Two-Round MPC: Information-Theoretic and Black-Box

Sanjam Garg[*]
University of California, Berkeley
sanjamg@berkeley.edu

Yuval Ishai[†]
Technion
yuvali@cs.technion.ac.il

Akshayaram Srinivasan
University of California, Berkeley
akshayaram@berkeley.edu

Abstract

We continue the study of protocols for secure multiparty computation (MPC) that require only two rounds of interaction. The recent works of Garg and Srinivasan (Eurocrypt 2018) and Benhamouda and Lin (Eurocrypt 2018) essentially settle the question by showing that such protocols are implied by the minimal assumption that a two-round oblivious transfer (OT) protocol exists. However, these protocols inherently make a non-black-box use of the underlying OT protocol, which results in poor concrete efficiency. Moreover, no analogous result was known in the information-theoretic setting, or alternatively based on one-way functions, given an OT correlations setup or an honest majority.

Motivated by these limitations, we study the possibility of obtaining information-theoretic and "black-box" implementations of two-round MPC protocols. We obtain the following results:

- **Two-round MPC from OT correlations.** Given an OT correlations setup, we get protocols that make a black-box use of a pseudorandom generator (PRG) and are secure against a malicious adversary corrupting an arbitrary number of parties. For a semi-honest adversary, we get similar information-theoretic protocols for branching programs.
- **New NIOT constructions.** Towards realizing OT correlations, we extend the DDH-based *non-interactive OT* (NIOT) protocol of Bellare and Micali (Crypto '89) to the malicious security model, and present new NIOT constructions from the Quadratic Residuosity Assumption (QRA) and the Learning With Errors (LWE) assumption.
- **Two-round black-box MPC with strong PKI setup.** Combining the two previous results, we get two-round MPC protocols that make a *black-box* use of any DDH-hard or QRA-hard group. The protocols can offer security against a malicious adversary, and require a PKI setup that depends on the number of parties and the size of computation, but not on the inputs or the identities of the participating parties.
- **Two-round honest-majority MPC from secure channels.** Given secure point-to-point channels, we get protocols that make a black-box use of a pseudorandom generator

[*]Research supported in part from 2017 AFOSR YIP Award, DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550-15-1-0274, and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the author and do not reflect the official policy or position of the funding agencies.

[†]Supported by ERC grant 742754 (project NTSC), ISF grant 1709/14, NSF-BSF grant 2015782, and a grant from the Ministry of Science and Technology, Israel and Department of Science and Technology, Government of India.

(PRG), as well as information-theoretic protocols for branching programs. These protocols can tolerate a semi-honest adversary corrupting a strict minority of the parties, where in the information-theoretic case the complexity is exponential in the number of parties.

1 Introduction

There is an enormous body of work on the round complexity of protocols for secure multiparty computation (MPC). While the feasibility of *constant-round* MPC has been established a long time ago [Yao86, BB89, BMR90], some of the most basic questions about the *exact* number of rounds required for MPC remained wide open until recently.

A single round of interaction is clearly insufficient to realize the standard notion of MPC. The focus of this work is on MPC protocols that require only two rounds. Two-round MPC protocols are not only interesting because of the quantitative aspect of minimizing the number of rounds, but also because of the following qualitative advantage. In a two-round MPC protocol, a party can send its first round messages and then go offline until all second-round messages are received and the output can be computed. (In fact, for two-round protocols over insecure channels, the first round messages can be publicly posted.) Moreover, the first round messages can be potentially reused for several computations in which the receiver's input remains the same. Indeed, in the two-party setting, such two-round protocols are sometimes referred to as "non-interactive secure computation" [IKO$^+$11].

The state of the art on two-round MPC can be briefly summarized as follows. Unless otherwise specified, we restrict our attention to *semi-honest* adversaries, who may non-adaptively corrupt an arbitrary subset of parties, and allow the protocols to use a common *random* string.

In the information-theoretic setting, 2-round protocols over secure point-to-point channels are known to exist with $t < n/3$ corrupted parties [IK00], leaving open the existence of similar protocols with an optimal threshold of $t < n/2$. These information-theoretic protocols, like all current general constant-round protocols in the information-theoretic setting, have complexity that grows polynomially with n and with the *branching program* size of the function being computed, and thus can only efficiently apply to rich but limited function classes such as NC^1, NL, or other log-space classes.

Settling for computational security, the above information-theoretic protocols imply (via the multi-party garbling technique of [BMR90]) similar protocols for *circuits*, capturing all polynomial-time computable functions, where the protocols only require a black-box use of any pseudorandom generator (PRG), or equivalently a one-way function. In this setting too, it was open whether the optimal[1] threshold of $t < n/2$ can be achieved.

Under stronger cryptographic assumptions, a lot of recent progress has been made on two-round MPC protocols that tolerate an arbitrary number of corrupted parties. The first such protocols required a public-key infrastructure (PKI) setup, where each party can post a public key before its input is known, and were based on the Learning With Errors (LWE) assumption via threshold fully homomorphic encryption [AJW11]. This was followed by protocols without PKI setup, first under indistinguishability obfuscation [GGHR14] or witness encryption [GLS15], and later under LWE via multi-key fully homomorphic encryption [MW16] or spooky encryption [DHRW16]. Using PKI

[1]Protocols that offer security with no honest majority imply oblivious transfer. Thus, they provably do not admit a *black-box* reduction to a PRG [IR89], and a non-black-box reduction would be considered a major breakthrough in cryptography.

setup, two-round protocols could also be constructed under the Decisional Diffie-Hellman (DDH) assumption via homomorphic secret sharing [BGI17, BGI$^+$18].

In recent works, a new general technique for collapsing rounds via "protocol garbling" [GS17] has been used by Garg and Srinivasan [GS18] and Benhamouda and Lin [BL18] to settle the minimal assumptions required for two-round MPC. These works show that general two-round MPC can be based on any two-round protocol for *oblivious transfer* (OT) [Rab81, EGL85], namely a protocol allowing a receiver to obtain only one of two bits held by a sender without revealing the identity of the chosen bit. This assumption is clearly necessary, since two-round OT is an instance of two-round general MPC.

