim rewinds R^* and forwards it a freshly sampled second round $((\mu_S^{(i)})_{i\in\mathcal{A}'}, \mathcal{A}')$. This process is repeated until the cut-and-choose sets $\mathcal{A}'$ allows the simulator to obtain good defenses for at least $2/3$ of the the alive sessions in the main thread. The Sim aborts whenever there are more than $m/9$ bad defenses.

A combinatorial analysis shows that setting $m = O(\lambda)$ and $t_R = t_S = m/3$ suffices in order to ensure that: (i) the simulator runs in expected polynomial time, and (ii) the simulator aborts with probability that is negligibly close to the probability that the honest party would have aborted in the real world.

At this point, Sim can extract a bit $b^{(i)}$ for $2/3$ of the alive sessions. This in turn allows to define b as the value $\hat{b}^{(i)} = b^{(i)} \oplus d^{(i)}$ that appears at least $t/2$ times, where $t = 2/3|\mathsf{Alive}| = 2m/9$. Note that at this point the simulator will have received good defenses for at least 2/3 of the alive sessions, otherwise the simulator will have aborted. The simulator forwards b to the OT ideal functionality and completes the simulation with R^* by using the value s_b returned from the functionality, along with a uniformly random string s'_{1-b}.

The rationale behind the above simulation strategy is that whenever a bit b appears more than $t/2$ times the adversary can only learn the value s_b or nothing at all. Note that R^* requires at least t shares to compute s_{1-b}. Out of the $m/3$ sessions in Alive, assume R^* is able to learn both strings for $|\mathsf{Alive}| - t = m/9$ shares. Now if there exist greater than $t/2 = m/9$ shares for the bit b, then there exist at most $m/9 - 1$ shares for s_{1-b}. Thus the adversary is only able to learn at most $m/9 + (m/9 - 1)$ shares of s_{1-b}. This allows the simulator to randomly sample s_{1-b} in the simulation. On the other hand if the adversary plays honestly it learns s_b as in the real-world protocol.

Simulator for malicious senders. The simulator Sim for the case of malicious senders is based on similar ideas. Let S^* be a malicious sender. This time, Sim starts by sampling the first round $(\mu_R^{(i)})_{i\in[m]}$ exactly as the honest receiver would do, upon which S^* replies with $((\mu_S^{(i)})_{i\in[m]\setminus\mathcal{A}}, \mathcal{A})$. Hence, the simulator generates the third round $((\delta_R^{(i)})_{i\in\mathcal{A}}, (d^{(i)})_{i\in\mathsf{Alive}}, \mathcal{B})$ as the honest receiver would do, except that the bits $d^{(i)}$ are picked uniformly at random (as Sim does not know b).

Next, the malicious sender sends the final round $((\delta_S^{(i)})_{i\in\mathcal{B}}, (\gamma_0^{(i)}, \gamma_1^{(i)})_{i\in\mathsf{Alive}})$. Now, after checking the defenses from the sender are good, Sim rewinds S^* and forwards it a freshly sampled third round $((\delta_R^{(i)})_{i\in\mathcal{A}}, (d^{(i)})_{i\in\mathsf{Alive}'}, \mathcal{B}')$ where $\mathsf{Alive}' = [m] \setminus (\mathcal{A} \cup \mathcal{B}')$. This process is repeated until the cut-and-choose sets $\mathcal{B}'$ allows the Sim to obtain defenses $\delta_S^{(i)} = ((\kappa_0^{(i)}, \kappa_1^{(i)}), \rho_S^{(i)})$ for 2/3 of the alive sessions in the main thread. An analysis similar to the case of malicious receivers shows that Sim runs in expected polynomial time and aborts only with negligible probability.

At this point, the simulator can use the keys $(\kappa_0^{(i)}, \kappa_1^{(i)})$ for each index corresponding to an alive session for which S^* showed a good defense, in order to decrypt both ciphertexts $(\gamma_0^{(i)}, \gamma_1^{(i)})$. This allows Sim to extract both s_0 and s_1, after re-aligning the index of the shares consistently with the values $d^{(i)}$ used in the simulation of the main thread. Moreover, the fact that the underlying OT protocol satisfies privacy against malicious senders ensures that the bits $b^{(i)}$ used by the simulator in the alive session of the main thread are indistinguishable from random, and thus so are the values $d^{(i)}$ (as in the simulation).

Defensible privacy and the ROM. It is not hard to see that the proof for the case of malicious receivers actually only requires the underlying OT protocol to satisfy privacy against defensible (rather than malicious) receivers. Intuitively, this is because the sender can check the defenses of the receiver *before* sending the messages that contain its actual input.[6]

While many known constructions of 2-round OT satisfy the stronger property of privacy against malicious receivers, we observe that in the ROM one can obtain privacy against defensible receivers from any OT protocol with semi-honest privacy. The idea is to simply force the receiver to use good randomness by hashing a random string along with the choice bit.

[6] Unfortunately, the opposite is not true as the receiver obtains the defenses from the sender *after* it has already sent the bits $d^{(i)}$. This is the reason why we need to assume privacy against malicious (rather than defensible) senders for the underlying OT protocol. Nevertheless, recall that we also show the latter property comes for free in the case of 2-round protocols.

1.3 Comparison with Friolo et al. [12]

Besides using a very different approach, which leads to a significantly more efficient compiler, the main difference between our compiler and the one by Friolo et al. [12] is in terms of starting assumptions. Our compiler starts from any 2-round privacy-only OT, whereas [12] starts from any 2-round strongly-uniform OT. While is tempting to consider the latter as a much weaker assumption than the former and hence the resulting compiler more general, we notice that the two assumptions are somewhat incomparable. To appreciate the difference, first, consider an edge case where we try to instantiate the two compilers using any 2-round *universally composable* OT protocol. Since universal composability does not necessarily[7] imply strong uniformity, the compiler of [12] would not work. In contrast, our compiler would work as universal composability implies privacy-only.

Also, it is not hard to see that 2-round strongly-uniform OT is not implied by a 2-round semi-honest OT. Indeed, it is easy to come up with a 2-round semi-honest OT that is not strongly uniform. For example, consider the classical 2-round OT construction based on PKE with oblivious key generation instantiated with a contrived PKE where each public key is concatenated with a dummy bit that always equals zero. This PKE scheme still implies a 2-round semi-honest OT, yet the distribution of public keys is far from uniform (and thus the OT protocol is not strongly uniform).

We do acknowledge however that building a 2-round strong uniform semi-honest OT protocol appears to be an easier task than building a 2-round privacy-only OT protocol.

One may wonder whether the compiler of [12] can be simplified if starting with a 2-round privacy-only OT, instead of a strong-uniform OT. For instance, by removing the burden of the commit-and-open protocol on the receiver side. We note that the commit-and-open protocol in [12] is essential to achieve input extraction for both sides, which is required in order to prove simulation-based security. Hence, even when starting with a privacy-only OT protocol, the compiler of [12] cannot forgo the use of commit-and-open (or some other mechanism) for the simulator to extract the inputs.

