Simple, Black-Box Constructions of Adaptively Secure Protocols[⋆]

Seung Geol Choi[1], Dana Dachman-Soled[1], Tal Malkin[1], and Hoeteck Wee[2⋆⋆]

[1] Columbia University {sgchoi,dglasner,tal}@cs.columbia.edu
[2] Queens College, CUNY hoeteck@cs.qc.cuny.edu

Abstract. We present a compiler for transforming an oblivious transfer (OT) protocol secure against an adaptive semi-honest adversary into one that is secure against an adaptive malicious adversary. Our compiler achieves security in the universal composability framework, assuming access to an ideal commitment functionality, and improves over previous work achieving the same security guarantee in two ways: it uses black-box access to the underlying protocol and achieves a constant multiplicative overhead in the round complexity. As a corollary, we obtain the first constructions of adaptively secure protocols in the stand-alone model using black-box access to a low-level primitive.

1 Introduction

Secure multi-party computation (MPC) allows several mutually distrustful parties to perform a joint computation without compromising, to the greatest extent possible, the privacy of their inputs or the correctness of the outputs. An important criterion in evaluating the security guarantee is *how many* parties an adversary is allowed to corrupt and *when* the adversary determines which parties to corrupt. In this work, we focus on MPC protocols secure against an adversary that corrupts an arbitrary number of parties, and in addition, *adaptively* determines who and when to corrupt during the course of the computation. Even though the latter is a very natural and realistic assumption about the adversary, most of the MPC literature only addresses security against a static adversary, namely one that chooses (and fixes) which parties to corrupt before the protocol starts executing.

In the absence of an honest majority, secure MPC protocols can only be realized under computational assumptions. From both a theoretical and practical stand-point, it is desirable for these protocols to be based on general hardness assumptions, and in addition, to require only black-box access to the primitive guaranteed by the assumption (that is, the protocol refers only to the input/output behavior of the primitive). Indeed, the first MPC protocols achieving security

[⋆] supported in part by NSF Grants CNS-0716245, CCF-0347839, and SBE-0245014.
[⋆⋆] part of this work was done while a post-doc at Columbia University.

without an honest majority [GMW87] require non-black-box access to the underlying cryptographic primitives: the first step in the construction is to obtain protocols that are secure against semi-honest adversaries, and the second handles malicious behavior by having the parties prove in zero knowledge that they are adhering to the protocol constructions. It is the second step that requires the code of the underlying primitive with the use of general NP reductions to prove statements in zero knowledge. This aversely affects both computational complexity and communication complexity of the resulting protocol as well as the complexity of implementing the protocol.

In a recent work, Ishai et al. [IKLP06] exhibited MPC protocols that are secure against a static adversary corrupting any number of parties and that rely only on black-box access to a low-level primitive, such as (enhanced) trapdoor permutations and homomorphic encryption schemes. This, along with the follow-up work of Haitner [H08], resolves the theoretical question of the minimal assumptions under which we may obtain black-box constructions of secure MPC protocols against a static adversary. The main technical contribution in both works is to construct a secure protocol for a specific two-party functionality, that of oblivious transfer (OT). The general result then follows from a classic result of Kilian's [K88] showing that any multi-party functionality can be securely computed using black-box access to a secure OT protocol. However, none of these works addresses security against an adaptive adversary, which begs the following question:

> Is it possible to construct MPC protocols secure against a malicious, adaptive adversary that may corrupt any number of parties, given only black-box access to a low-level primitive?

Towards resolving this question, Ishai, Prabhakaran and Sahai [IPS08] established an analogue of Kilian's result for an adaptive adversary. While there has been fairly extensive work on secure OT protocols against a static malicious adversary (e.g. [NP01,K05,PVW08]), very few - namely [B98,CLOS02,KO04] - provide security against an adaptive adversary; moreover, all of those which do follow [GMW87] paradigm and exploit non-black-box access to the underlying primitive.

1.1 Our results

Our main technical contribution is the construction of efficient OT protocols that achieve security against an adaptive adversary, while relying only upon black-box access to some low-level primitive. Specifically, we provide a compiler that transforms an OT protocol secure against a semi-honest, adaptive adversary into one that is secure against a malicious, adaptive adversary, given only black-box access to the underlying OT protocol and an "ideal" commitment scheme. In addition, we achieve security in the universal composability (UC) model, where a protocol may be executed concurrently with an unknown number of other protocols [C01]. This is a notable improvement over afore-mentioned works of Ishai et al. [IKLP06,H08] which provide a compiler for semi-honest OT to malicious OT, but only for static adversaries in the stand-alone model.[3]

Theorem 1 (informal). *There exists a black-box construction of a protocol that UC realizes OT against a malicious, adaptive adversary in the $\mathcal{F}_{\mathrm{COM}}$-hybrid model, starting from any protocol that UC realizes OT against a semi-honest, adaptive adversary.*[4] *Moreover, the construction achieves a constant multiplicative blow-up in the number of rounds.*

Our construction also improves upon the earlier work of Canetti et. al [CLOS02] achieving the same guarantee; their construction is non-black-box and incurs a blow-up in round complexity proportional to the depth of the circuit computing the semi-honest OT protocol. Combined with the 2-round semi-honest OT protocol in [CLOS02,CDMW08], we obtain the first constant-round protocol for OT in the $\mathcal{F}_{\mathrm{COM}}$-hybrid model secure against a malicious, adaptive adversary.[5] Moreover, the protocol uses black-box access to a low-level primitive, that of trapdoor simulatable cryptosystems[6], which may in turn be based on the RSA, DDH, worst-case lattice assumptions or hardness of factoring.

The key conceptual insight underlying the construction is to view the [IKLP06,H08] compiler as an instantiation of the [GMW87] paradigm in the $\mathcal{F}_{\mathrm{COM}}$-hybrid model, except enforcing consistency via cut-and-choose techniques instead of using zero-knowledge proofs. This perspective leads naturally to a simpler, more modular, and more direct analysis of the previous compiler for static adversaries. In addition, we immediately obtain a string OT protocol,

[3] We note that our construction does not improve on the computational complexity of the previous compiler, as measured by the number of invocations of the underlying semi-honest OT protocol. However, we believe our construction may be combined with the OT extension protocol in [IPS08, Section 5.3] to achieve better efficiency.

[4] In both the semi-honest and the malicious OT protocols, we allow the adaptive adversary to corrupt both the sender and the receiver.

[5] In an independent work [GWZ08], Garay, Wichs and Zhou also constructed a constant-round protocol for OT in the common reference string model, secure against a malicious, adaptive adversary. Their underlying assumptions are comparatively more restrictive.

[6] Trapdoor simulatable cryptosystems are introduced in [CDMW08], as a relaxation of simulatable cryptosystems [DN00]. These are semantically secure encryption schemes with special algorithms for "obliviously" sampling public keys and ciphertexts without learning the respective secret keys and plaintexts. In addition, both of these oblivious sampling algorithms are efficiently invertible given the corresponding secret key.

which is important for obtaining round-efficient MPC protocols [LP07,IPS08]. Showing that the modified compiler achieves UC security against an adaptive adversary requires new insight in constructing a simulator and in the analysis. We defer a more detailed discussion of the construction to Section 2, and instead focus here on the applications to secure MPC derived by combining our OT protocol with various MPC protocols in the $\mathcal{F}_{\text{OT}}$-hybrid model in [IPS08].