Remaining challenges. Despite apparently settling the problem of two-round MPC, many challenges still remain. First and foremost, the recent OT-based protocols from [GS18, BL18] inherently make a *non-black-box* use of the underlying OT protocol. This results in poor concrete efficiency, which is unfortunate given the appealing features of two-round MPC discussed above. Second, the recent results leave open the possibility of obtaining information-theoretic security, or alternatively, computational security using symmetric cryptography, in other natural settings. These include protocols for the case of an *honest majority* ($t < n/2$) using secure point-to-point channels,[2] or alternatively protocols for dishonest majority based on an ideal OT oracle. Finally, the two-round MPC protocols from [GS18, BL18] did not seem to apply to the more general *client-server* setting, where only clients hold inputs and receive outputs, and communication only involves messages from clients to servers and from servers to clients.[3]

1.1 Our Contribution

In this work we address the above challenges, focusing mainly on the goal of constructing information-theoretic and "black-box" implementations of two-round MPC protocols. We obtain the following results:

Two-round MPC from OT correlations. We start by studying two-round MPC using an *OT correlations setup*, which can be viewed as a minimal[4] setup for MPC with no honest majority under assumptions that are weaker than OT. An OT correlation setup allows each pair of parties to share many independent instances of correlated randomness where party P_i gets a pair of random bits (or strings) (s_0, s_1) and party P_j gets a random bit b and the bit s_b. Using such an OT correlations setup, we get protocols that make a black-box use of a PRG and are secure against either a semi-honest[5] or malicious adversary corrupting an arbitrary number of parties. For a

[2] A recent work of Ananth, Choudhuri, Goel, and Jain [ACGJ18] obtains honest-majority, two-round MPC protocols from one-way functions satisfying the notion of security with abort against malicious adversaries. Our work was done in part following a public announcement of this result.

[3] An additional disadvantage of the protocols from [GS18, BL18] compared to most earlier protocols is that their communication complexity is always bigger than the circuit size of the function being computed. However, breaking this circuit size barrier under general assumptions such as OT would require a major breakthrough, regardless of round complexity.

[4] Two-round MPC was previously known to follow from a *global* correlated randomness setup that includes garbled circuits [CEMY09, IMO18] or truth-tables [IKM$^+$13] whose keys are secret-shared between all parties. Our setup assumption is weaker in that it only involves a simple *pairwise* correlation.

[5] Our protocol for semi-honest adversaries is expensive but not prohibitively so. With some simple optimizations, the online communication consists of roughly $1750 \cdot n^3$ standard garbled circuits, which is about 135 times the cost of

semi-honest adversary, we get similar *information-theoretic* protocols for branching programs.

This OT correlation setup can be implemented with good concrete efficiency via OT extension [IKNP03], requiring roughly 128 bits of communication per string-OT. Alternatively, the communication complexity of the setup can be made independent of the circuit size (at a much higher computational cost) by using homomorphic secret sharing based on LWE, DDH, or DCRA [BGI16, DHRW16, FGJI17, BCG$^+$17]. Finally, a fully non-interactive option for implementing the OT correlation setup is discussed next.

New NIOT constructions. An appealing method of realizing the OT correlation setup is via *non-interactive OT* (NIOT) [BM90]. An NIOT protocol is the OT analogue of non-interactive key exchange: it allows two parties to obtain a joint OT correlation via a simultaneous message exchange. We present several new constructions of NIOT. First, we extend the DDH-based construction from [BM90] to the malicious security model, improving over an earlier construction based on bilinear maps from [GS17]. Second, we present new NIOT constructions from the Quadratic Residuosity Assumption (QRA) and from LWE.

Two-round black-box MPC with strong PKI setup. Combining the protocols based on OT correlations and the NIOT constructions, we get two-round MPC protocols that make a *black-box* use of any DDH-hard or QRA-hard group. The protocols can offer security against a malicious adversary, and require a strong PKI setup that depends on the number of parties and the size of computation, but not on the inputs or the identity of the participating parties. This is arguably the first "black box" two-round MPC protocol that does not rely on an honest majority or a correlated randomness setup. Our DDH-based protocol can be compared with previous DDH-based two-round MPC protocols from [BGI$^+$18] that require a weaker PKI setup and have better asymptotic communication complexity, but make a non-black-box use of the underlying group except when there are n clients and 2 servers.

Two-round honest-majority MPC from secure channels. Given secure point-to-point channels, we get protocols that make a black-box use of a PRG, as well as information-theoretic protocols for branching programs. These protocols can tolerate a semi-honest adversary corrupting a strict minority of the parties, where in the information-theoretic case the complexity of the protocol grows quasi-polynomially with the number of parties. Our work leaves open the question of eliminating this slightly super-polynomial dependence as well as the question of obtaining similar results for malicious adversaries. This question has been resolved in the concurrent and independent work of Applebaum, Brakerski and Tsabary [ABT18].

From standard MPC to client-server MPC. Finally, we present a general (non-black-box) transformation that allows converting previous two-round MPC protocols (including the recent OT-based protocols from [GS18, BL18]) to the stronger client-server model. Concretely, we use a PRG to transform any n-party, two-round, MPC protocol with security against semi-honest adversaries corrupting an arbitrary subset of parties to a similar protocol with n clients and m servers, where in the first round each client sends a message to each server and in the second round each server sends a message to each client. The resulting protocol is secure against a semi-honest adversary that

the BMR protocol [BMR90], and the total number of OTs required by the setup is less than 7% of the communication.

corrupts an arbitrary subset of clients and a strict subset of the servers. This setting is particularly appealing when clients would like to be offline except when their input changes or they would like to receive an output.

1.2 Overview of Techniques

In this subsection, we describe the main techniques used to obtain our results.

1. We start with a high-level overview of the OT correlations model and describe the technical challenges in constructing a non-interactive OT protocol.
2. Later, we will show how to use OT correlations to make the compiler of Garg and Srinivasan [GS18] information theoretic. This gives efficient, two-round protocols in the OT correlations model with information theoretic security for branching programs and computational security for circuits making black-box use of a pseudorandom generator.
3. We then explain the main ideas in constructing a two-round, protocol in the honest majority setting with secure point-to-point channels.

OT Correlations Model. The OT correlation is modeled by a two-party ideal functionality. When this functionality is invoked by a (sender, receiver) pair, it samples three bits (s_0, s_1) and b uniformly at random and provides (s_0, s_1) to the sender and (b, s_b) to the receiver. For simplicity, we focus only for the case where sender's output (s_0, s_1) are bits as there are perfect, round-preserving reductions from bit OT correlations to string OT correlations (refer [BCS96, BCW03]). Given such OT correlations, there is an information theoretic, two-round OT protocol as follows. In the first round, the receiver sends $u = b \oplus c$ to the sender where c is the choice bit and in the second round, the sender computes $(x_0, x_1) = (m_0 \oplus s_u, m_1 \oplus s_{1 \oplus u})$ and sends them to the receiver. The receiver outputs $x_c \oplus r_b$.