We refer the reader to the full version[35] for a more detailed comparison between the efficiency of our compiler and the one of [12] in terms of communication complexity. For the sake of concreteness, we also discuss there an explicit instantiation of both compilers in the plain model, based on the hardness of LWE. In this case, we can instantiate our compiler using the 2-round OT protocol by Brakerski and Döttling [5] (which satisfies privacy against malicious, and hence defensible, receivers and semi-honest senders, and thus suffices for our compiler in Theorem 1). For the compiler in [12], instead, we can use the 2-round strongly-uniform OT protocol based on the PKE scheme by Peikert et al. [40]. As we show, this results in a communication complexity of $\tilde{O}(\lambda^3)$ for our compiler, against $\tilde{O}(\lambda^6)$ for the compiler of [12] (where λ is the security parameter).

1.4 Related Work

The compilers in [37, 12, 10] yield the only known round-optimal black-box constructions of simulatable OT in the plain model. In this section, we survey other relevant work on black-box constructions for two-party functionalities. Lindell, Oxman and Pinkas [33], relying on ideas from Ishai, Prabhakaran and Sahai [27], provide a significantly more efficient black-box compiler from semi-honest OT to malicious OT, which however takes at least 8 rounds

---

[7] See, e.g., [11] for a concrete example of a 2-round universally composable OT protocol that is not strongly uniform.

(and thus is not round-optimal). Pass and Wee [38] build commitment and zero-knowledge protocols from black-box access to one-way functions. Hazay and Venkitasubramaniam [24] improve this result by giving a round-optimal construction. More recently, a rich body of work [44, 39, 16] culminated in round-optimal black-box constructions for non-malleable commitments [17, 18], and commit-and-prove [31].

In the CRS model, Choi, Dachman-Soled, Malkin and Wee [8] show a black-box compiler from adaptive semi-honest OT into constant-round adaptive UC-secure two-party computation. Kiyoshima, Lin and Venkitasubramaniam in [32] provide a unified approach to build black-box protocols under trusted setup assumptions, improving on a previous approach by Hazay and Venkitasubramaniam [25]. These works are not round-optimal and are not in the plain model.

2 Preliminaries

Notation. We denote with $\lambda \in \mathbb{N}$ the security parameter. For $n \in \mathbb{N}$, we let $[n] = \{1, \ldots, n\}$. A negligible function, denoted $\mathsf{negl}(\lambda)$, is a function that vanishes faster than the reciprocal of any polynomial $\mathsf{poly}(\lambda)$. We use standard notation for computational/statistical indistinguishability of distribution ensembles.

For an interactive protocol Π between parties A, B holding inputs x, y respectively, we denote the transcript of a protocol execution as $\langle A(x), B(y)\rangle$. Additionally, we let $\mathbf{View}^{B}_{\Pi,A}(\lambda, x, y)$ be the random variable corresponding to the view of A in a run of Π with input x when interacting with B with input y. This view consists of A's input, randomness, and messages received.

Oblivious Transfer. Oblivious transfer (OT) is a two-party protocol Π in which a sender S has two input strings $s_0, s_1 \in \{0,1\}^\lambda$, and a receiver R has a choice bit $b \in \{0,1\}$. An OT protocol is called *non-trivial* if for any pair of strings $s_0, s_1 \in \{0,1\}^\lambda$, and for any $b \in \{0,1\}$, after participating in the interactive protocol, S outputs nothing and R learns s_b. Below, we recall relevant security notions for OT protocols.

Simulatable OT. The standard security definition for OT compares an execution of Π in the real world, where an attacker can corrupt either the sender S or the receiver R, with an execution in the ideal world where a trusted-third party knows all inputs and computes the output on behalf of the players. The corresponding ideal functionality is depicted in Figure 1. In what follows, we denote by $\mathbf{Real}_{\Pi,R^*(z)}(\lambda, s_0, s_1, b)$ (resp., $\mathbf{Real}_{\Pi,S^*(z)}(\lambda, s_0, s_1, b)$) the output of the malicious receiver R^* (resp., sender S^*) during a real execution of the protocol Π (with s_0, s_1 as inputs of the sender, b as choice bit of the receiver, and z as auxiliary input for the adversary), and by $\mathbf{Ideal}_{\mathcal{F}_{\mathsf{OT}},\mathsf{Sim}^{R^*(z)}}(\lambda, s_0, s_1, b)$ (resp., $\mathbf{Ideal}_{\mathcal{F}_{\mathsf{OT}},\mathsf{Sim}^{S^*(z)}}(\lambda, s_0, s_1, b)$) the output of the malicious receiver R^* (resp., sender S^*) in an ideal execution where the parties (with analogous inputs) interact with $\mathcal{F}_{\mathsf{OT}}$, and where the simulator is given black-box access to the adversary.

▶ **Definition 2** (Simulatable OT). *We say that* $\Pi = (S, R)$ securely computes $\mathcal{F}_{\mathsf{OT}}$ *if the following holds:*

- *For every non-uniform PPT malicious receiver R^*, there exists a non-uniform PPT simulator* Sim *such that*

$$\left\{\mathbf{Real}_{\Pi,R^*(z)}(\lambda, s_0, s_1, b)\right\}_{\lambda,s_0,s_1,b,z} \overset{c}{\approx} \left\{\mathbf{Ideal}_{\mathcal{F}_{\mathsf{OT}},\mathsf{Sim}^{R^*(z)}}(\lambda, s_0, s_1, b)\right\}_{\lambda,s_0,s_1,b,z},$$

where $\lambda \in \mathbb{N}, s_0, s_1 \in \{0,1\}^\lambda, b \in \{0,1\}$*, and* $z \in \{0,1\}^*$.

Ideal Functionality $\mathcal{F}_{\mathsf{OT}}$:

- Upon receiving message $(\texttt{send}, s_0, s_1, S, R)$ from S, where $s_0, s_1 \in \{0,1\}^\lambda$, store s_0, s_1 and answer $\texttt{send}$ to R and Sim.
- Upon receiving message $(\texttt{receive}, b)$ from R, where $b \in \{0,1\}$, send s_b to R and $\texttt{receive}$ to S and Sim, and halt. If no message $(\texttt{send}, \cdot)$ was previously sent, do nothing.

Figure 1 Ideal functionality for oblivious transfer.

- *For every non-uniform PPT malicious sender S^*, there exists a non-uniform PPT simulator* Sim *such that*

$$\left\{\mathbf{Real}_{\Pi,S^*(z)}(\lambda, s_0, s_1, b)\right\}_{\lambda,s_0,s_1,b,z} \overset{c}{\approx} \left\{\mathbf{Ideal}_{\mathcal{F}_{\mathsf{OT}},\mathsf{Sim}^{S^*(z)}}(\lambda, s_0, s_1, b)\right\}_{\lambda,s_0,s_1,b,z},$$

where $\lambda \in \mathbb{N}, s_0, s_1 \in \{0,1\}^\lambda, b \in \{0,1\}$, and $z \in \{0,1\}^*$.