MPC in the $\mathcal{F}_{\text{COM}}$-hybrid model. Combining our OT protocol with [IPS08, Theorem 2], we obtain UC-secure MPC protocols in the $\mathcal{F}_{\text{COM}}$-hybrid model against a malicious, adaptive adversary, which improves upon [CLOS02] in that we only require black-box access to the underlying primitive:

Theorem 2 (informal). *There exists a protocol in the $\mathcal{F}_{\text{COM}}$-hybrid model that uses black-box access to a (trapdoor) simulatable cryptosystem and UC realizes any well-formed multi-party functionality against a malicious, adaptive adversary that may corrupt any number of parties.*

The round complexity of the protocol is proportional to the depth of the circuit computing the functionality. By combining our OT protocol with [IPS08, Theorem 3], we obtain a constant-round MPC protocol in the $\mathcal{F}_{\text{COM}}$ with the same guarantees, except that the adversary is limited to corrupting up to $m - 1$ parties for a m-party functionality. The advantage of constructing UC-secure MPC protocols in the $\mathcal{F}_{\text{COM}}$-hybrid model is that they may be combined with many of the existing UC feasibility results under various set-up or modeling assumptions e.g. [CLOS02,BCNP04,CDPW07,K07], almost all of which start by showing how to UC realize $\mathcal{F}_{\text{COM}}$ in some new security model[7]. Moreover, if the protocol realizing $\mathcal{F}_{\text{COM}}$ uses black-box access to a low-level primitive, so will the combined protocol.

MPC in the stand-alone model. Next, we present our results for the stand-alone model with adaptive post-execution corruptions [C00], which is a weaker notion of security than UC security with adaptive corruptions (and in particular, our protocols in the $\mathcal{F}_{\text{COM}}$-hybrid model achieve this notion of security). Here, there is a constant-round two-party protocol that uses black-box access to a one-way function and securely realizes $\mathcal{F}_{\text{COM}}$ in the plain model without any set-up assumptions [PW09]. This immediately yields the following corollaries (via the composition theorem in [C00]):

[7] This is because it is impossible to UC realize any non-trivial 2-party functionality in the plain model (even against static adversaries) [CKL06,C01].

Corollary 1 (informal). *There exists a constant-round string OT protocol that uses black-box access to a (trapdoor) simulatable cryptosystem and is secure in the stand-alone model against a malicious, adaptive adversary.*

Corollary 2 (informal). *There exists a protocol that uses black-box access to a (trapdoor) simulatable cryptosystem and securely computes any well-formed multi-party functionality in the stand-alone model against a malicious, adaptive adversary that may corrupt any number of parties.*

Both of these results improve on the work of Beaver's [B98] which achieve similar security guarantees but with non-black-box access to the underlying primitive.

Corollary 3 (informal). *For any constant $m \geq 2$, there exists a constant-round protocol that uses black-box access to a (trapdoor) simulatable cryptosystem and securely computes any well-formed m-party functionality in the stand-alone model against a malicious, adaptive adversary that may corrupt up to $m - 1$ parties.*

This extends a result of Katz and Ostrovsky [KO04] which presents a 4-round protocol achieving the same security guarantee for two parties but relying on non-black-box access to the underlying primitive.

2 Construction

High-level description. We provide an overview of the [IKLP06,H08] compiler. Our presentation is slightly different from, and simpler than, that in the original works, and is closer in spirit to the [GMW87] compiler. Our presentation is easier to adapt to the UC setting and the adaptive setting (and also OT with strings instead of bits) since we do not need to rely on the intermediate notion and construction of a defensible OT protocol.[8] We focus on the main transformation Comp (shown in Fig 3), which "boosts" the security guarantee of an OT protocol Π from security against semi-honest receivers to security against malicious receivers while preserving the security guarantee for corrupted senders.

Phase I: Random tape generation. The sender and the receiver engage in a coin-tossing (in the well) protocol to determine a collection of $2n$ random strings for the receiver.

[8] Specifically, the previous compiler proceeds in two phases. The first [H08] transforms any semi-honest OT protocol into defensible OT protocols. A defensible OT protocol provides an intermediate level of security interpolating semi-honest and malicious OT. The second [IKLP06] transforms any defensible OT protocol into a malicious one.

Phase II: Basic execution. The sender and the receiver engage in $2n$ parallel executions of Π with random inputs: the sender will choose its own inputs randomly and independently for each of the $2n$ executions, whereas the inputs and randomness for the receiver are determined by the preceding coin-tossing protocol (one random string for each execution of Π).

Phase III: Cut-and-choose. The sender and the receiver engage in a coin-tossing protocol to pick a random subset Q of n executions, and the receiver proves that it acted accordingly to Π for the n executions in Q by revealing the inputs and randomness used for those executions. The sender verifies that the inputs and randomness are indeed consistent with both the n executions of Π and the coin-tossing protocol, and if so, we are guaranteed that the receiver must have behaved honestly in at least one of the n executions of Π not in Q (except with negligible probability).

Phase IV: Combiner. We may then apply a combiner that (essentially) yields a single secure OT protocol, starting a collection of n OT protocols all of which guarantee security against a malicious sender, but only one of which guarantee security against a malicious receiver.

To obtain a full-fledged string-OT protocol secure against both a malicious sender and a malicious receiver starting from a semi-honest bit-OT protocol, we proceed as in [IKLP06], with the addition of Step 3 to directly obtain a string-OT protocol and with references to semi-honest instead of defensible adversaries:

1. Use Comp to obtain a bit-OT protocol secure against a semi-honest sender and a malicious receiver.
2. Use OT reversal [WW06] (shown in Fig 4) to obtain a bit-OT protocol secure against a malicious sender and a semi-honest receiver.
3. Repeat in parallel to obtain a string-OT protocol secure against a malicious sender and a semi-honest receiver.
4. Use Comp again to obtain a string-OT protocol secure against malicious sender and receiver.

In this work, we will view the above construction in the $\mathcal{F}_{\text{COM}}$-hybrid model, where the $\mathcal{F}_{\text{COM}}$ functionality is used to implement the coin-tossing protocol in Phases I and III [B81,CR03]. To instantiate the protocol in the plain stand-alone model, we will need to replace $\mathcal{F}_{\text{COM}}$ with an extractable trapdoor commitment scheme. This is different from the original [IKLP06,H08] compiler, where a standard commitment scheme is used in the Phase I commitments. We will use the same construction for an adaptive adversary except a minor simplification to Comp: the sender picks the challenge Q in Phase III (even when it may be malicious) We note here that we will exploit the extractability of the Phase I commitments in a crucial way when handling an adaptive adversary.

Functionality $\mathcal{F}_{\rm COM}$

1. Upon receiving input $(\mathsf{commit}, sid, P_j, x)$ from P_i where $x \in \{0,1\}^m$, internally record the tuple (P_i, P_j, x) and send the message (sid, P_i, P_j) to the adversary; When receiving $(\texttt{ok})$ from the adversary, output $(\mathsf{receipt}, sid, P_i)$ to P_j. Ignore all subsequent $(\mathsf{commit}, ...)$ inputs.
2. Upon receiving a value (reveal, sid) from P_i, where a tuple (P_i, P_j, x) is recorded, send (x) to the adversary; When receiving $(\texttt{ok})$ from the adversary, output $(\mathsf{reveal}, sid, x)$ to P_j.

Fig. 1. String Commitment Functionality

Functionality $\mathcal{F}_{\rm OT}$

1. Upon receiving input $(\mathsf{sender}, \mathit{sid}, s_0, s_1)$ from $\mathbf{S}$ where $s_0, s_1 \in \{0,1\}^\ell$, record the pair (sid, s_0, s_1).
2. Upon receiving input $(\mathsf{receiver}, \mathit{sid}, r)$ from $\mathbf{R}$ where $r \in \{0,1\}$, send (sid, s_r) to $\mathbf{R}$ and (sid) to the adversary, and halt. If no $(\mathsf{sender}, \mathit{sid}, \ldots)$ message was previously sent, send nothing to $\mathbf{R}$.