Bellare-Micali Non-Interactive Oblivious Transfer. Bellare and Micali [BM90] gave an efficient, single-round protocol based on Decisional Diffie-Hellman (DDH) assumption [DH76] for computing OT correlations when the adversary corrupting either of the two parties is semi-honest. The protocol is in the common reference string model and is as follows. Let us assume that $\mathbb{G}$ is a DDH hard group and g is a generator. The CRS is an uniform group element X. The sender chooses $a \leftarrow \mathbb{Z}_p^*$ and sends $A = g^a$ to the receiver. The receiver chooses a random $b \leftarrow \mathbb{Z}_p^*$ and sends $(B_0, B_1) = (g^b, X/g^b)$ in a randomly permuted order. The sender computes (B_0^a, B_1^a) and outputs it and the receiver computes A^b and outputs it. The receiver's choice bit b is statistically hidden from an adversarial sender and the string s_{1-b} is computationally hidden from the receiver based on the DDH assumption. However, this protocol only works in the semi-honest model as there is no efficient way to extract the receiver's choice bit or the sender's correlations. In [GS17], Garg and Srinivasan additionally used Groth-Sahai proofs [GS08] to enable efficient extraction of the correlations from a malicious adversary but this construction relies on bilinear maps.

Our Construction of Non-Interactive Oblivious Transfer. Our approach of constructing non-interactive oblivious transfer is via a generalization of the dual-mode framework introduced in the work of Peikert, Vaikuntanathan and Waters [PVW08]. In the dual mode framework, the common reference string can be in one of two indistinguishable modes: namely, the receiver extraction mode or the sender extraction mode. In the receiver extraction mode, the CRS trapdoor enables the simulator to extract the receiver's correlation b and in the sender extraction mode, the it enables the simulator to extract the sender's correlation (s_0, s_1) from the malicious party. In either of the two modes, the secrets of the honest party are statistically hidden. We give efficient instantiations of this framework from DDH, Quadratic Residuocity assumption [GM82] and the Learning with Errors assumption [Reg05]. Our DDH and QR based constructions make black-box use of the underlying group. We stress that constructions of dual-mode cryptosystem in [PVW08] do not yield non-interactive oblivious transfer and we need to come up with new constructions. We refer the reader to Section 3.1 for the details.

Round-Collapsing Compiler in the OT Correlations Model. Independent works by Benhemouda and Lin [BL18] and Garg and Srinivasan [GS18] gave a "round-collapsing" compiler that takes an arbitrary multi-round MPC protocol and collapses it to two-rounds assuming the existence of a two-round oblivious transfer and garbled circuits. The compiler makes use of the code of the underlying protocol and thus, if the underlying protocol performs cryptographic operations then the resultant two-round protocol makes non-black box use of cryptography. In this work, we will use OT correlations to modify the compiler of [GS18] so that the resulting protocol makes black-box use of cryptography even if the underlying protocol performs cryptographic operations. Let us see how this is done.

We start by observing that OT correlations allow for perfect (resp., statistical) information-theoretic protocols in the presence of an arbitrary number of semi-honest (resp., malicious) corrupted parties. Hence, we we will round-collapse, perfectly/statistically secure protocols that are in the OT-hybrid model (e.g., [GMW87, Kil88, IPS08]). We first give a reduction from perfectly/statistically secure protocols in the OT-hybrid model to a perfectly/statistically secure protocols in the OT correlations model. This reduction has a property that all the OT correlations are generated before the actual execution of the protocol and the operations performed in the protocol are information theoretic. Another useful property is that number of OT correlations needed depends only the number of parties and the size of the computation to be performed and in particular, is independent of the actual inputs. At a high level, this reduction relies on the fact that OT correlations can be used to perform information theoretic OTs. Now, given such a protocol in the OT correlations model, we modify the compiler of Garg and Srinivasan to have a pre-processing phase where all the OT correlations needed for the underlying protocol and those consumed by the round-collapsing compiler are generated. Later, these OT correlations are used to perform information theoretic OTs both in the underlying protocol and the round-collapsing compiler. Additionally, we also replace the garbled circuits used in the round-collapsing compiler with a perfectly secure analogue, namely a so-called "decomposable randomized encodings" for low-depth circuits [IK00, AIK04]. With these changes to the [GS18] compiler, we get a perfectly secure two-round protocol in the OT correlations model for constant size functions. Later, we use a result from [BGI$^+$18] to bootstrap this to a perfectly secure, two-round protocol in the OT correlations model for NC^0 circuits. Two immediate corollaries of this result are a perfectly secure, two-round protocol in the OT correlations model for polynomial sized branching programs and a computationally secure, two-round protocol in the OT correlations model for arbitrary circuits making black-box use of a pseudorandom generator.

Two-round Protocol in the Honest Majority Setting. To construct a two-round protocol in the plain model (with secure point-to-point channels) when the adversary corrupts a strict minority of the parties, we use the same high level idea of the [GS18] compiler. That is, we take a larger round protocol secure with honest majority and round-collapse it to two-rounds. Two immediate issues arise: (1) The first issue is that the round-collapsing compiler requires the existence of two-round oblivious transfer, (2) the second issue is that round-collapsing compiler could only compress protocols in the presence of a broadcast channels and fails for protocols with secure channels. To address the first issue, we construct a perfectly secure, two-round OT protocol in the presence of honest majority (building on the work of [IKP10]) and to address the second issue, we give a generalization of the [GS18] compiler to compress protocols that may require secure channels. We then use this OT protocol in parallel with the round-collapsing compiler of [GS18] (enhanced to work for protocols with secure channels) to obtain a two-round protocol in the honest majority setting. However, the resulting communication complexity of the protocol grows super-polynomially with the number of parties n. Still, for constant n, the protocol is efficient.

2 Preliminaries

We recall some standard cryptographic definitions in this section. Let λ denote the security parameter. A function $\mu(\cdot) : \mathbb{N} \to \mathbb{R}^+$ is said to be negligible if for any polynomial $\mathsf{poly}(\cdot)$ there exists λ_0 such that for all $\lambda > \lambda_0$ we have $\mu(\lambda) < \frac{1}{\mathsf{poly}(\lambda)}$. We will use $\mathsf{negl}(\cdot)$ to denote an unspecified negligible function and $\mathsf{poly}(\cdot)$ to denote an unspecified polynomial function.

For a probabilistic algorithm A, we denote $A(x; r)$ to be the output of A on input x with the content of the random tape being r. When r is omitted, $A(x)$ denotes a distribution. For a finite set S, we denote $x \leftarrow S$ as the process of sampling x uniformly from the set S. We will use PPT to denote Probabilistic Polynomial Time algorithm.

2.1 Decomposable Randomized Encoding

We recall the definitions of randomized encoding [Yao86, IK00, AIK04].