Privacy-Only OT. A weaker guarantee than simulatable security is the so-called *privacy* property, which does not require the existence of a simulator. Roughly, privacy for the receiver means that the choice bit is computationally hidden, whereas privacy for the sender means that the receiver can obtain at most one input from the sender. Below, we formalize this notion for different adversarial behaviours and assuming that the input of the players are uniformly random (which will suffice for our purpose).

Semi-honest privacy. The definition below formalizes privacy in the semi-honest setting (i.e., when corrupted parties do not deviate from the protocol).

▶ **Definition 3** (Semi-honest privacy for random inputs). *Let $\Pi = (S, R)$ be a non-trivial OT protocol. We say that $\Pi = (S, R)$ is* private for random inputs against semi-honest receivers *if the following holds:*

$$\left\{\mathbf{View}^{S}_{\Pi,R}(\lambda, s_0, s_1, b), s_{1-b}\right\}_{\lambda,b} \overset{c}{\approx} \left\{\mathbf{View}^{S}_{\Pi,R}(\lambda, s_0, s_1, b), s'\right\}_{\lambda,b}$$

where $\lambda \in \mathbb{N}$, $b \in \{0,1\}$, and $s_0, s_1, s' \leftarrow\!\!\$\ \{0,1\}^\lambda$. Similarly, Π is private for random inputs against semi-honest senders *if the following holds:*

$$\left\{\mathbf{View}^{R}_{\Pi,S}(\lambda, s_0, s_1, b), b\right\}_{\lambda,s_0,s_1} \overset{c}{\approx} \left\{\mathbf{View}^{R}_{\Pi,S}(\lambda, s_0, s_1, b), b'\right\}_{\lambda,s_0,s_1}$$

where $\lambda \in \mathbb{N}$, $s_0, s_1 \in \{0,1\}^\lambda$, and $b, b' \leftarrow\!\!\$\ \{0,1\}$.

Malicious privacy. The definition below (adapted from [23]) formalizes privacy in the malicious setting (i.e., when corrupted parties can arbitrarily deviate from the protocol).

▶ **Definition 4** (Malicious privacy). *Let $\Pi = (S, R)$ be a non-trivial OT protocol.*

Π is private for random inputs against malicious senders *if for every non-uniform PPT malicious sender S^*:*

$$\left\{\mathbf{View}^{R}_{\Pi,S^*(z)}(\lambda, s_0, s_1, b), b\right\}_{\lambda,s_0,s_1,z} \approx_c \left\{\mathbf{View}^{R}_{\Pi,S^*(z)}(\lambda, s_0, s_1, b), b'\right\}_{\lambda,s_0,s_1,z}$$

where $\lambda \in \mathbb{N}$, $s_0, s_1 \in \{0,1\}^\lambda$, $z \in \{0,1\}^*$, and $b, b' \leftarrow\!\!\$\ \{0,1\}$.

Defensible privacy. Following Haitner et al. [19, 20], we call *defense* by R^* an input $b \in \{0,1\}$ and a random tape $\rho_R \in \{0,1\}^*$ provided by the receiver at the end of the protocol. Intuitively, a defense is *good* if the honest receiver using this very input and randomness would have sent the exact same messages as the malicious receiver sent. A similar notion can be considered for the sender, where defenses are of the form (s_0, s_1, ρ_S) with $s_0, s_1 \in \{0,1\}^\lambda$ and $\rho_S \in \{0,1\}^*$.

The definition below formalizes the concept of good defense in the special case of 2-round OT protocols, which we model as follows. Let OTR a PPT algorithm taking as input the choice bit $b \in \{0,1\}$ and the random tape $\rho_R \in \{0,1\}^*$ for the receiver, and outputting a message $\mu_R \in \{0,1\}^*$ for the sender; similarly, let OTS be a PPT algorithm taking as input the strings $s_0, s_1 \in \{0,1\}^\lambda$ and the random tape $\rho_S \in \{0,1\}^*$ for the sender, as well as a message $\mu_R \in \{0,1\}^*$ from the receiver, and outputting a message $\mu_S \in \{0,1\}^*$ for the receiver. Finally, let OTD be a PPT algorithm taking as input the choice bit $b \in \{0,1\}$ and the random tape $\rho_R \in \{0,1\}^*$ for the receiver, as well as message μ_S from the sender, and outputting a value s in $\{0,1\}^\lambda$.

▶ **Definition 5** (Good defense for 2-round OT). *Let $\Pi = (\mathsf{OTR}, \mathsf{OTS}, \mathsf{OTD})$ be a 2-round OT protocol. Fix any transcript $\tau = (\mu_R, \mu_S) \in (\{0,1\}^*)^2$ for Π. We say that the pair $\delta_R = (b, \rho_R)$ (resp. $\delta_S = (s_0, s_1, \rho_S)$), where $b \in \{0,1\}$, constitutes a* good defense *by the receiver (resp. by the sender) for τ in Π if it holds that $\mu_R = \mathsf{OTR}(b; \rho_R)$ (resp. $\mu_S = \mathsf{OTS}(s_0, s_1, \mu_R; \rho_S)$).*

Loosely speaking, an OT protocol has defensible privacy if the privacy property holds against malicious adversaries that can provide a good defense.

▶ **Definition 6** (Defensible privacy for random inputs). *Let $\Pi = (S, R)$ be a non-trivial OT protocol. We say that Π is* private for random inputs against defensible receivers *if for every non-uniform PPT malicious receiver R^*:*

$$\left\{\Gamma\left(\mathbf{View}^S_{\Pi, R^*(z)}(\lambda, (s_0, s_1), b), s_{1-b}\right)\right\}_{\lambda, b, z} \approx_c \left\{\Gamma\left(\mathbf{View}^S_{\Pi, R^*(z)}(\lambda, s_0, s_1, b), s'\right)\right\}_{\lambda, b, z}$$

where $\lambda \in \mathbb{N}$, $b \in \{0,1\}$, $z \in \{0,1\}^*$, $s_0, s_1, s' \leftarrow\$ \{0,1\}^\lambda$, and $\Gamma(v, *)$ is set to $(v, *)$ if following the execution R^* outputs a good defense (and to $\perp$ otherwise). Similarly, Π is private for random inputs against defensible senders *if for every non-uniform PPT malicious sender S^*:*

$$\left\{\Gamma\left(\mathbf{View}^R_{\Pi, S^*(z)}(\lambda, (s_0, s_1), b), b\right)\right\}_{\lambda, s_0, s_1, z} \approx_c \left\{\Gamma\left(\mathbf{View}^R_{\Pi, S^*(z)}(\lambda, s_0, s_1, b), b'\right)\right\}_{\lambda, s_0, s_1, z}$$

where $\lambda \in \mathbb{N}$, $s_0, s_1 \in \{0,1\}^\lambda$, $z \in \{0,1\}^*$, $b, b' \leftarrow\$ \{0,1\}$, and $\Gamma(v, *)$ is set to $(v, *)$ if following the execution S^* outputs a good defense (and to $\perp$ otherwise).