Fig. 2. Oblivious Transfer Functionality

Improved analysis for static adversaries. We sketch an improved analysis of Comp for static adversaries in the stand-alone $\mathcal{F}_{\rm COM}$-hybrid model (i.e. showing that if Π is secure against a semi-honest receiver, then $\mathsf{Comp}(\Pi)$ is secure against a malicious receiver). Our simulator for a malicious receiver $\mathbf{R}^*$ is similar to that in [IKLP06], except we extract $\mathbf{R}^*$'s input to $\mathcal{F}_{\rm OT}$ (in the ideal model) from Phase I instead of Phase III. This way, we achieve straight-line and strict polynomial time simulation. In the reduction[9], we will need to use repeated sampling to estimate for each $r = 1, 2, \ldots, n$, a quantity related to the probability that an honest sender interacting with a malicious receiver $\mathbf{R}^*$ in $\mathsf{Comp}(\Pi)$ does not abort at the end of Phase III, and amongst the remaining executions not in Q, exactly r are consistent with the random tape determined in Phase I. This is reminiscent of the analysis in [H08, Lemma 3] but much simpler. Putting everything together, we obtain the following result[10]:

Proposition 1. *There exists a black-box construction of a string-OT protocol secure against a static, malicious adversary in the stand-alone $\mathcal{F}_{\rm COM}$-hybrid model, starting from any bit-OT protocol secure against a static, semi-honest adversary. Moreover, the construction achieves a constant multiplicative blow-*

[9] This step is not necessary if we use a non-uniform reduction.

[10] We believe our analysis also extends to static adversaries in the UC model and we plan to look into that in the full version of this paper.

INITIALIZATION.

Sender input: $(\mathsf{sender}, sid, s_0, s_1)$ where $s_0, s_1 \in \{0,1\}^\ell$.
Receiver input: $(\mathsf{receiver}, sid, r)$ where $r \in \{0,1\}$.

PHASE I: RANDOM TAPE GENERATION.

1. $\mathbf{R}$ chooses $2n$ random strings $(r_1^{\mathsf{R}}, \tau_1^{\mathsf{R}}), \ldots, (r_{2n}^{\mathsf{R}}, \tau_{2n}^{\mathsf{R}})$ and sends $(\mathsf{commit}, sid_i, r_i^{\mathsf{R}}, \tau_i^{\mathsf{R}})$ to $\mathcal{F}_{\mathrm{COM}}$, for $i = 1, 2, \ldots, 2n$.
2. Upon receiving $(\mathsf{receipt}, sid_1), \ldots, (\mathsf{receipt}, sid_{2n})$ from $\mathcal{F}_{\mathrm{COM}}$, $\mathbf{S}$ sends $2n$ random strings $(r_1^{\mathsf{S}}, \tau_1^{\mathsf{S}}), \ldots, (r_{2n}^{\mathsf{S}}, \tau_{2n}^{\mathsf{S}})$.
3. $\mathbf{R}$ sets $r_i = r_i^{\mathsf{R}} \oplus r_i^{\mathsf{S}}$ and $\tau_i = \tau_i^{\mathsf{R}} \oplus \tau_i^{\mathsf{S}}$, for $i = 1, 2, \ldots, 2n$.

PHASE II: BASIC EXECUTION.

1. $\mathbf{S}$ chooses $2n$ pairs of random inputs $(s_1^0, s_1^1), \ldots, (s_{2n}^0, s_{2n}^1)$.
2. $\mathbf{S}$ and $\mathbf{R}$ engages in $2n$ parallel executions of the protocol Π. In the ith execution, $\mathbf{S}$ inputs (s_i^0, s_i^1) and $\mathbf{R}$ inputs r_i with randomness τ_i and obtains output $s_i^{r_i}$.

PHASE III: CUT-AND-CHOOSE.

1. $\mathbf{S}$ chooses a random $q = (q_1, \ldots, q_n) \in \{0,1\}^n$. The string q is used to define a set of indices $Q \subset \{1, 2, \ldots, 2n\}$ of size n in the following way: $Q = \{2i - q_i\}_{i=1}^n$.
2. For every $i \in Q$, $\mathbf{R}$ sends (reveal, sid_i) to $\mathcal{F}_{\mathrm{COM}}$ and upon receiving $(\mathsf{reveal}, sid_i, r_i^{\mathsf{R}}, \tau_i^{\mathsf{R}})$ from $\mathcal{F}_{\mathrm{COM}}$, $\mathbf{S}$ computes (r_i, τ_i).
3. $\mathbf{S}$ checks that for all $i \in Q$, (r_i, τ_i) is consistent with $\mathbf{R}$'s messages in the i'th execution of Π. If not, $\mathbf{S}$ aborts and halts.

PHASE IV: COMBINER.

1. For every $j \notin Q$, $\mathbf{R}$ computes $\alpha_j = r \oplus r_j$ and sends $\{\alpha_j\}_{j \notin Q}$ to $\mathbf{S}$.
2. $\mathbf{S}$ computes $\sigma_0 = s_0 \oplus (\bigoplus_{j \notin Q} s_j^{\alpha_j})$ and $\sigma_1 = s_1 \oplus (\bigoplus_{j \notin Q} s_j^{1-\alpha_j})$ and sends (σ_0, σ_1).
3. $\mathbf{R}$ computes and outputs $s_r = \sigma_r \oplus (\bigoplus_{j \notin Q} s_j^{r_j})$.

Fig. 3. THE COMPILER $\mathsf{Comp}(\Pi)$

up in the number of rounds, and has a strict polynomial-time and straight-line simulator.

Our result and analysis even for static adversaries offers several improvements over that in [IKLP06,H08]:

– The simulator in [IKLP06] uses rewinding and runs in expected polynomial time, even in the $\mathcal{F}_{\mathrm{COM}}$-hybrid model.
– Our result immediately yields string-OT protocols and in a constant number of rounds.
– We eliminate many of the tedious steps in the analysis in both [IKLP06] and [H08], most notably verifying that the OT reversal protocol in [WW06] works for defensible adversaries [IKLP06, Claim 5.2]. The overall analysis is simpler, more modular, and more intuitive.

As shown in [PW09], there exists a constant-round protocol that securely realizes $\mathcal{F}_{\text{COM}}$ against a static adversary in the stand-alone model and that uses black-box access to a one-way function. Combined with Proposition 1, we obtain a constant-round string-OT protocol secure against a static, malicious adversary, relying only on black-box access to a constant-round bit-OT protocol secure against a static, semi-honest adversary.

Achieving security against adaptive adversaries. In order to cope with adaptive adversaries in $\mathsf{Comp}(\Pi)$, we will further modify our simulator for static adversaries. The main difference lies in how we simulate the sender messages in Phase II of $\mathsf{Comp}(\Pi)$. For static adversaries, we may simply follow the honest sender strategy for $\mathsf{Comp}(\Pi)$ (i.e., run the honest sender in Π on random inputs for all $2n$ parallel executions of Π). This simulation strategy fails against an adaptive adversary because we will then be unable to present random tapes that are consistent with different sender's inputs and the protocol transcript if the sender is corrupted at the end of the protocol. Instead, we will simulate honest sender messages in Phase II against a malicious receiver as follows:

1. For each i, extract the receiver's input and randomness for the i'th execution of Π from the commitment in Phase I.
2. Upon receiving a message from the receiver in the i'th execution of Π, check if all of the receiver's messages so far are consistent with its input and randomness. If so, generate the sender's response by using the simulator for Π. Otherwise, corrupt the sender in the i'th execution of Π to obtain its input and random tape and complete the simulation of the sender's messages using the honest sender strategy.

Organization. We present our analysis of Comp and OT reversal for adaptive adversaries in the UC model in Sections 3 and 4 respectively. We defer the proof of Proposition 1 for static adversaries to the full version of the paper. Henceforth, we will always refer to adaptive adversaries.

3 Achieving security against a malicious receiver

In this section, we show that Comp boosts the security guarantee from security against semi-honest receivers to security against malicious receivers.