Definition 2.1 (Randomized Encoding) *Let $f : \{0,1\}^n \to \{0,1\}^m$ be some function. We say that a function $\widehat{f} : \{0,1\}^n \times \{0,1\}^\rho \to \{0,1\}^m$ is a perfect randomized encoding of f if for every input $x \in \{0,1\}$, the distribution $\widehat{f}(x;r)$ induced by an uniform choice of $r \overset{\$}{\leftarrow} \{0,1\}^\rho$, encodes the string $f(x)$ in the following sense:*

- ***Correctness.*** *There exists a decoding algorithm* Dec *such that for every $x \in \{0,1\}^n$, it holds that:*
$$\Pr_{r \overset{\$}{\leftarrow} \{0,1\}^\rho} [\mathsf{Dec}(\widehat{f}(x;r)) = f(x)] = 1$$

- ***Privacy:*** *There exists a randomized algorithm S such that for every $x \in \{0,1\}^n$ and uniformly chosen $r \overset{\$}{\leftarrow} \{0,1\}^\rho$ it holds that*
$$S(f(x)) \textit{ is distributed identically to } \widehat{f}(x;r).$$

Definition 2.2 (Decomposable Randomized Encoding) *We say that $\widehat{f}(x;r)$ is decomposable if $\widehat{f}$ can be written as $\widehat{f}(x;r) = (\widehat{f}_0(r), \widehat{f}_1(x_1;r), \ldots, \widehat{f}_n(x_n;r))$ where $\widehat{f}_i$ is chooses between two vectors based on x_i , i.e., it can be written as $\mathbf{a}_{i,x_i}$ and $(\mathbf{a}_{i,0}, \mathbf{a}_{i,1})$ arbitrarily depend on the randomness r. We will use $\widehat{f}(;r)$ to denote $(\widehat{f}_0(r), (\mathbf{a}_{1,0}, \mathbf{a}_{1,1}), \ldots, (\mathbf{a}_{n,0}, \mathbf{a}_{n,1}))$.*

We will recall the following two constructions of randomized encoding.

Lemma 2.3 ([Kil88, IK00]) *Let $f : \{0,1\}^n \to \{0,1\}^m$ be a function computable in NC^0. Then f has a perfectly secure decomposable randomized encoding $\widehat{f}$ where the size of the encoding is $2^{O(d)}(n+m)$ where d is the depth of the circuit.*

Lemma 2.4 ([Yao86]) *Let $f : \{0,1\}^n \to \{0,1\}^m$ be a function computable by an arbitrary circuit. Assuming the existence of one-way functions, f has a computationally secure randomized encoding $\widehat{f}$.*

2.2 Universal Composability Framework

We work in the the Universal Composition (UC) framework [Can01] to formalize and analyze the security of our protocols. (Our protocols can also be analyzed in the stand-alone setting, using the composability framework of [Can00], or in other UC-like frameworks, like that of [PW00].) We give the details in Appendix A. We only focus on static (non-adaptive) adversaries but we note that our perfectly secure protocols are also secure against adaptive adversaries.

3 OT Correlations Functionality

In this section, we define the $\mathcal{F}_{\mathrm{OTCor}}$ functionality in Figure 1. Intuitively, the $\mathcal{F}_{\mathrm{OTCor}}$ functionality obtains a bit b from the receiver and samples two bits (s_0, s_1) randomly from $\{0,1\}$ and outputs (s_0, s_1) to the sender and s_b to the receiver.[6] In the definition, we focus on the case where the sender's output are just two bits (s_0, s_1) instead of two strings as there are efficient reductions from 1-out-of-2 string OTs to 1-out-of-2 bit OTs using self-intersecting codes or randomness extractors [BCS96, BCW03]. By abusing notation, we will interchangeably use the same functionality to sample two strings instead of two bits.

We first discuss two generic ways from literature for realizing $\mathcal{F}_{\mathrm{OTCor}}$ functionality and then give two new ways for realizing it.

OT Extension. We first note that any OT protocol can be used to realize $\mathcal{F}_{\mathrm{OTCor}}$ functionality. A more efficient way would be to use an oblivious transfer extension protocol [Bea96, IKNP03, ALSZ13, ALSZ15, KOS15]. Any OT extension protocol with security against semi-honest/malicious adversaries can be used to realize the $\mathcal{F}_{\mathrm{OTCor}}$ functionality against semi-honest/malicious adversaries. The only downside of this approach is that it involves multiple rounds of interaction (which is inherent if we want to make black-box use of cryptography [GMMM18]).

[6]Here, we let the receiver to choose the bit b and provide as input to the functionality. We can also work with a weaker formulation wherein the functionality can sample a random bit b. However, we chose this formulation as it will lead to concrete improvements in the cost of our two-round MPC protocols.

Parametrized with parties $P_1, \ldots, P_n$ and an adversary $\mathcal{S}$ controlling a subset of the parties. Let H be the set of parties not controlled by the adversary.

On receiving $(sid, \mathbf{receiver}, pid, b)$ (where $b \in \{0,1\}$) or $(sid, \mathbf{sender}, pid)$ from a party with id pid, store this message.

On receiving (sid, pid_1, pid_2) from a party with id pid_1, check if $(sid, \mathbf{receiver}, pid_2, b)$ and $(sid, \mathbf{sender}, pid_1)$ are stored. If not stored, then do nothing. Else, do the following:

- If both $pid_1, pid_2 \in H$, sample $(s_0, s_1) \xleftarrow{\$} \{0,1\}$, send (s_0, s_1) to the party pid_1 and s_b to the party pid_2.
- If $pid_1 \notin H$ but $pid_2 \in H$ then send the message $(\mathbf{sender}, pid_1)$ to S and receive (s_0, s_1) from S. Send s_b to the party pid_2.
- If $pid_1 \in H$ but $pid_2 \notin H$, send the message $(\mathbf{receiver}, pid_2)$ to S and receive s_b from S. Sample $s_{1-b} \xleftarrow{\$} \{0,1\}$ and send (s_0, s_1) to the party pid_1.
- If both $pid_1, pid_2 \notin H$, ignore the message.

Figure 1: OT Correlations Functionality $\mathcal{F}_{\mathrm{OTCor}}$.