Secret Sharing. A threshold secret sharing scheme allows to share an input string $s \in \{0,1\}^\lambda$ into n shares $s_1, \ldots, s_n \in \{0,1\}^\lambda$ in such a way that it is possible to efficiently recover s from any subset of at least t shares, while at the same time an attacker corrupting up to $t-1$ share holders obtains no information about the secret.

▶ **Definition 7** (Secret sharing). *An (n, t)-*secret sharing scheme *over $\{0,1\}^\lambda$ is defined by a pair of algorithms $(\mathsf{Share}, \mathsf{Recon})$, where Share is a randomized mapping of an input $s \in \{0,1\}^\lambda$ to shares $s = (s_1, s_2, \ldots, s_n) \in (\{0,1\}^\lambda)^n$, and Recon is a function mapping a subset $\mathcal{I}$ of $[n]$, along with the corresponding shares $s_\mathcal{I} = (s_i)_{i \in \mathcal{I}}$, to a value in $\{0,1\}^\lambda$, such that the following holds:*

1. **Reconstruction.** *For all $s \in \{0,1\}^\lambda$, and for all sets $\mathcal{I} \subseteq [n]$ with $|\mathcal{I}| \geq t$, the output of $\mathsf{Recon}(\mathcal{I}, s_\mathcal{I})$ such that $(s_1, \ldots, s_n) \leftarrow\$ \mathsf{Share}(s)$ is equal to s.*

2. **Security.** *For all $s \in \{0,1\}^\lambda$, and for all sets $\mathcal{I} \subseteq [n]$ with $|\mathcal{I}| < t$, the joint distribution $s_\mathcal{I} = (s_i)_{i \in \mathcal{I}}$ of shares received by the subset of parties $\mathcal{I}$, where $(s_1, \ldots, s_n) \leftarrow\$ \mathsf{Share}(s)$, is independent of the secret s.*

We call a share *valid* if it is a λ-bit string. For our construction, we will implicitly assume that running algorithm Recon upon input any sequence of t valid shares (possibly outside the support of Share) still yields a λ-bit message. Note that, e.g., Shamir's secret sharing [42] satisfies this property.

3 Observations on Two-Round OT

Here, we collect a few observations on 2-round OT protocols. First, in Section 3.1, we show that any 2-round OT protocol with privacy against semi-honest senders is already private against malicious senders. Next, in Section 3.2, we overview existing 2-round OT protocols that satisfy the notion of privacy against defensible receivers. Furthermore, we show a simple compiler in the ROM that adds the latter property to any 2-round OT protocol with semi-honest privacy only.

3.1 Privacy against Malicious Senders

The lemma below states that any 2-round OT protocol with privacy against semi-honest senders is already private against malicious senders.[8] Intuitively, this is the case since in a 2-round OT protocol the only thing a sender can do is to respond to the first message sent by the receiver, however, this does not help to learn the choice bit of the receiver. While we prove the lemma for the case of privacy with random inputs, a similar statement holds for any distribution of the receiver's choice bit.

▶ **Lemma 8.** *Any 2-round OT protocol that is private (for random inputs) against semi-honest senders is also private (for random inputs) against malicious senders.*

We defer the proof of this lemma to the full version [35].

3.2 Privacy against Defensible Receivers

Two-round OT protocols (for random inputs) with privacy against defensible receivers exist under standard number-theoretic assumptions, including DDH [36, 1], QR and DCR [21], and LWE [5]. In fact, these protocols satisfy the stronger property of privacy against *malicious* receivers.

In this section, we present a round-preserving transform turning any 2-round OT protocol with semi-honest privacy into one with privacy against defensible receivers in the ROM. Intuitively, our transform ensures that the receiver cannot influence the randomness used to generate the first OT message. This is achieved by hashing the choice bit b along with a random string ρ_R chosen by the receiver. We refer the reader to the full version [35] for a formal proof of the statement below.

▶ **Lemma 9.** *If there exists a 2-round OT protocol with privacy (for random inputs) against semi-honest senders and receivers, then there exists a 2-round OT protocol with privacy (for random inputs) against defensible receivers and semi-honest senders in the ROM.*

[8] A similar observation appears in [12, p. 12].

Protocol $\Pi = (S, R)$

Sender's Input: $s_0, s_1 \in \{0,1\}^\lambda$ Receiver's Input: $b \in \{0,1\}$

1. $(R \to S)$: For each $i \in [m]$, the receiver picks $b^{(i)} \leftarrow\$ \{0,1\}$ and $\rho_R^{(i)} \leftarrow\$ \{0,1\}^*$, lets $\mu_R^{(i)} = \mathsf{OTR}(b^{(i)}; \rho_R^{(i)})$, and sends $(\mu_R^{(i)})_{i\in[m]}$ to the sender.
2. $(S \to R)$: The sender's first message is computed as follows.
   a. Sample $\alpha_1, \dots, \alpha_{t_R} \leftarrow\$ [m]$ and let $\mathcal{A} = \{\alpha_1, \dots, \alpha_{t_R}\}$.
   b. For each $i \in [m] \setminus \mathcal{A}$, pick $\kappa_0^{(i)}, \kappa_1^{(i)} \in \{0,1\}^\lambda$ and $\rho_S^{(i)} \leftarrow\$ \{0,1\}^*$, and let $\mu_S^{(i)} = \mathsf{OTS}(\kappa_0^{(i)}, \kappa_1^{(i)}, \mu_R^{(i)}; \rho_S^{(i)})$.

   Send $(\mu_S^{(i)})_{i\in[m]\setminus\mathcal{A}}$ and $\mathcal{A}$ to the receiver.
3. $(R \to S)$: The receiver's second message is computed as follows.
   a. For each $i \in \mathcal{A}$, let $\delta_R^{(i)} = (\rho_R^{(i)}, b^{(i)})$.
   b. Sample $\beta_1, \dots, \beta_{t_S} \leftarrow\$ [m] \setminus \mathcal{A}$ and let $\mathcal{B} = \{\beta_1, \dots, \beta_{t_S}\}$.
   c. Let $\mathsf{Alive} = [m] \setminus (\mathcal{A} \cup \mathcal{B})$. For each $i \in \mathsf{Alive}$, let $\kappa_{b^{(i)}}^{(i)} = \mathsf{OTD}(b^{(i)}, \mu_S^{(i)}; \rho_R^{(i)})$ and define $d^{(i)} = b^{(i)} \oplus b$.