Proposition 2. *Suppose* Π *is a protocol that UC realizes* $\mathcal{F}_{\text{OT}}$ *against a semi-honest, adaptive adversary, and let* $\mathsf{Comp}(\Pi)$ *be the protocol obtained by applying the compiler in Fig 3 to* Π*. Then,* $\mathsf{Comp}(\Pi)$ *UC realizes* $\mathcal{F}_{\text{OT}}$ *against an adaptive adversary with a semi-honest sender and a malicious receiver.*

Moreover, if Π *is in addition secure against a malicious, adaptive sender, then* $\mathsf{Comp}(\Pi)$ *UC realizes* $\mathcal{F}_{\text{OT}}$ *against an adaptive adversary with malicious sender and receiver.*

A hybrid execution. To facilitate the analysis, we introduce an intermediate setting (inspired by [IKOS07]) in which the protocol $\mathsf{Comp}(\Pi)$ is executed, where there is again a sender $\mathbf{S}$ and a receiver $\mathbf{R}$ and in addition $2n$ pairs of "virtual" parties $(\mathbf{S}_1, \mathbf{R}_1), \ldots, (\mathbf{S}_{2n}, \mathbf{R}_{2n})$. The i'th execution of Π in $\mathsf{Comp}(\Pi)$ will be executed by $\mathbf{S}_i$ and $\mathbf{R}_i$ with inputs from $\mathbf{S}$ and $\mathbf{R}$ respectively. We will require that $\mathbf{R}_1, \ldots, \mathbf{R}_{2n}$ are always semi-honest; i.e. they use a truly random tape for Π. Moreover, the environment is not aware of the "virtual parties".

PHASE I/II: BASIC EXECUTION.[11] $\mathbf{S}$ chooses $2n$ pairs of random inputs $(s_1^0, s_1^1), \ldots, (s_{2n}^0, s_{2n}^1)$ and $\mathbf{R}$ chooses $2n$ random inputs $r_1, \ldots, r_{2n}$. For each $i = 1, \ldots, 2n$, $\mathbf{S}$ activates $\mathbf{S}_i$ with $(\mathsf{sender}, \mathit{sid}_i, s_i^0, s_i^1)$ and $\mathbf{R}$ activates $\mathbf{R}_i$ with $(\mathsf{receiver}, \mathit{sid}_i, r_i)$. In $\text{HYBRID}_{\Pi,\mathcal{A},\mathcal{Z}}$, the parties $\mathbf{S}_i$ and $\mathbf{R}_i$ execute Π in parallel. In $\text{HYBRID}_{\mathcal{F}_{\text{OT}},\mathcal{A},\mathcal{Z}}$, the parties $\mathbf{S}_i$ and $\mathbf{R}_i$ call the ideal functionality $\mathcal{F}_{\text{OT}}$.

PHASE III: CUT-AND-CHOOSE. $\mathbf{S}$ chooses a random $q \in \{0,1\}^n$ which identifies $Q \subset \{1, 2, \ldots, 2n\}$ as in $\mathsf{Comp}(\Pi)$ and sends q to $\mathbf{R}$. $\mathbf{S}$ checks that for all $i \in Q$, $\mathbf{S}_i$ is not corrupted. Otherwise, abort.

PHASE IV: COMBINER. Proceed as in Phase IV of $\mathsf{Comp}(\Pi)$.

We say that an adversary $\mathcal{A}$ in the hybrid execution is well-formed if it satisfies the following properties:

- When $\mathcal{A}$ corrupts $\mathbf{S}$, it also corrupts each of $\mathbf{S}_1, \ldots, \mathbf{S}_{2n}$. Moreover, if $\mathbf{S}$ is semi-honest, then $\mathbf{S}_1, \ldots, \mathbf{S}_{2n}$ are semi-honest.
- When $\mathcal{A}$ corrupts $\mathbf{R}$, it also corrupts each of $\mathbf{R}_1, \ldots, \mathbf{R}_{2n}$. Moreover, $\mathbf{R}_1, \ldots, \mathbf{R}_{2n}$ are always semi-honest, even if $\mathbf{R}$ is malicious.
- If $\mathbf{R}$ is corrupted, then $\mathcal{A}$ may corrupt any of $\mathbf{S}_1, \ldots, \mathbf{S}_{2n}$ with semi-honest behavior, without corrupting $\mathbf{S}$.
- Upon receiving the set Q in Phase III from $\mathbf{S}$, $\mathcal{A}$ may corrupt all of $\mathbf{R}_j, j \in Q$ with semi-honest behavior, even if neither $\mathbf{R}$ nor $\mathbf{S}$ is corrupted. However, if $\mathbf{R}$ is not corrupted, then $\mathbf{R}_j, j \notin Q$ are also not corrupted.

We will also stipulate that the communication channels between $\mathbf{S}$ and each of $\mathbf{S}_1, \ldots, \mathbf{S}_{2n}$ are private and authenticated. The same holds for the communication channels between $\mathbf{R}$ and each of $\mathbf{R}_1, \ldots, \mathbf{R}_{2n}$. In addition, $\mathbf{S}$ learns whether each of $\mathbf{S}_1, \ldots, \mathbf{S}_{2n}$ is corrupted.

[11] The choice of notation is so that Phase III always corresponds to cut-and-choose and Phase IV corresponds to combiner in both $\mathsf{Comp}(\Pi)$ and in the hybrid executions.

Lemma 1. *For every adversary $\mathcal{A}$ that interacts with* $\mathsf{Comp}(\Pi)$ *in the* $\mathcal{F}_{\text{COM}}$*-hybrid model, there exists a well-formed adversary* $\mathcal{A}'$ *that interacts in the hybrid execution running* Π*, such that for every environment* $\mathcal{Z}$*,*

$$\text{EXEC}^{\mathcal{F}_{\text{COM}}}_{\mathsf{Comp}(\Pi),\mathcal{A},\mathcal{Z}} \equiv \text{HYBRID}_{\Pi,\mathcal{A}',\mathcal{Z}}$$

In the first step, we show how to enforce semi-honest behavior of $\mathbf{R}_1, \ldots, \mathbf{R}_{2n}$ in $\text{HYBRID}_{\Pi,\mathcal{A}',\mathcal{Z}}$. The high-level strategy is as follows: if a corrupted receiver in $\mathsf{Comp}(\Pi)$ deviates from semi-honest behavior in the i'th execution of Π in Phase III, we corrupt $\mathbf{S}_i$ in $\text{HYBRID}_{\Pi,\mathcal{A}',\mathcal{Z}}$ to obtain its input and randomness, and continue the simulation by running the honest sender strategy.

Proof. As usual, $\mathcal{A}'$ works by invoking a copy of $\mathcal{A}$ and running a simulated interaction of $\mathcal{A}$ with $\mathcal{Z}$ and the parties $\mathbf{S}$ and $\mathbf{R}$. We will refer to the communication of $\mathcal{A}'$ with $\mathcal{Z}$ and $\mathsf{Comp}(\Pi)$ as external communication, and that with the simulated $\mathcal{A}$ as internal communication. More precisely, $\mathcal{A}'$ works as follows:

Simulating the communication with $\mathcal{Z}$: Every input value that $\mathcal{A}'$ receives from $\mathcal{Z}$ externally is written into the adversary $\mathcal{A}$'s input tape (as if coming from $\mathcal{A}$'s environment). Every output value written by $\mathcal{A}$ on its output tape is copied to $\mathcal{A}'$'s own output tape (to be read by the external $\mathcal{Z}$).

Simulating the case when neither party is corrupted:

PHASE I. $\mathcal{A}'$ internally passes $\mathcal{A}$ the message $(\mathsf{receipt}, sid_1), (\mathsf{receipt}, sid_2), \ldots, (\mathsf{receipt}, sid_{2n})$ as if sent from $\mathcal{F}_{\text{COM}}$ to $\mathbf{S}$. Then, $\mathcal{A}'$ chooses $2n$ random strings $(r_1^{\mathsf{S}}, \tau_1^{\mathsf{S}}), \ldots, (r_{2n}^{\mathsf{S}}, \tau_{2n}^{\mathsf{S}})$, and simulates $\mathbf{S}$ sending $\mathbf{R}$ those $2n$ strings.