Homomorphic Secret Sharing/ Threshold FHE. A reusable and a non-interactive approach to realize the weaker formulation wherein the receiver's choice bit is sampled randomly by the functionality is to use Homomorphic Secret Sharing (HSS) [BGI16, BGI17, BGI$^+$18, BCG$^+$17]. Using Homomorphic Secret Sharing, each party can generate a HSS encoding of a randomly chosen PRG seed and broadcasts this encoding to all other parties. When an OT correlation is to be generated, the parties (using the encodings) locally compute a functionality that expands the receiver's and the sender's PRG seed to the required length and samples the prescribed OT correlation from the expanded seeds. At the end of this local computation, the parties hold an additive secret sharing of the OT correlation and the actual correlation can be obtained non-interactively by sending these additive shares to the receiver. This approach is reusable as the encodings just needs to be sent once and can be resused to generate fresh correlations each time.[7] We also note that we can replace the above homomorphic secret sharing with any threshold FHE construction [MW16, DHRW16, BGG$^+$18]. The downsides of using HSS or threshold FHE is that they make non-black box use of one-way functions in expanding the short seed to a pseudorandom string and they are computationally expensive when compared to the OT extension. Additionally, HSS requires the use of secure channels between every pairs of parties.

In Section 3.1, we describe a non-interactive approach to realize $\mathcal{F}_{\mathrm{OTCor}}$. The advantage of this approach over HSS/threshold-FHE is that it makes black-box use of a groups where either DDH or QR is hard (we also provide an efficient construction from the LWE assumption). However, unlike HSS/threshold-FHE they are not reusable. In Section 3.2, we give a two-round, information theoretic protocol (with security against semi-honest adversaries) in the client-server model for realizing $\mathcal{F}_{\mathrm{OTCor}}$ when a majority of the servers are honest.

[7]The HSS constructions in [BGI16, BGI17, BGI$^+$18, BCG$^+$17] have a polynomial error probability and this might leak information about the correlations to an adversary. [BCG$^+$17] mentions two ways to prevent such leakages: either bootstrap random pads or use a punctured OT [BGI17]. We refer the reader to [BCG$^+$17] for the details.

3.1 Realizing $\mathcal{F}_{\mathrm{OTCor}}$: Non-Interactive Oblivious Transfer

In this subsection, we define a Non-Interactive Oblivious Transfer (NIOT) and show how to realize $\mathcal{F}_{\mathrm{OTCor}}$ functionality from NIOT. Later, we give constructions of NIOT based on the Decisional Diffie-Hellman (DDH), Quadratic Residuocity (QR) and Learning with Errors (LWE).

Definition. A Non-Interactive Oblivious Transfer (NIOT) is a tuple of algorithms $(\mathsf{K}_{\mathrm{R}}, \mathsf{K}_{\mathrm{S}}, \mathsf{Sen}, \mathsf{Rec}, \mathsf{out}_{\mathrm{S}}, \mathsf{out}_{\mathrm{R}})$ having the following syntax, correctness and security guarantees.

- K_{R} and K_{S} are randomized algorithms that take as input the security parameter (encoded in unary) and output a common random string σ along with some trapdoor information τ.
- Sen is a randomized algorithm that takes σ as input and outputs $\mathsf{msg}_{\mathrm{S}}$ along with secret randomness ω.
- Rec is a randomized algorithm that takes σ and a bit b as input and outputs $\mathsf{msg}_{\mathrm{R}}$ along with secret randomness ρ_b.
- $\mathsf{out}_{\mathrm{S}}$ is a deterministic algorithm that takes as input σ, $\mathsf{msg}_{\mathrm{R}}$ and the secret randomness ω and outputs two bits k_0, k_1.
- $\mathsf{out}_{\mathrm{R}}$ is a deterministic algorithm that takes as σ, $\mathsf{msg}_{\mathrm{S}}$ and the secret randomness ρ_b and outputs a bit k'_b.

Correctness. We require that for all $b \in \{0, 1\}$,

$$\begin{aligned}\Pr\big[k'_b = k_b \quad : \quad & (\sigma, \tau) \leftarrow \mathsf{K}_{\mathrm{R}}(1^\lambda), (\mathsf{msg}_{\mathrm{S}}, \omega) \leftarrow \mathsf{Sen}(\sigma), (\mathsf{msg}_{\mathrm{R}}, \rho_b) \leftarrow \mathsf{Rec}(\sigma, b),\\ & (k_0, k_1) \leftarrow \mathsf{out}_{\mathrm{S}}(\sigma, \omega, \mathsf{msg}_{\mathrm{R}}), k'_b \leftarrow \mathsf{out}_{\mathrm{R}}(\sigma, \rho_b, \mathsf{msg}_{\mathrm{S}})\big] \geq 1 - \mathsf{negl}(\lambda)\end{aligned}$$

Security. We require the following security properties to hold.

- **CRS Indistinguishability.** We require that

$$\Big\{\sigma : (\sigma, \tau) \leftarrow \mathsf{K}_{\mathrm{R}}(1^\lambda)\Big\} \overset{c}{\approx} \Big\{\sigma : (\sigma, \tau) \leftarrow \mathsf{K}_{\mathrm{S}}(1^\lambda)\Big\}$$

- **Sender Security.** We require that there exists a PPT algorithm $\mathsf{Ext}_{\mathrm{R}}$ such that for all non-uniform PPT adversarial Rec^* the following two distributions are statistically close.

$$\left\{\begin{array}{l}(\sigma, \tau) \leftarrow \mathsf{K}_{\mathrm{R}}(1^\lambda),\\ (\mathsf{msg}_{\mathrm{S}}, \omega) \leftarrow \mathsf{Sen}(\sigma),\\ \mathsf{msg}_{\mathrm{R}} \leftarrow \mathsf{Rec}^*(\sigma, \mathsf{msg}_{\mathrm{S}})\\ (k_0, k_1) \leftarrow \mathsf{out}_{\mathrm{S}}(\sigma, \omega, \mathsf{msg}_{\mathrm{R}}):\\ \text{Output } (\mathsf{msg}_{\mathrm{S}}, \mathsf{msg}_{\mathrm{R}}, k_0, k_1)\end{array}\right\} \overset{s}{\approx} \left\{\begin{array}{l}(\sigma, \tau) \leftarrow \mathsf{K}_{\mathrm{R}}(1^\lambda),\\ (\mathsf{msg}_{\mathrm{S}}, \omega) \leftarrow \mathsf{Sen}(\sigma),\\ \mathsf{msg}_{\mathrm{R}} \leftarrow \mathsf{Rec}^*(\sigma, \mathsf{msg}_{\mathrm{S}})\\ b' \leftarrow \mathsf{Ext}_{\mathrm{R}}(\sigma, \mathsf{msg}_{\mathrm{R}}, \tau):\\ (k_0, k_1) \leftarrow \mathsf{out}_{\mathrm{S}}(\sigma, \omega, \mathsf{msg}_{\mathrm{R}}),\\ \ell_{b'} := k_{b'}, \ell_{1-b'} \leftarrow \{0, 1\}:\\ \text{Output } (\mathsf{msg}_{\mathrm{S}}, \mathsf{msg}_{\mathrm{R}}, \ell_0, \ell_1).\end{array}\right\}$$