   Send $(\delta_R^{(i)})_{i\in\mathcal{A}}$ and $(d^{(i)})_{i\in\mathsf{Alive}}$, along with the set $\mathcal{B}$, to the sender.
4. $(S \to R)$: The sender's last message is computed as follows.
   a. For each $i \in \mathcal{A}$, check that $\delta_R^{(i)}$ is good. If not, abort.
   b. For each $i \in \mathcal{B}$, let $\delta_S^{(i)} = (\kappa_0^{(i)}, \kappa_1^{(i)}, \rho_S^{(i)})$.
   c. Let $|\mathsf{Alive}| = n = m - t_R - t_S$. Run $(s_0^{(i)})_{i\in\mathsf{Alive}} \leftarrow\$ \mathsf{Share}(s_0)$ and $(s_1^{(i)})_{i\in\mathsf{Alive}} \leftarrow\$ \mathsf{Share}(s_1)$ (i.e., secret share s_0 and s_1 and assign one share to each of the alive sessions).
   d. For each $i \in \mathsf{Alive}$, let $\gamma_{0\oplus d^{(i)}}^{(i)} = \kappa_{d^{(i)}}^{(i)} \oplus s_0^{(i)}$ and $\gamma_{1\oplus d^{(i)}}^{(i)} = \kappa_{1\oplus d^{(i)}}^{(i)} \oplus s_1^{(i)}$. (Note that this implies $\gamma_{b\oplus d^{(i)}}^{(i)} = \kappa_{b\oplus d^{(i)}}^{(i)} \oplus s_b^{(i)}$ for all $b \in \{0,1\}$.)

   Send $(\delta_S^{(i)})_{i\in\mathcal{B}}$ and $(\gamma_0^{(i)}, \gamma_1^{(i)})_{i\in\mathsf{Alive}}$ to the receiver.
5. (Receiver's Output:) The receiver determines the final output as follows. For each $i \in \mathcal{B}$, check that $\delta_S^{(i)}$ is good. If not, abort. Else, for each $i \in \mathsf{Alive}$, let $s_b^{(i)} = \gamma_{b^{(i)}}^{(i)} \oplus \kappa_{b^{(i)}}^{(i)}$, pick any subset $\mathcal{I}$ of $(s_b^{(i)})_{i\in\mathsf{Alive}}$ containing at least t *valid* shares, and return $s_b = \mathsf{Recon}((s_b^{(i)})_{i\in\mathcal{I}})$.

Figure 2 Formal description of our black-box compiler.

4 Our Compiler

In this section, we show a black-box compiler for obtaining round-optimal simulatable OT starting from any 2-round OT satisfying privacy for random inputs against defensible receivers (cf. Definition 6) and malicious senders (cf. Definition 4). Recall that, by Lemma 8, the latter property follows from privacy for random inputs against semi-honest senders.

4.1 Protocol Description

Let $\Pi' = (\mathsf{OTR}, \mathsf{OTS}, \mathsf{OTD})$ be a 2-round OT protocol. We transform Π' into a 4-round OT protocol Π as depicted in Figure 2. Intuitively, the protocol Π proceeds as follows:

Round 1: The receiver starts m parallel sessions of the underlying OT protocol Π'. In each session $i \in [m]$, it uses a uniformly random choice bit $b^{(i)}$, which yields a message $\mu_R^{(i)}$. Hence, it forwards $\mu_R^{(1)}, \dots, \mu_R^{(m)}$ to the sender.

Round 2: The sender picks random indices $\alpha_1, \ldots, \alpha_{t_R} \in [m]$ for cut-and-choose, where $t_R = m/3$. For the remaining sessions, it picks two random strings $(\kappa_0^{(i)}, \kappa_1^{(i)})$ and computes $\mu_S^{(i)}$ as a response to the message $\mu_R^{(i)}$ using the underlying OT protocol. Looking ahead, these random strings will serve as masks to hide the input messages in the last round of the protocol. Hence, it forwards the indices $\alpha_1, \ldots, \alpha_{t_R}$ and the messages $\mu_S^{(i)}$ (for all sessions but the ones selected for cut-and-choose).

Round 3: For each session that the sender asked to open, the receiver prepares a defense $\delta_R^{(i)} = (b^{(i)}, \rho_R^{(i)})$ which explains the message $\mu_R^{(i)}$ that was sent in the first round. Then, it picks random indices $\beta_1, \ldots, \beta_{t_S} \in [m]\backslash\{\alpha_1, \ldots, \alpha_{t_R}\}$ for cut-and-choose, where $t_S = m/3$; looking ahead, in the next round the sender will have to provide defenses for those indices, which allows to extract the inputs of the sender in the simulation proof. Denote by Alive the set of indices corresponding to sessions which are still alive (i.e., that were not selected for cut-and-choose). Using the underlying OT protocol, the receiver obtains the mask $\kappa_{b^{(i)}}^{(i)}$ and forwards to the sender an adjusting bit $d^{(i)} = b^{(i)} \oplus b$ for each alive session; intuitively, the bit $d^{(i)}$ tells the sender how to encrypt the input strings in the last round.

Round 4: The sender first checks that each of the defenses $\delta_R^{(i)}$ are good (and aborts if not). The privacy property (against defensible receivers) of the underlying OT protocol guarantees that the receiver does not learn the masks $\kappa_{1-b^{(i)}}^{(i)}$ corresponding to these sessions. Additionally, the sender prepares its own defenses $\delta_S^{(i)} = (\kappa_0^{(i)}, \kappa_1^{(i)}, \rho_S^{(i)})$ for each index $i \in \mathcal{B}$. Hence, it secret shares the input strings s_0 and s_1, obtaining $n = m - t_R - t_S$ shares $(s_0^{(i)})_{i\in\mathsf{Alive}}$ and $(s_1^{(i)})_{i\in\mathsf{Alive}}$, so that each pair of shares can be associated to a single alive session. Finally, the sender uses the key $\kappa_{0\oplus d^{(i)}}^{(i)}$ (resp. $\kappa_{1\oplus d^{(i)}}^{(i)}$) in order to encrypt the share $s_0^{(i)}$ (resp. $s_1^{(i)}$) in the i-th session, and forwards the resulting pairs of ciphertexts $(\gamma_0^{(i)}, \gamma_1^{(i)})_{i\in\mathsf{Alive}}$ to the receiver.

Output Computation: Since $b^{(i)} = b \oplus d^{(i)}$, the receiver can use the keys $\kappa_{b^{(i)}}^{(i)}$ in order to obtain all the shares $(s_b^{(i)})_{i\in\mathsf{Alive}}$ and thus reconstruct s_b using any subset of t *valid* shares[9] (and abort if there are less than t valid shares).

The theorem below states the security of our compiler.

▶ **Theorem 10.** *Let $\Pi' = (\mathsf{OTR}, \mathsf{OTS}, \mathsf{OTD})$ be a 2-round OT protocol with privacy for random inputs against defensible receivers and against malicious senders, and let $(\mathsf{Share}, \mathsf{Recon})$ be a t-out-of-n secret sharing scheme. Then, for parameters m, t_R, t_S, t, n such that $m = O(\lambda)$, $t_R = t_S = n = m/3$ and $t = 2n/3$, the protocol $\Pi = (S, R)$ from Figure 2 securely realizes $\mathcal{F}_{\mathsf{OT}}$.*

We defer an overview of the proofs to the appendix and a more formal treatment to the full version[35].