PHASE II. For each round in the protocol Π, if it is the receiver's turn, then for each $i = 1, \ldots, 2n$, $\mathcal{A}'$ obtains β_i from $\mathbf{R}_i$ for the corresponding round. Next, $\mathcal{A}'$ internally passes $\mathcal{A}$ the message $(\beta_1, \ldots, \beta_{2n})$, as if sent from $\mathbf{R}$ to $\mathbf{S}$. The sender's turn is handled analogously.

PHASE III. When $\mathbf{S}$ externally sends q which determines Q, then for each $i \in Q$: corrupt $\mathbf{R}_i$ to obtain (r_i, τ_i) and compute $r_i^{\mathsf{R}} = r_i \oplus r_i^{\mathsf{S}}$ and $\tau_i^{\mathsf{R}} = \tau_i \oplus \tau_i^{\mathsf{S}}$. Send $(\mathsf{reveal}, sid_i, r_i^{\mathsf{R}}, \tau_i^{\mathsf{R}})$ to $\mathcal{A}$ as if coming from $\mathcal{F}_{\text{COM}}$.

PHASE IV. Just forward all the messages between $\mathbf{S}$ and $\mathbf{R}$.

Simulating the case when only the sender is corrupted: This is essentially the same as when neither party is corrupted, except the values $(r_1^{\mathsf{S}}, \tau_1^{\mathsf{S}}), \ldots, (r_{2n}^{\mathsf{S}}, \tau_{2n}^{\mathsf{S}})$ in Phase I and the value q in Phase III are chosen by $\mathcal{A}$.

Simulating the case when only the receiver is corrupted:

PHASE I. $\mathcal{A}'$ externally corrupts $(\mathbf{R}_1, \ldots, \mathbf{R}_{2n})$ to obtain $(\tau_1, \ldots, \tau_{2n})$ and picks $2n$ random values $r_1, \ldots, r_{2n}$. Next, $\mathcal{A}'$ obtains from $\mathcal{A}$ the messages $(\mathsf{commit}, \mathit{sid}_i, r_i^{\mathrm{R}}, \tau_i^{\mathrm{R}})$ as sent by $\mathbf{R}$ to $\mathcal{F}_{\mathrm{COM}}$. Then, $\mathcal{A}'$ sets $r_i^{\mathrm{S}} = r_i \oplus r_i^{\mathrm{R}}$ and $\tau_i^{\mathrm{S}} = \tau_i \oplus \tau_i^{\mathrm{R}}$ for $i = 1, 2, \ldots, 2n$ and internally passes $(r_1^{\mathrm{S}}, \tau_1^{\mathrm{S}}), \ldots, (r_{2n}^{\mathrm{S}}, \tau_{2n}^{\mathrm{S}})$ to $\mathcal{A}$ as if sent by $\mathbf{S}$ to $\mathbf{R}$.

PHASE II. We need to simulate the external messages sent by $\mathbf{S}$ in $\mathsf{Comp}(\Pi)$ (with the "help" of $\mathbf{S}_1, \ldots, \mathbf{S}_{2n}$). If $\mathbf{R}$ behaves consistently in the ith execution of Π, we will just obtain the corresponding message from $\mathbf{S}_i$; otherwise, we will corrupt $\mathbf{S}_i$ so that we may compute those messages.

First, we handle receiver messages in $\mathsf{Comp}(\Pi)$. Whenever $\mathcal{A}$ sends a message $(\beta_1, \ldots, \beta_{2n})$ from $\mathbf{R}$ where β_i is the message in the ith parallel execution of Π, do the following for each $i = 1, \ldots, 2n$:

- If $\mathbf{R}_i$ has not aborted and β_i is consistent with (r_i, τ_i), deliver the corresponding message from $\mathbf{R}_i$ to $\mathbf{S}_i$.
- If $\mathbf{R}_i$ has not aborted and β_i is not consistent with (r_i, τ_i), $\mathcal{A}'$ tells $\mathbf{R}_i$ to abort. In addition, $\mathcal{A}'$ corrupts $\mathbf{S}_i$ to obtain its input (s_i^0, s_i^1) and its randomness.
- If $\mathbf{R}_i$ has aborted, then record β_i and do nothing.

Next, we handle sender messages in $\mathsf{Comp}(\Pi)$. Whenever $\mathcal{A}$ expects a message $(\gamma_1, \ldots, \gamma_{2n})$ from $\mathbf{S}$, where γ_i is the message in the ith parallel execution of Π, do the following for each $i = 1, \ldots, 2n$:

- If $\mathbf{S}_i$ is corrupted, then $\mathcal{A}'$ computes γ_i according to $\mathbf{S}_i$'s input and randomness and the previous messages from $\mathbf{R}_i$.
- If $\mathbf{S}_i$ is not corrupted, then set γ_i to be the corresponding message sent from $\mathbf{S}_i$ to $\mathbf{R}_i$.

$\mathcal{A}'$ then sends $(\gamma_1, \ldots, \gamma_{2n})$ to $\mathcal{A}$ as if sent by $\mathbf{S}$ to $\mathbf{R}$.

PHASE III. Deliver q sent externally by $\mathbf{S}$ to $\mathbf{R}$. Check that for all $i \in Q$, $\mathbf{S}_i$ is not corrupted. Otherwise, abort.

PHASE IV. Just forward all the messages between $\mathbf{S}$ and $\mathbf{R}$.

Dealing with corruption of parties: When the simulated $\mathcal{A}$ internally corrupts $\mathbf{R}$, $\mathcal{A}'$ externally corrupts $\mathbf{R}$ and thus $\mathbf{R}_1, \ldots, \mathbf{R}_{2n}$, and learns the values $r_1, \ldots, r_{2n}$ and $\tau_1, \ldots, \tau_{2n}$ (in addition to the input r). $\mathcal{A}'$ then sets $r_i^{\mathrm{R}} = r_i \oplus r_i^{\mathrm{S}}$ and $\tau_i^{\mathrm{R}} = \tau_i \oplus \tau_i^{\mathrm{S}}$ for $i = 1, 2, \ldots, 2n$ and internally passes $(r_1^{\mathrm{R}}, \tau_1^{\mathrm{R}}), \ldots, (r_{2n}^{\mathrm{R}}, \tau_{2n}^{\mathrm{R}})$ to $\mathcal{A}$ as the randomness for $\mathbf{R}$ in $\mathsf{Comp}(\Pi)$. Similarly, when the simulated $\mathcal{A}$ internally corrupts $\mathbf{S}$, $\mathcal{A}'$ externally corrupts $\mathbf{S}$ and thus $\mathbf{S}_1, \ldots, \mathbf{S}_{2n}$ and learns the values $(s_1^0, s_1^1), \ldots, (s_{2n}^0, s_{2n}^1)$ along with the randomness used by $\mathbf{S}_1, \ldots, \mathbf{S}_{2n}$ in the $2n$ executions of Π. $\mathcal{A}'$

then internally passes all of these values to $\mathcal{A}$ as the randomness for $\mathbf{S}$ in $\mathsf{Comp}(\Pi)$. In addition, for all $i \in Q$, $\mathcal{A}'$ passes the value $(r_i^{\mathrm{R}}, \tau_i^{\mathrm{R}})$ to $\mathcal{A}$ as the value sent from $\mathcal{F}_{\mathrm{COM}}$ to $\mathbf{S}$ in Phase III.