- **Receiver Security.** We require that there exists a PPT algrithm $\mathsf{Ext}_{\mathrm{S}}$ such that for all non-uniform PPT adversarial Sen^* and for all $b \in \{0,1\}$, the following two distributions are statistically close.

$$\left\{\begin{array}{l}(\sigma,\tau) \leftarrow \mathsf{K}_{\mathrm{S}}(1^\lambda),\\ (\mathsf{msg}_{\mathrm{R}},\rho_b) \leftarrow \mathsf{Rec}(\sigma,b),\\ \mathsf{msg}_{\mathrm{S}} \leftarrow \mathsf{Sen}^*(\sigma,\mathsf{msg}_{\mathrm{R}}),\\ k'_b \quad\leftarrow\quad \mathsf{out}_{\mathrm{R}}(\sigma,\rho_b,\mathsf{msg}_{\mathrm{S}}):\\ \text{Output } (\mathsf{msg}_{\mathrm{S}},\mathsf{msg}_{\mathrm{R}},k'_b)\end{array}\right\} \overset{s}{\approx} \left\{\begin{array}{l}(\sigma,\tau) \leftarrow \mathsf{K}_{\mathrm{S}}(1^\lambda),\\ (\mathsf{msg}_{\mathrm{R}},\rho_0,\rho_1) \ \leftarrow\ \mathsf{Ext}_{\mathrm{S}}(\sigma,\tau),\\ \mathsf{msg}_{\mathrm{S}} \leftarrow \mathsf{Sen}^*(\sigma,\mathsf{msg}_{\mathrm{R}}),\\ k'_b \quad\leftarrow\quad \mathsf{out}_{\mathrm{R}}(\sigma,\rho_b,\mathsf{msg}_{\mathrm{S}}):\\ \text{Output } (\mathsf{msg}_{\mathrm{S}},\mathsf{msg}_{\mathrm{R}},k'_b)\end{array}\right\}$$

3.1.1 NIOT $\Rightarrow \mathcal{F}_{\mathrm{OTCor}}$

In this subsection, we give a realization of the $\mathcal{F}_{\mathrm{OTCor}}$ functionality from any non-interactive oblivious transfer.

Theorem 3.1 *Assuming the existence of non-interactive oblivious transfer, there is a single round protocol for realizing $\mathcal{F}_{\mathrm{OTCor}}$ against malicious adversaries in the common reference string model.*

Construction. We give a construction realizing the $\mathcal{F}_{\mathrm{OTCor}}$ functionality in Figure 2.

Let $(\mathsf{K}_{\mathrm{R}},\mathsf{K}_{\mathrm{S}},\mathsf{Sen},\mathsf{Rec},\mathsf{out}_{\mathrm{S}},\mathsf{out}_{\mathrm{R}})$ be a non-interactive oblivious transfer.

Inputs: Party P_i for $i \in [n]$, receives a session id *sid*.

Common Reference String: For every $i,j \in [n]$, sample $(\sigma_{i,j},\tau_{i,j}) \leftarrow \mathsf{K}_{\mathrm{R}}(1^\lambda)$. Publish $\{\sigma_{i,j}\}_{i,j\in[n]}$ as the common reference string.

Let us assume that P_i is the sender and P_j is the receiver.

Message sent by $P_i \to P_j$: Compute $(\mathsf{msg}_{\mathrm{S}},\omega) \leftarrow \mathsf{Sen}(\sigma_{i,j})$ and send $\mathsf{msg}_{\mathrm{S}}$ to P_j.

Message sent by $P_j \to P_i$: On input $b \in \{0,1\}$, compute $(\mathsf{msg}_{\mathrm{R}},\rho_b) \leftarrow \mathsf{Rec}(\sigma_{i,j},b)$. Send $\mathsf{msg}_{\mathrm{R}}$ to P_i.

Computation: P_i sets $(s_0,s_1) := \mathsf{out}_{\mathrm{S}}(\sigma_{i,j},\omega,\mathsf{msg}_{\mathrm{R}})$. P_j sets $s_b := \mathsf{out}_{\mathrm{R}}(\sigma_{i,j},\rho_b,\mathsf{msg}_{\mathrm{S}})$.

Figure 2: Realizing the $\mathcal{F}_{\mathrm{OTCor}}$ functionality

Description of the Simulator. We assume that $\mathcal{A}$ is static and hence the set of honest parties H is known before the execution of the protocol. Recall the properties of $\mathsf{Ext}_{\mathrm{R}}$ and $\mathsf{Ext}_{\mathrm{S}}$ from the definition of non-interactive oblivious transfer.

Simulating the CRS. For every $i \in [n]$,

- If $P_i \in H$, sample $(\sigma_{i,j},\tau_{i,j}) \leftarrow \mathsf{K}_{\mathrm{R}}(1^\lambda)$ for every $j \in [n] \setminus \{i\}$.
- If $P_i \notin H$, sample $(\sigma_{i,j},\tau_{i,j}) \leftarrow \mathsf{K}_{\mathrm{S}}(1^\lambda)$ for every $j \in [n] \setminus \{i\}$.

Publish $\{\sigma_{i,j}\}_{i,j\in[n]}$ as the common reference string.

Simulating the interaction with $\mathcal{Z}$. For every input value for the set of corrupted parties that S receives from $\mathcal{Z}$, S writes that value to $\mathcal{A}$'s input tape. Similarly, the output of $\mathcal{A}$ is written as the output on S's output tape.

Simulating the interaction with $\mathcal{A}$: For every concurrent interaction with the session identifier *sid* that $\mathcal{A}$ may start and for every choice of sender P_i and the receiver P_j, the simulator does the following:

- **Both $P_i, P_j \in H$:**
  1. Compute $(\mathsf{msg}_{\mathrm{S}}, \omega) \leftarrow \mathsf{Sen}(\sigma_{i,j})$ on behalf of P_i and send $\mathsf{msg}_{\mathrm{S}}$ to P_j.
  2. Sample $b \leftarrow \{0,1\}$ and compute $(\mathsf{msg}_{\mathrm{R}}, \rho_b) \leftarrow \mathsf{Rec}(\sigma_{i,j}, b)$ on behalf of P_j. Send $\mathsf{msg}_{\mathrm{R}}$ to P_i.
- **$P_i \in H$ and $P_j \notin H$:**
  1. Compute $(\mathsf{msg}_{\mathrm{S}}, \omega) \leftarrow \mathsf{Sen}(\sigma_{i,j})$ on behalf of P_i and send $\mathsf{msg}_{\mathrm{S}}$ to $\mathcal{A}$.
  2. $\mathcal{A}$ outputs $\mathsf{msg}_{\mathrm{R}}$.
  3. Run $b' \leftarrow \mathsf{Ext}_{\mathrm{R}}(\sigma_{i,j}, \tau_{i,j}, \mathsf{msg}_{\mathrm{R}})$.
  4. Compute $(s_0, s_1) := \mathsf{out}_{\mathrm{S}}(\sigma_{i,j}, \omega, \mathsf{msg}_{\mathrm{R}})$.
  5. Send $s_{b'}$ to the $\mathcal{F}_{\mathrm{OTCor}}$ functionality and output whatever $\mathcal{A}$ outputs.
- **$P_i \notin H$ and $P_j \in H$:**
  1. Compute $(\mathsf{msg}_{\mathrm{R}}, \rho_0, \rho_1) \leftarrow \mathrm{S}(\sigma_{i,j}, \tau_{i,j})$ and send $\mathsf{msg}_{\mathrm{R}}$ to $\mathcal{A}$.
  2. $\mathcal{A}$ outputs $\mathsf{msg}_{\mathrm{S}}$.
  3. Compute $s_b := \mathsf{out}_{\mathrm{R}}(\sigma_{i,j}, \rho_b, \mathsf{msg}_{\mathrm{S}})$ for all $b \in \{0,1\}$.
  4. Send (s_0, s_1) to the $\mathcal{F}_{\mathrm{OTCor}}$ functionality and output whatever $\mathcal{A}$ outputs.

Lemma 3.2 *Assuming the security of non-interactive oblivious transfer, for every $\mathcal{Z}$ that obeys the rules of interaction for UC security we have* $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}} \overset{c}{\approx} \mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$.

Proof We now show that no environment can distinguish the real world execution with adversary $\mathcal{A}$ and an ideal world execution with adversary S. We consider three cases.

- **Case-1:** $P_i, P_j \in H$. Note that $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ is identically distributed to $(\{\sigma_{i,j}\}_{i,j\in[n]}, \mathsf{msg}_{\mathrm{S}}, \mathsf{msg}_{\mathrm{R}}, (\ell_0, \ell_1), \ell_b)$ where $\mathsf{msg}_{\mathrm{S}}$ and $\mathsf{msg}_{\mathrm{R}}$ are obtained as in the description of the simulator and $(\ell_0, \ell_1) \leftarrow \{0,1\}$. $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ is identically distributed to $(\{\sigma_{i,j}\}_{i,j\in[n]}, \mathsf{msg}_{\mathrm{S}}, \mathsf{msg}_{\mathrm{R}}, (s_0, s_1), s_b)$ where $(\mathsf{msg}_{\mathrm{R}}, \rho_b) \leftarrow \mathsf{Rec}(\sigma_{i,j}, b)$, $(\mathsf{msg}_{\mathrm{S}}, \omega) \leftarrow \mathsf{Sen}(\sigma_{i,j})$ and $(s_0, s_1) \leftarrow \mathsf{out}_{\mathrm{S}}(\sigma_{i,j}, \omega, \mathsf{msg}_{\mathrm{R}})$ We show through a sequence of hybrids that $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ is computationally indistinguishable to $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.

  $\underline{\mathsf{Hybrid}_1}$: This hybrid is same as $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ except that we choose a random $\ell_{1-b} \leftarrow \{0,1\}$ and the sender outputs it instead of s_{1-b}. Hybrid_1 is statistically close to $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ from the sender security of non-interactive oblivious transfer.

$\underline{\mathsf{Hybrid}_2}$: This hybrid is same as Hybrid_1 except that we generate $\sigma_{i,j}$ as $(\sigma_{i,j}, \tau_{i,j}) \leftarrow \mathsf{K}_{\mathrm{S}}(1^\lambda)$. Hybrid_1 is computationally indistinguishable to Hybrid_1 from the CRS indistinguishability.

$\underline{\mathsf{Hybrid}_3}$: This hybrid is same as Hybrid_2 except that the receiver's message encodes $1 - b$ instead of b. Hybrid_3 is statistically close to Hybrid_2 from receiver security of non-interactive oblivious transfer.

$\underline{\mathsf{Hybrid}_4}$: This hybrid is same as Hybrid_3 except that we generate $\sigma_{i,j}$ as $(\sigma_{i,j}, \tau_{i,j}) \leftarrow \mathsf{K}_{\mathrm{R}}(1^\lambda)$. Again, Hybrid_4 is computationally indistinguishable to Hybrid_3 from the CRS indistinguishability.

$\underline{\mathsf{Hybrid}_5}$: This hybrid is same as Hybrid_4 except that we choose a random $\ell_b \leftarrow \{0, 1\}$ and the sender outputs it instead of s_b. Hybrid_4 is statistically close to Hybrid_4 from the sender security of non-interactive oblivious transfer.

$\underline{\mathsf{Hybrid}_6}$: This hybrid is same as Hybrid_5 except that we generate $\sigma_{i,j}$ as $(\sigma_{i,j}, \tau_{i,j}) \leftarrow \mathsf{K}_{\mathrm{S}}(1^\lambda)$. Again, Hybrid_5 is computationally indistinguishable to Hybrid_6 from the CRS indistinguishability.

$\underline{\mathsf{Hybrid}_7}$: This hybrid is same as Hybrid_6 except that the receiver's message encodes a randomly chosen c instead of b. Hybrid_3 is statistically close to Hybrid_2 from receiver security of non-interactive oblivious transfer. Notice that Hybrid_5 is distributed identically to $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.

- **Case-2:** $P_i \in H$ and $P_j \notin H$. It follows from the sender security of non-interactive oblivious transfer that $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ is statistically close to $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$.
- **Case-3:** $P_i \notin H$ and $P_j \in H$. It follows from the receiver security of non-interactive oblivious transfer that $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ is statistically close to $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$.

■

3.1.2 NIOT from Decisional Diffie-Hellman

In this subsection we give a construction of non-interactive oblivious transfer from the Decisional Diffie-Hellman (DDH) assumption. Bellare and Micali in [BM90] gave a protocol in the semi-honest setting from DDH assumption. Garg and Srinivasan [GS17] gave a protocol against malicious adversaries from bilinear maps. We review the DDH assumption below.

Decisional Diffie-Hellman Assumption [DH76]. Let $\mathsf{Setup}_{\mathrm{DDH}}$ be a randomized algorithm that takes a security parameter as input and outputs $(p, \mathbb{G}, g)$ such that p is a prime, $\mathbb{G}$ is a descriptions of group of order p and g is a random generator of $\mathbb{G}$.