5 Conclusions

We have shown a compiler for turning any 2-round *privacy-only* (i.e., game-based) OT protocol against *malicious* adversaries into a *round-optimal simulatable* OT protocol against *malicious* adversaries. Our transform is *black-box* (in that it only uses the underlying 2-round OT protocol as an oracle), *information-theoretic* (in that it does not rely on cryptographic assumptions), and in the *plain model* (in that it does not require any form of trusted setup). In fact, our compiler works even assuming the underlying 2-round OT protocol satisfies privacy against *semi-honest* senders and *defensible* receivers.

[9] Recall that a share is called valid if it is a λ-bit string.

It remains an open problem to find a similar transform (with the same properties) starting with any 2-round OT protocol satisfying only privacy against *semi-honest* (rather than *defensible*) receivers. Also, it would be interesting to find simple constructions of 2-round OT protocols in the plain model that are private against *defensible* receivers, and that rely on hardness assumptions from which we do not know how to obtain 2-round OT protocols that are private against *malicious* receivers (e.g., CDH, LPN, and Subset Sum).

References

1 William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 119–135. Springer, Heidelberg, May 2001. `doi:10.1007/3-540-44987-6_8`.

2 Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and applications. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 547–557. Springer, Heidelberg, August 1990. `doi:10.1007/0-387-34805-0_48`.

3 Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. `doi:10.1145/168588.168596`.

4 Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 500–532. Springer, Heidelberg, April / May 2018. `doi:10.1007/978-3-319-78375-8_17`.

5 Zvika Brakerski and Nico Döttling. Two-message statistically sender-private OT from LWE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 370–390. Springer, Heidelberg, November 2018. `doi:10.1007/978-3-030-03810-6_14`.

6 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. URL: `https://eprint.iacr.org/2017/1165`.

7 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*, pages 597–608. ACM Press, November 2014. `doi:10.1145/2660267.2660374`.

8 Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Simple, black-box constructions of adaptively secure protocols. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 387–402. Springer, Heidelberg, March 2009. `doi:10.1007/978-3-642-00457-5_23`.

9 Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 73–88. Springer, Heidelberg, February / March 2013. `doi:10.1007/978-3-642-36362-7_6`.

10 Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Oblivious transfer from trapdoor permutations in minimal rounds. In *Theory of Cryptography Conference*, pages 518–549. Springer, 2021.

11 Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, Daniel Masny, and Daniel Wichs. Two-round oblivious transfer from CDH or LPN. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 768–797. Springer, Heidelberg, May 2020. `doi:10.1007/978-3-030-45724-2_26`.

12 Daniele Friolo, Daniel Masny, and Daniele Venturi. A black-box construction of fully-simulatable, round-optimal oblivious transfer from strongly uniform key agreement. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 111–130. Springer, Heidelberg, December 2019. `doi:10.1007/978-3-030-36030-6_5`.

13 Juan A. Garay. Efficient and universally composable committed oblivious transfer and applications. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 297–316. Springer, Heidelberg, February 2004. `doi:10.1007/978-3-540-24638-1_17`.

14 Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 468–499. Springer, Heidelberg, April / May 2018. `doi:10.1007/978-3-319-78375-8_16`.

15 Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st FOCS*, pages 325–335. IEEE Computer Society Press, November 2000. `doi:10.1109/SFCS.2000.892121`.

16 Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *53rd FOCS*, pages 51–60. IEEE Computer Society Press, October 2012. `doi:10.1109/FOCS.2012.47`.

17 Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 1128–1141. ACM Press, June 2016. `doi:10.1145/2897518.2897657`.

18 Vipul Goyal and Silas Richelson. Non-malleable commitments using Goldreich-Levin list decoding. In David Zuckerman, editor, *60th FOCS*, pages 686–699. IEEE Computer Society Press, November 2019. `doi:10.1109/FOCS.2019.00047`.

19 Iftach Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 412–426. Springer, Heidelberg, March 2008. `doi:10.1007/978-3-540-78524-8_23`.

20 Iftach Haitner, Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions of protocols for secure computation. *SIAM Journal on Computing*, 40(2):225–266, 2011.

21 Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *Journal of Cryptology*, 25(1):158–193, January 2012. `doi:10.1007/s00145-010-9092-8`.

22 Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. ISC. Springer, Heidelberg, 2010. `doi:10.1007/978-3-642-14303-8`.

23 Carmit Hazay and Yehuda Lindell. *Efficient secure two-party protocols: Techniques and constructions*. Springer Science & Business Media, 2010.

24 Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. Round-optimal fully black-box zero-knowledge arguments from one-way permutations. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 263–285. Springer, Heidelberg, November 2018. `doi:10.1007/978-3-030-03807-6_10`.

25 Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. On black-box complexity of universally composable security in the CRS model. *Journal of Cryptology*, 32(3):635–689, July 2019. `doi:10.1007/s00145-019-09326-y`.

26 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 406–425. Springer, Heidelberg, May 2011. `doi:10.1007/978-3-642-20465-4_23`.

27 Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Heidelberg, August 2008. `doi:10.1007/978-3-540-85174-5_32`.

28 Stanislaw Jarecki and Vitaly Shmatikov. Efficient two-party secure computation on committed inputs. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 97–114. Springer, Heidelberg, May 2007. `doi:10.1007/978-3-540-72540-4_6`.

29 Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 335–354. Springer, Heidelberg, August 2004. `doi:10.1007/978-3-540-28628-8_21`.

30 Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992. doi:10.1145/129712.129782.

31 Susumu Kiyoshima. Round-optimal black-box commit-and-prove with succinct communication. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 533–561. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56880-1_19.

32 Susumu Kiyoshima, Huijia Lin, and Muthuramakrishnan Venkitasubramaniam. A unified approach to constructing black-box UC protocols in trusted setup models. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 776–809. Springer, Heidelberg, November 2017. doi:10.1007/978-3-319-70500-2_26.

33 Yehuda Lindell, Eli Oxman, and Benny Pinkas. The IPS compiler: Optimizations, variants and concrete efficiency. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 259–276. Springer, Heidelberg, August 2011. doi:10.1007/978-3-642-22792-9_15.

34 Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *Journal of Cryptology*, 25(4):680–722, October 2012. doi:10.1007/s00145-011-9107-0.

35 Varun Madathil, Chris Orsini, Alessandra Scafuro, and Daniele Venturi. From privacy-only to simulatable ot: Black-box, round-optimal, information-theoretic. *Cryptology ePrint Archive*, 2022.

36 Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *12th SODA*, pages 448–457. ACM-SIAM, January 2001.

37 Rafail Ostrovsky, Silas Richelson, and Alessandra Scafuro. Round-optimal black-box two-party computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 339–358. Springer, Heidelberg, August 2015. doi:10.1007/978-3-662-48000-7_17.