It is straight-forward to verify that in Phase III, checking $\mathbf{S}_i$ is not corrupted in $\mathrm{HYBRID}_{\Pi,\mathcal{A}',\mathcal{Z}}$ is identical to $\mathbf{R}$ behaving consistently in the ith execution of Π in $\mathsf{Comp}(\Pi)$. Thus, the abort condition at the end of Phase III are identical. We may therefore conclude that the ensembles EXEC and HYBRID are identical. □

Lemma 2. *For every well-formed adversary $\mathcal{A}'$ that interacts in the hybrid execution running Π, there exists a well-formed adversary $\mathcal{A}''$ that interacts in the hybrid execution running $\mathcal{F}_{\mathrm{OT}}$, such that for every environment $\mathcal{Z}$,*

$$\mathrm{HYBRID}_{\Pi,\mathcal{A}',\mathcal{Z}} \overset{c}{\approx} \mathrm{HYBRID}_{\mathcal{F}_{\mathrm{OT}},\mathcal{A}'',\mathcal{Z}}$$

Proof (sketch). The idea is that we may interpret $\mathrm{HYBRID}_{\Pi,\mathcal{A}',\mathcal{Z}}$ as an execution involving $4n+2$ parties $\mathbf{S}, \mathbf{R}, \mathbf{S}_1, \ldots, \mathbf{S}_{2n}, \mathbf{R}_1, \ldots, \mathbf{R}_{2n}$ jointly running some protocol that uses Π as a sub-routine, and $\mathrm{HYBRID}_{\mathcal{F}_{\mathrm{OT}},\mathcal{A}'',\mathcal{Z}}$ as an execution involving the same $4n+2$ parties running the same protocol except with an ideal $\mathcal{F}_{\mathrm{OT}}$ functionality instead of Π. The claim then follows from the UC composition [C01]. □

Lemma 3. *For every well-formed adversary $\mathcal{A}''$ that interacts in the hybrid execution running $\mathcal{F}_{\mathrm{OT}}$, there exists an ideal-process adversary $\mathcal{S}$, such that for every environment $\mathcal{Z}$,*

$$\mathrm{HYBRID}_{\mathcal{F}_{\mathrm{OT}},\mathcal{A}'',\mathcal{Z}} \overset{s}{\approx} \mathrm{IDEAL}_{\mathcal{F}_{\mathrm{OT}},\mathcal{S},\mathcal{Z}}$$

Proof. Again, we first specify $\mathcal{S}$ depending on the corruption pattern:

Simulating the communication with $\mathcal{Z}$: Every input value that $\mathcal{S}$ receives from $\mathcal{Z}$ externally is written into the adversary $\mathcal{A}''$'s input tape (as if coming from $\mathcal{A}''$'s environment). Every output value written by $\mathcal{A}''$ on its output tape is copied to $\mathcal{S}$'s own output tape (to be read by the external $\mathcal{Z}$).

Simulating the case when neither party is corrupted:

PHASE I/II. Send $(sid_1, \ldots, sid_{2n})$ internally to $\mathcal{A}''$ as if sent from $\mathcal{F}_{\mathrm{OT}}$.

PHASE III. Send a random $q \in \{0,1\}^n$ as if sent from $\mathbf{S}$ to $\mathbf{R}$. For each $i \in Q$, when $\mathcal{A}''$ corrupts $\mathbf{R}_i$, pick a random $r_i \in \{0,1\}$ and a random $s_i^{r_i} \in \{0,1\}^{\ell}$.

PHASE IV. Send random $\{\alpha_j\}_{j \notin Q}$ as if sent from $\mathbf{R}$ and random (σ_0, σ_1) as if sent from $\mathbf{S}$.

Simulating the case when only the sender is corrupted:

PHASE I/II. When $\mathcal{A}''$ sends $(\mathsf{sender}, \mathit{sid}_i, s_i^0, s_i^1)$ to $\mathcal{F}_{\text{OT}}$ as $\mathbf{S}_i$, then $\mathcal{S}$ records (s_i^0, s_i^1). Then, send $(\mathit{sid}_1, \ldots, \mathit{sid}_{2n})$ internally to $\mathcal{A}''$ as if sent from $\mathcal{F}_{\text{OT}}$.

PHASE III. Proceed as in the case neither party is corrupted, except q is chosen by $\mathcal{A}''$.

PHASE IV. Send random $\{\alpha_j\}_{j \notin Q}$ to $\mathcal{A}''$ as if sent from $\mathbf{R}$. When $\mathcal{A}''$ sends (σ_0, σ_1) as $\mathbf{S}$, compute $s_0 = \sigma_0 \oplus (\bigoplus_{j \notin Q} s_j^{\alpha_j})$ and $s_1 = \sigma_1 \oplus (\bigoplus_{j \notin Q} s_j^{1-\alpha_j})$. Next, send $(\mathsf{sender}, s_0, s_1)$ to $\mathcal{F}_{\text{OT}}$ as if sent from $\mathbf{S}$.

Simulating the case when only the receiver is corrupted:

PHASE I/II. $\mathcal{S}$ picks $2n$ pairs of random inputs $(s_1^0, s_1^1), \ldots, (s_{2n}^0, s_{2n}^1)$. If $\mathcal{A}''$ sends $(\mathsf{receiver}, \mathit{sid}_i, r_i)$ to $\mathcal{F}_{\text{OT}}$ as $\mathbf{R}_i$, record r_i and pass $(\mathit{sid}_i, s_i^{r_i})$ to $\mathcal{A}''$ as if sent by $\mathcal{F}_{\text{OT}}$ to $\mathbf{R}_i$. If $\mathcal{A}''$ corrupts $\mathbf{S}_i$, then $\mathcal{S}$ presents (s_i^0, s_i^1) as $\mathbf{S}_i$'s input to $\mathcal{A}''$.

PHASE III. Pick a random $q \in \{0,1\}^n$ and send q to $\mathcal{A}''$ as if coming from $\mathbf{S}$. Compute $Q \subset \{1, 2, \ldots, 2n\}$ as in $\mathsf{Comp}(\Pi)$. Check that for all $i \in Q$, $\mathbf{S}_i$ is not corrupted. Otherwise, $\mathcal{S}$ simulates an abort from $\mathbf{S}$.

PHASE IV. Compute $j^* \notin Q$ where $\mathbf{S}_{j^*}$ is not corrupted; output $\mathsf{failure}$ if such a j^* does not exist. When $\mathcal{A}''$ sends $\{\alpha_j\}_{j \notin Q}$ as $\mathbf{R}$, compute $r = \alpha_{j^*} \oplus r_{j^*}$ and send $(\mathsf{receiver}, \mathit{sid}, r)$ to $\mathcal{F}_{\text{OT}}$. Upon receiving (sid, s_r) from $\mathcal{F}_{\text{OT}}$, compute (σ_0, σ_1) so that σ_r is consistent with s_r as follows:

- If $r = 0$, then $\sigma_0 = s_0 \oplus (\bigoplus_{j \notin Q} s_j^{\alpha_j})$ and σ_1 is a random string in $\{0,1\}^\ell$.
- If $r = 1$, then σ_0 is a random string in $\{0,1\}^\ell$ and $\sigma_1 = s_1 \oplus (\bigoplus_{j \notin Q} s_j^{1-\alpha_j})$.

$\mathcal{S}$ then sends (σ_0, σ_1) to $\mathcal{A}''$ as if sent by $\mathbf{S}$ to $\mathbf{R}$.

Dealing with corruptions: Corruptions of $\mathbf{R}_1, \ldots, \mathbf{R}_{2n}, \mathbf{S}_1, \ldots, \mathbf{S}_{2n}$ may be handled as above. For corruptions of $\mathbf{R}$ and $\mathbf{S}$, we will consider two cases depending on the corruption schedule. In the first case, at least one of the parties is corrupted before the message (σ_0, σ_1) is sent.

- Once $\mathbf{S}$ is corrupted, $\mathcal{S}$ learns the actual input (s_0, s_1). If $\mathbf{S}$ is corrupted before the messages (σ_0, σ_1) are computed, then $\mathcal{S}$ may simply present $(s_1^0, s_1^1), \ldots, (s_{2n}^0, s_{2n}^1)$ (as chosen in Phase I) as the randomness of $\mathbf{S}$. Otherwise, $\mathcal{S}$ modifies $s_{j^*}^{1-r_{j^*}}$ (if necessary) so that both relations $\sigma_0 = s_0 \oplus (\bigoplus_{j \notin Q} s_j^{\alpha_j})$ and $\sigma_1 = s_1 \oplus (\bigoplus_{j \notin Q} s_j^{1-\alpha_j})$ are satisfied.

  - Once $\mathbf{R}$ is corrupted, $\mathcal{S}$ learns the actual input r. If $\mathbf{R}$ is corrupted before the messages $\{\alpha_j\}_{j \notin Q}$ are computed, then $\mathcal{S}$ may simply present $(r_1, \ldots, r_{2n})$ (as chosen in Phase I) as the randomness of $\mathbf{R}$. Otherwise, $\mathcal{S}$ modifies $\{r_j\}_{j \notin Q}$ so that $r_j = r \oplus \alpha_j$. In addition, $\mathcal{S}$ presents $s_i^{r_i}$ as the output of $\mathbf{R}_i, i = 1, 2, \ldots, 2n$.

In the other case, neither party is corrupted when the message (σ_0, σ_1) is sent.

  - Once $\mathbf{S}$ is corrupted, we will modify both $s_{j^*}^0$ and $s_{j^*}^1$ so that (σ_0, σ_1) is consistent with (s_0, s_1).
  - Once $\mathbf{R}$ is corrupted, we will first modify $\{r_j\}_{j \notin Q}$ as in the previous case and then modify $s_{j^*}^{r_{j^*}}$ so that σ_r is consistent with s_r.

We claim that if $\mathcal{S}$ does not output $\mathsf{failure}$, then the ensembles $\text{HYBRID}_{\mathcal{F}_{\text{OT}}, \mathcal{A}'', \mathcal{Z}}$ and $\text{IDEAL}_{\mathcal{F}_{\text{OT}}, \mathcal{S}, \mathcal{Z}}$ are identical. This is clear up to the end of Phase III. For Phase IV, observe that if $\mathbf{S}$ and $\mathbf{S}_{j^*}$ are not corrupted, then from the view of $\mathcal{A}''$ and $\mathcal{Z}$ in $\text{HYBRID}_{\mathcal{F}_{\text{OT}}, \mathcal{A}'', \mathcal{Z}}$, the string $s_{j^*}^{1-r_{j^*}}$ is truly random. As such, σ_{1-r} is also truly random. Similarly, if $\mathbf{R}$ is not corrupted, then from the view of $\mathcal{A}''$ and $\mathcal{Z}$, the n values $\{r_j\}_{j \notin Q}$ are truly random and thus $\{\alpha_j\}_{j \notin Q}$ are also truly random. Furthermore, if neither $\mathbf{S}$ nor $\mathbf{R}$ is corrupted just before the message (σ_0, σ_1) is sent, then from the view of $\mathcal{A}''$ and $\mathcal{Z}$, both $s_{j^*}^0$ and $s_{j^*}^1$ are truly random, and thus both σ_0 and σ_1 are truly random.

It remains to show that $\mathcal{S}$ outputs $\mathsf{failure}$ with negligible probability. Observe that $\mathcal{S}$ only outputs $\mathsf{failure}$ if at the start of Phase IV, all of the following conditions hold:

- Neither party has aborted. In addition, the sender is honest at the start of Phase IV, so the challenge q is chosen at random.
- Amongst the n pairs of parties $(\mathbf{S}_1, \mathbf{S}_2), \ldots, (\mathbf{S}_{2n-1}, \mathbf{S}_{2n})$, exactly one party in each pair is corrupted. Otherwise, if there is a pair where both parties are corrupted, then $\mathbf{S}$ will abort at the end of Phase III; and if there is a pair where neither party is corrupted, then there is an uncorrupted $\mathbf{S}_{j^*}$.
- The set Q corresponding to the challenge q is exactly the set of n uncorrupted parties (one in each pair).

Clearly, the last condition only holds with probability 2^{-n} over a random choice of q. $\square$

4 Malicious sender and semi-honest receiver

In this section, we reverse the OT protocol from the previous section to obtain one that is secure for a malicious sender and a semi-honest receiver. The

INITIALIZATION.

Sender input: $(\mathsf{sender}, sid, s_0, s_1)$ where $s_0, s_1 \in \{0,1\}$.
Receiver input: $(\mathsf{receiver}, sid, r)$ where $r \in \{0,1\}$.

PHASE I: CALL $\mathcal{F}_{\mathrm{OT}}$.

1. $\mathbf{R}$ chooses a bit $\rho \in \{0,1\}$ and sends $(\mathsf{sender}, sid, \rho, \rho \oplus r)$ to $\mathcal{F}_{\mathrm{OT}}$.
2. $\mathbf{S}$ sends $(\mathsf{receiver}, sid, s_0 \oplus s_1)$ to $\mathcal{F}_{\mathrm{OT}}$.

PHASE II: REVERSE.

1. Upon receiving (sid, a) from $\mathcal{F}_{\mathrm{OT}}$, $\mathbf{S}$ computes $\alpha = s_0 \oplus a$ and sends α to $\mathbf{R}$.
2. Upon receiving α, $\mathbf{R}$ computes and outputs $\rho \oplus \alpha$.

Fig. 4. THE OT-REVERSAL PROTOCOL ψ

construction (shown in Fig 4) is the same as that in [WW06], the novelty lies in the analysis which establishes security against an adaptive adversary. We note that the analysis though tedious, is fairly straight-forward.

Proposition 3. *For every adaptive adversary $\mathcal{A}$ that interacts with the protocol ψ in the $\mathcal{F}_{\mathrm{OT}}$-hybrid model, there exists an adaptive adversary $\mathcal{S}$ that interacts with $\mathcal{F}_{\mathrm{OT}}$, such that for every environment $\mathcal{Z}$,*

$$\mathrm{EXEC}^{\mathcal{F}_{\mathrm{OT}}}_{\psi,\mathcal{A},\mathcal{Z}} \equiv \mathrm{IDEAL}_{\mathcal{F}_{\mathrm{OT}},\mathcal{S},\mathcal{Z}}.$$

Moreover, the corruption pattern in $\mathcal{S}$ is the reverse of that in $\mathcal{A}$.

Proof (sketch). As usual, $\mathcal{S}$ works by invoking a copy of $\mathcal{A}$ and running a simulated interaction of $\mathcal{A}$ with $\mathcal{Z}$ and the parties $\mathbf{S}$ and $\mathbf{R}$ in the $\mathcal{F}_{\mathrm{OT}}$-hybrid model. We will refer to the communication of $\mathcal{S}$ with $\mathcal{Z}$ and ψ as external communication, and that with the simulated $\mathcal{A}$ as internal communication. In addition, we will refer to the $\mathcal{F}_{\mathrm{OT}}$ functionality in the real execution as the internal $\mathcal{F}_{\mathrm{OT}}$, and that in the ideal execution as the external $\mathcal{F}_{\mathrm{OT}}$. $\mathcal{S}$ works as follows:

Simulating the communication with $\mathcal{Z}$: Every input value that $\mathcal{S}$ externally receives from $\mathcal{Z}$ is written into the adversary $\mathcal{A}$'s input tape (as if coming from $\mathcal{A}$'s environment). Every output value written by $\mathcal{A}$ on its output tape is copied to $\mathcal{S}$'s own output tape (to be read by the external $\mathcal{Z}$).