38 Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 403–418. Springer, Heidelberg, March 2009. doi:10.1007/978-3-642-00457-5_24.

39 Rafael Pass and Hoeteck Wee. Constant-round non-malleable commitments from sub-exponential one-way functions. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 638–655. Springer, Heidelberg, May / June 2010. doi:10.1007/978-3-642-13190-5_32.

40 Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2008. doi:10.1007/978-3-540-85174-5_31.

41 Michael O. Rabin. How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187, 2005. URL: http://eprint.iacr.org/2005/187.

42 Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

43 Wolfgang Stadje. The collector's problem with group drawings. *Advances in Applied Probability*, 22(4):866–882, 1990. doi:10.2307/1427566.

44 Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *51st FOCS*, pages 531–540. IEEE Computer Society Press, October 2010. doi:10.1109/FOCS.2010.87.

A Simulator for Malicious Receivers

The simulator Sim, with oracle access to R^*, is defined in Fig 3. The definition below is useful to reason about the simulator.

▶ **Definition 11.** *Let* $\mathsf{Bad} \subset \mathsf{Alive}$ *be the set of indices for which the simulator does not get a defense.*

Main thread:

1. Upon receiving $\mu_R^{(1)}, \ldots, \mu_R^{(m)}$ from R^*:
   a. Sample indices $\alpha_1, \ldots, \alpha_{t_R} \leftarrow\$ [m]$ and let $\mathcal{A} = \{\alpha_1, \ldots, \alpha_{t_R}\}$.
   b. For each $i \in [m] \setminus \mathcal{A}$, pick $\kappa_0^{(i)}, \kappa_1^{(i)} \in \{0,1\}^\lambda$ and $\rho_S^{(i)} \leftarrow\$ \{0,1\}^*$, and let $\mu_S^{(i)} = \mathsf{OTS}(\kappa_0^{(i)}, \kappa_1^{(i)}, \mu_R^{(i)}; \rho_S^{(i)})$.
   c. Send $\mu_S^{(i)}$ (for each $i \in [m] \setminus \mathcal{A}$) and $\mathcal{A}$ to R^*.
2. Upon receiving $((\delta_R^{(i)})_{i\in\mathcal{A}}, (d^{(i)})_{i\in\mathsf{Alive}}, \mathcal{B})$ from R^* (where $\mathsf{Alive} = [m] \setminus \mathcal{A} \cup \mathcal{B}$), check that all defenses $\delta_R^{(i)} = (b^{(i)}, \rho_R^{(i)})$ are good. If not, simulate the sender aborting. Else initialize $\mathsf{Bits} = \emptyset$ and $\mathsf{ctr} = 0$

   Rewind thread:

   a. Sample $\mathcal{A}' = \{\alpha'_1, \ldots, \alpha'_{t_R}\}$ randomly as in Item 1a.
   b. Recompute $\mu_S^{(i)}$ using fresh randomness (for each $i \in [m]\setminus\mathcal{A}'$), send these messages along with $\mathcal{A}'$ to R^* and receive $((\delta_R^{(i)})_{i\in\mathcal{A}'}, \mathcal{B}', (d^{(i)})_{i\in\mathsf{Alive}'})$ in response, where $\mathsf{Alive}' = [m] \setminus (\mathcal{A}' \cup \mathcal{B}')$.
   c. For every defense $\delta_R^{(i)} = (b^{(i)}, \rho_R^{(i)})$ corresponding to an index $i \in \mathsf{Alive}$ (from the main thread) that was not observed in a previous rewind: If the defense is good add the bit $b^{(i)}$ to the set Bits.
   d. Increment $\mathsf{ctr} = \mathsf{ctr} + 1$. If $\mathsf{ctr} = 2^\lambda$, abort.
   e. If $|\mathsf{Bits}| + m/9 < |\mathsf{Alive}|$ go to Item 2a; else proceed to the next step.

3. Complete the simulation in the main thread as follows.
   a. For each $b^{(i)} \in \mathsf{Bits}$, compute $\hat{b}^{(i)} = b^{(i)} \oplus d^{(i)}$.
   b. Let b be a bit that appears among the $\hat{b}^{(i)}$ more than $t/2$ times
   c. Forward b to $\mathcal{F}_{\mathsf{OT}}$ obtaining $s_b \in \{0,1\}^\lambda$, sample $s'_{1-b} \leftarrow\$ \{0,1\}^\lambda$ and simulate the final message $((\delta_S^{(i)})_{i\in\mathcal{B}}, (\gamma_0^{(i)}, \gamma_1^{(i)})_{i\in\mathsf{Alive}})$ as the honest sender would do.

Figure 3 Simulator against a malicious receiver.

▶ **Lemma 12.** *Conditioned on the fact that* Sim *did not abort in the main thread, the cardinality of* Bad *(Definition 11) is strictly less than $m/9$ with overwhelming probability.*

Proof. We defer this proof to the full version [35]. ◀

▶ **Lemma 13.** *The above simulator* Sim *runs in expected time that is polynomial in m and λ except with negligible probability.*

Proof. The number of rewind attempts needed to cover all indices of $\mathsf{Alive} \setminus \mathsf{Bad}$ is a variation of the coupon collector's problem [43] which has a polynomial time solution. We defer proofs to the full version [35]. ◀

Proof by hybrids. We next prove by a sequence of hybrids that the distribution of the output of R^* in the real world is computationally close to that in the ideal world with the above defined simulator. The hybrids are described[10] below:

Hybrid $\mathbf{Hyb}_0(\lambda, s_0, s_1, b)$: This is identical to $\mathbf{Real}_{\Pi,R^*(z)}(\lambda, s_0, s_1, b)$.

Hybrid $\mathbf{Hyb}_1(\lambda, s_0, s_1, b)$: Identical to the previous experiment, except that we now perform the rewinding as done by the simulator before concluding the protocol.

Hybrid $\mathbf{Hyb}_2(\lambda, s_0, s_1, b)$: Identical to the previous experiment except when generating the message $((\delta_S^{(i)})_{i\in\mathcal{B}}, (\gamma_0^{(i)}, \gamma_1^{(i)})_{i\in\mathsf{Alive}})$ redefine $\gamma^{(i)}_{1-b^{(i)}} = \hat{\kappa}_{1-b^{(i)}} \oplus s^{(i)}_{1-\hat{b}^{(i)}}$ using an independent $\hat{\kappa}_{1-b^{(i)}} \leftarrow\!\!\$ \{0,1\}^\lambda$ (instead of $\kappa^{(i)}_{1-b^{(i)}}$) for each bit $b^{(i)} \in \mathsf{Bits}$ (recall that $\hat{b}^{(i)} = b^{(i)} \oplus d^{(i)}$).

Hybrid $\mathbf{Hyb}_3(\lambda, s_0, s_1, b)$: Identical to the previous experiment except for the following difference. After successfully completing the rewinding proceed as follows: For each $b^{(i)} \in \mathsf{Bits}$, compute $\hat{b}^{(i)} = b^{(i)} \oplus d^{(i)}$. Let b be a bit that appears among the $\hat{b}^{(i)}$ more than $t/2$ times. Forward b to $\mathcal{F}_{\mathsf{OT}}$ obtaining $s_b \in \{0,1\}^\lambda$, sample $s'_{1-b} \leftarrow\!\!\$ \{0,1\}^\lambda$ and simulate the final message $((\delta_S^{(i)})_{i\in\mathcal{B}}, (\gamma_0^{(i)}, \gamma_1^{(i)})_{i\in\mathsf{Alive}})$ as the honest sender would do.

Since the final hybrid is identically distributed to the ideal world with the above defined simulator Sim, it remains to show that the above hybrids are all computationally indistinguishable. In the full version[35] we show that the hybrids are indistinguishable.

B Simulator for Malicious Senders

The simulator Sim, with oracle access to S^*, proceeds as follows in Figure 4.

▶ **Lemma 14.** *The above simulator* Sim *runs in expected time that is polynomial in* (m, λ).

Proof. The analysis is similar to that of Lemma 13 and we defer a more formal treatment to the full version [35]. ◀

Proof by hybrids. We next prove by a sequence of hybrids that the distribution of the output of S^* in the real world is computationally close to that in the ideal world with the above defined simulator. The hybrids are described below:

Hybrid $\mathbf{Hyb}_0(\lambda, s_0, s_1, b)$: This is identical to $\mathbf{Real}_{\Pi,S^*(z)}(\lambda, s_0, s_1, b)$.

Hybrid $\mathbf{Hyb}_1(\lambda, s_0, s_1, b)$: Identical to the previous experiment, except that we now perform the rewinding as done by the simulator before concluding the protocol.

Hybrid $\mathbf{Hyb}_2(\lambda, s_0, s_1, b)$: Identical to the previous experiment except that the $d^{(i)}$ are sampled randomly and for each$i \in [m] \setminus (\mathcal{A} \cup \mathcal{B})$ compute $\kappa^{(i)}_{b^{(i)}} = \mathsf{OTD}(b^{(i)}, \mu_S^{(i)}; \rho_R^{(i)})$. And the $s_b^{(i)}$ is computed as $\gamma^{(i)}_{b\oplus d^{(i)}} \oplus \kappa^{(i)}_{b\oplus d^{(i)}}$

Since the final hybrid is identically distributed to the ideal world with the above defined simulator Sim, it remains to show that the above hybrids are all computationally indistinguishable. We defer the formal proofs to the full version[35] where we show that the hybrids are indistinguishable.

[10] Note that the hybrids further depend on the malicious receiver R^* and on the OT protocol Π, but we omit those to simplify notation.

Main thread:

1. For each $i \in [m]$, pick $b^{(i)} \leftarrow\$ \{0,1\}$ and $\rho_R^{(i)} \leftarrow\$ \{0,1\}^*$, compute $\mu_R^{(i)} = \mathsf{OTR}(b^{(i)}; \rho_R^{(i)})$ and send $(\mu_R^{(i)})_{i\in[m]}$ to S^*.
2. Upon receiving $((\mu_S^{(i)})_{i\in[m]\setminus\mathcal{A}}, \mathcal{A})$ from S^*:
   a. For each $i \in \mathcal{A}$, let $\delta_R^{(i)} = (b^{(i)}, \rho_R^{(i)})$.
   b. Sample $\beta_1, \ldots, \beta_{t_S} \leftarrow\$ [m] \setminus \mathcal{A}$ and let $\mathcal{B} = \{\beta_1, \ldots, \beta_{t_S}\}$.
   c. Let $\mathsf{Alive} = [m] \setminus (\mathcal{A} \cup \mathcal{B})$. For each $i \in \mathsf{Alive}$, compute $\kappa_{b^{(i)}}^{(i)} = \mathsf{OTD}(b^{(i)}, \mu_S^{(i)}; \rho_R^{(i)})$ and sample $d^{(i)} \leftarrow\$ \{0,1\}$.
   d. Send $((\delta_R^{(i)})_{i\in\mathcal{A}}, (d^{(i)})_{i\in\mathsf{Alive}}, \mathcal{B})$ to S^*.
3. Upon receiving $((\delta_S^{(i)})_{i\in\mathcal{B}}, (\gamma_0^{(i)}, \gamma_1^{(i)})_{i\in\mathsf{Alive}})$ from S^*, check that all defenses $\delta_S^{(i)} = (\kappa_0^{(i)}, \kappa_1^{(i)}, \rho_S^{(i)})$ are good. If not, simulate the receiver aborting. Else initialize $\mathsf{Keys} = \emptyset$ and $\mathsf{ctr} = 0$.

Rewind thread:

a. Sample randomly $\mathcal{B}' = \{\beta_1, \ldots, \beta_{t_S}\}$ as in Item 2b.
b. Let $\mathsf{Alive}' = [m] \setminus (\mathcal{A} \cup \mathcal{B}')$. For $i \in \mathsf{Alive}'$, sample $d^{(i)} \leftarrow\$ \{0,1\}$ using fresh randomness, send these values along with $\mathcal{B}'$ and $(\delta_R^{(i)})_{i\in\mathcal{A}}$ to S^*, and receive $((\delta_S^{(i)})_{i\in\mathcal{B}}), (\gamma_0^{(i)}, \gamma_1^{(i)})_{i\in\mathsf{Alive}'})$ in response.
c. For every defense $\delta_S^{(i)} = (\kappa_0^{(i)}, \kappa_1^{(i)}, \rho_S^{(i)})$ corresponding to an index $i \in \mathsf{Alive}$ (from the main thread) that was not observed in a previous rewind: If the defense is good, add[a] the keys $(\kappa_0^{(i)}, \kappa_1^{(i)})$ to Keys.
d. Increment $\mathsf{ctr} = \mathsf{ctr} + 1$ and if $\mathsf{ctr} > 2^\lambda$ abort.
e. If $|\mathsf{Keys}| < |\mathsf{Alive}| - m/9$ go to Item 3a; else proceed to the next step.

[a] In case either of the two keys is not a λ-bit string, we assume the defense is bad.

4. For each pair of keys $(\kappa_0^{(i)}, \kappa_1^{(i)})$ in Keys, compute $s_0^{(i)} = \gamma_{0\oplus d^{(i)}}^{(i)} \oplus \kappa_{0\oplus d^{(i)}}^{(i)}$ and $s_1^{(i)} = \gamma_{1\oplus d^{(i)}}^{(i)} \oplus \kappa_{1\oplus d^{(i)}}^{(i)}$ using the adjusting bits $d^{(i)}$ sampled in the main thread. Finally, use these shares to reconstruct s_0 and s_1, forward (s_0, s_1) to $\mathcal{F}_{\mathsf{OT}}$ and output whatever S^* outputs.

Figure 4 Simulator for malicious sender.