Simulating the case when neither party is corrupted: $\mathcal{S}$ internally passes (sid) to $\mathcal{A}$ as if coming from the internal $\mathcal{F}_{\mathrm{OT}}$. When $\mathcal{S}$ receives (sid) from the external $\mathcal{F}_{\mathrm{OT}}$, $\mathcal{S}$ chooses $\alpha \in \{0,1\}$ at random and sends it to $\mathcal{A}$ as if coming from $\mathbf{S}$.

Simulating the case when only the sender is corrupted: When $\mathcal{A}$ sends $(\mathsf{receiver}, \mathit{sid}, d)$ to the internal $\mathcal{F}_{\mathrm{OT}}$ as $\mathbf{S}$, $\mathcal{S}$ chooses $a \in \{0,1\}$ at random and sends (sid, a) to $\mathcal{A}$ the output from the internal $\mathcal{F}_{\mathrm{OT}}$. When $\mathcal{A}$ sends α as $\mathbf{S}$, $\mathcal{S}$ sends $(\mathsf{sender}, \mathit{sid}, a \oplus \alpha, a \oplus \alpha \oplus d)$ to the external $\mathcal{F}_{\mathrm{OT}}$.

Simulating the case when only the receiver is corrupted: When $\mathcal{A}$ internally sends $(\mathsf{sender}, \mathit{sid}, a_0, a_1)$ to $\mathcal{F}_{\mathrm{OT}}$ as $\mathbf{R}$, $\mathcal{S}$ sets $\rho = a_0$, $r = a_0 \oplus a_1$ and externally sends $(\mathsf{receiver}, \mathit{sid}, r)$ to $\mathcal{F}_{\mathrm{OT}}$. Upon receiving (sid, s_r) externally from $\mathcal{F}_{\mathrm{OT}}$, $\mathcal{S}$ internally sends $\alpha = s_r \oplus \rho$ to $\mathcal{A}$ as if coming from $\mathbf{S}$.

Dealing with corruptions: When $\mathbf{R}$ is corrupted, $\mathcal{S}$ needs to present $\mathcal{A}$ with a consistent random tape comprising of a single bit ρ. When $\mathbf{S}$ is corrupted, $\mathcal{S}$ needs to present $\mathcal{A}$ with the output bit a which $\mathbf{S}$ receives from the internal $\mathcal{F}_{\mathrm{OT}}$. We consider four cases depending on the corruption schedule:

- Case 1: $\mathbf{R}$ is corrupted before it sends its input to the internal $\mathcal{F}_{\mathrm{OT}}$. In this case, $\mathbf{S}$ proceeds as in the case when only the receiver is corrupted to compute ρ and r. If and when $\mathbf{S}$ is corrupted, $\mathcal{S}$ computes $a = \rho \oplus rd$ where d is $\mathbf{S}$'s input to the internal $\mathcal{F}_{\mathrm{OT}}$ (set to $s_0 \oplus s_1$ if $\mathbf{S}$ is honest when it submits its input to the internal $\mathcal{F}_{\mathrm{OT}}$).
- Case 2: Neither party is corrupted when α is sent. In this case, $\mathcal{S}$ picks a random $\alpha \in \{0,1\}$. Then, when $\mathbf{R}$ is corrupted, $\mathcal{S}$ learns both its input r and its output s_r, and computes $\rho = \alpha \oplus s_r$. When $\mathbf{S}$ is corrupted, $\mathcal{S}$ learns its input s_0, s_1 and computes $a = \alpha \oplus s_0$.

If neither Case 1 nor Case 2 holds, then the adversary $\mathcal{A}$ corrupts either $\mathbf{R}$ or $\mathbf{S}$ (or both) and learns at least one of ρ and a before seeing α.

- Case 3: $\mathcal{A}$ learns a first. This means $\mathcal{A}$ corrupts $\mathbf{S}$ first and corrupts $\mathbf{R}$ (if at all) after $\mathbf{S}$ receives a from the internal $\mathcal{F}_{\mathrm{OT}}$. Then, $\mathcal{S}$ proceeds as in the case where only the sender is corrupted and picks a random $a \in \{0,1\}$. When $\mathbf{R}$ is corrupted, $\mathcal{S}$ learns r and computes $\rho = a \oplus rd$ (where d is again $\mathbf{S}$'s input to the internal $\mathcal{F}_{\mathrm{OT}}$).
- Case 4: $\mathcal{A}$ learns ρ first. This means either $\mathcal{A}$ corrupts $\mathbf{R}$ first, or $\mathcal{A}$ corrupts $\mathbf{R}$ before $\mathbf{S}$ receives a from the internal $\mathcal{F}_{\mathrm{OT}}$.[12] In this case, $\mathcal{S}$ picks $\rho \in \{0,1\}$ at random when $\mathbf{R}$ is corrupted, and subsequently (if and when $\mathcal{A}$ corrupts $\mathbf{S}$) computes $a = \rho \oplus rd$.

Finally, we need to check that $\mathrm{EXEC}^{\mathcal{F}_{\mathrm{OT}}}_{\psi,\mathcal{A},\mathcal{Z}} \equiv \mathrm{IDEAL}_{\mathcal{F}_{\mathrm{OT}},\mathcal{S},\mathcal{Z}}$, which is similar to that in [WW06] which addresses static corruptions. □

Acknowledgments. We thank Ran Canetti, Yehuda Lindell and Manoj Prabhakaran for helpful discussions.

[12] In particular, it could be that $\mathcal{A}$ corrupts $\mathbf{S}$ at the start of the protocol (learning nothing at this point), and then corrupts $\mathbf{R}$ immediately after it sends its input to the internal $\mathcal{F}_{\mathrm{OT}}$.

References

[B81] M. Blum. Coin flipping by telephone. In *CRYPTO*, 1981.

[B98] D. Beaver. Adaptively secure oblivious transfer. In *ASIACRYPT*, 1998.

[BCNP04] B. Barak, R. Canetti, J. B. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *FOCS*, 2004.

[C00] R. Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[C01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, 2001.

[CDMW08] S. G. Choi, D. Dachman-Soled, T. Malkin, and H. Wee. Non-committing encryption and adaptively secure protocols from weaker assumptions, 2008. manuscript.

[CDPW07] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally composable security with global setup. In *TCC*, 2007.

[CKL06] R. Canetti, E. Kushilevitz, and Y. Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *J. Cryptology*, 19(2):135–167, 2006.

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, 2002.

[CR03] R. Canetti and T. Rabin. Universal composition with joint state. In *CRYPTO*, 2003.

[DN00] I. Damgård and J. B. Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *CRYPTO*, 2000.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, 1987.

[GWZ08] J. A. Garay, D. Wichs, and H.-S. Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. Cryptology ePrint 2008/534, 2008.

[H08] I. Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In *TCC*, 2008.

[IKLP06] Y. Ishai, E. Kushilevitz, Y. Lindell, and E. Petrank. Black-box constructions for secure computation. In *STOC*, 2006.

[IKOS07] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, 2007.

[IPS08] Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, 2008.

[K88] J. Kilian. Founding cryptography on oblivious transfer. In *STOC*, 1988.

[K05] Y. T. Kalai. Smooth projective hashing and two-message oblivious transfer. In *EUROCRYPT*, 2005.

[K07] J. Katz. Universally composable multi-party computation using tamper-proof hardware. In *EUROCRYPT*, 2007.

[KO04] J. Katz and R. Ostrovsky. Round-optimal secure two-party computation. In *CRYPTO*, 2004.

[LP07] Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *EUROCRYPT*, 2007.

[NP01] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *SODA*, 2001.

[PVW08] C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, 2008.

[PW09] R. Pass and H. Wee. Black-box constructions of two-party protocols from one-way functions. In *TCC*, 2009. to appear.

[WW06] S. Wolf and J. Wullschleger. Oblivious transfer is symmetric. In *EUROCRYPT*, 2006.