Definition 3.3 (Decisional Diffie-Hellaman Assumption) *We say the Decisional Diffie-Hellman holds for the group generator* $\mathsf{Setup}_{\mathrm{DDH}}$ *if for all non-uniform polynomial time adversaries* $\mathcal{A}$ *we* *have*

$$\Big| \Pr\left[(p,\mathbb{G},g) \leftarrow \mathsf{Setup}_{\mathrm{DDH}}(1^\lambda); x,y \leftarrow \mathbb{Z}_p^* : \mathcal{A}(p,\mathbb{G},g,g^x,g^y,g^{xy}) = 1\right] -$$
$$\Pr\left[(p,\mathbb{G},g) \leftarrow \mathsf{Setup}_{\mathrm{DDH}}(1^\lambda); x,y,z \leftarrow \mathbb{Z}_p^* : \mathcal{A}(p,\mathbb{G},g,g^x,g^y,g^z) = 1\right] \Big| \leq \mathsf{negl}(\lambda)$$

Theorem 3.4 *Assuming the DDH assumption, there exists a construction of non-interactive oblivious transfer.*

We give our construction in Figure 3.

Correctness. We first argue correctness of the construction given in Figure 3. Note that $(\overline{g}_b)^r = (g_b^{s_b}h_b^{t_b})^r = (g_b^r)^{s_b}(h_b^r)^{t_b} = g^{s_b}h^{t_b}$. Thus, it follows that k_b is indeed the first bit of $(\overline{g}_b)^r$.

Security. CRS indistinguishability follows directly from the DDH assumption. We show sender and receiver security.

- **Sender Security.** We give details on the Ext_R algorithm. On input $\sigma := (g_0, h_0, g_1, h_1)$, trapdoor $\tau := (x_0, x_1) := (z, zy/x)$ and an adversarially generated message (g, h), it checks if $x_0 = x_1$. If yes, it aborts. If it does not abort, it checks if $h = g^{x_0}$. If that is the case, it outputs 0. Else, it checks if $h = g^{x_1}$ in which case it outputs 1. If $h \neq g^{x_0}$ or if $h \neq g^{x_1}$, it outputs 0.

  We first notice that since x and y are sampled randomly, the probability that Ext_R aborts is negligible. So, we can condition on the event that Ext_R does not abort (or, in other words, $x \neq y$). We consider the following cases.

  - **Case-1:** $h = g^{x_0}$. In this case, it is sufficient to show that for randomly chosen $s_1, t_1 \leftarrow \mathbb{Z}_p^*$, $g^{s_1}h^{t_1}$ is distributed uniformly in $\mathbb{G}$ conditioned on $g_1^{s_1}h_1^{t_1}$. Let $g := g_0^r$ and $h_0 := g_0^z$. Since, $h = g^{x_0}$, we have $h := h_0^r$. Thus, $g^{s_1}h^{t_1} := g_0^{rs_1}h_0^{rt_1}$. Notice that since $x \neq y$, for a randomly chosen $s_1, t_1 \leftarrow \mathbb{Z}_p^*$, $xs_1 + zyt_1$ and $rs_1 + zrt_1$ are random and independent of each other. It now follows that $g_1^{s_1}h_1^{t_1} := g_0^{xs_1+zyt_1}$ and $g^{s_1}h^{t_1} := g_0^{rs_1+zrt_1}$ are random and independent of each other.
  - **Case-2:** $h = g^{x_1}$. It can be shown via an identical argument to Case-1 that for a randomly chosen $s_0, t_0 \leftarrow \mathbb{Z}_p^*$, $g^{s_0}h^{t_0}$ is distributed uniformly in $\mathbb{G}$ conditioned on $g_0^{s_0}h_0^{t_0}$.
  - **Case-3:** $h \neq g^{x_0}$ and $h \neq g^{x_1}$. In this case, let $g := g_0^r$ and $h := h_0^{r'}$ for some $r \neq r'$ since $h \neq g^{x_0}$. As in Case-1, it can be easily seen that for a randomly chosen $s_1, t_1 \leftarrow \mathbb{Z}_p^*$ $xs_1 + zyt_1$ and $rs_1 + zr't_1$ are random and independent of each other unless $r := x$ and $r' = y$. If $r := x$ and $r' = y$ then $h := g^{x_1}$ which contradicts our assumption.

- **Receiver Security.** We give details on the Ext_S algorithm. On input $\sigma := (g_0, h_0, g_1, h_1)$ and the trapdoor $\tau := x$, Ext_S chooses $r \leftarrow \mathbb{Z}_p^*$ and outputs $\mathsf{msg}_\mathrm{R} = (g, h) := (g_0^r, h_0^r)$ and sets $\rho_0 := (r, 0)$ and $\rho_1 := (r/x, 1)$.

  Notice that msg_R is identically distributed to $\mathsf{Rec}(\sigma, b)$ for any $b \in \{0,1\}$. Let $(\overline{g}_0, \overline{g}_1)$ be any adversarially chosen msg_S. Let s_0, s_1, t_0, t_1 be such that $\overline{g}_b := g_b^{s_b}h_b^{t_b}$. It is sufficient to prove that $g^{s_1}h^{t_1} := (\overline{g}_1)^{r/x}$. Note that $g_1 := g_0^x$ and $h_1 := h_0^x$. Thus, $(\overline{g}_1)^{r/x} := (g_1^{s_1}h_1^{t_1})^{r/x} := (g_0^r)^{s_1}(h_0^r)^{t_1} := g^{s_1}h^{t_1}$.

- $\mathsf{K}_{\mathrm{R}}(1^\lambda)$:

  1. $(p, \mathbb{G}, g) \leftarrow \mathsf{Setup}_{\mathrm{DDH}}$.
  2. Set $g_0 := g$ and choose $z \leftarrow \mathbb{Z}_p^*$.
  3. Set $h_0 := g_0^z$.
  4. Choose $x, y \leftarrow \mathbb{Z}_p^*$.
  5. Set $g_1 := g_0^x$ and $h_1 := h_0^y$.
  6. Output $\sigma := (g_0, h_0, g_1, h_1)$ and the trapdoor $\tau := (z, zy/x)$.

- $\mathsf{K}_{\mathrm{S}}(1^\lambda)$:

  1. $(p, \mathbb{G}, g) \leftarrow \mathsf{Setup}_{\mathrm{DDH}}$.
  2. Set $g_0 := g$ and choose $h_0 \leftarrow \mathbb{G}$.
  3. Choose $x \leftarrow \mathbb{Z}_p^*$.
  4. Set $g_1 := g_0^x$ and $h_1 := h_0^x$.
  5. Output $\sigma := (g_0, h_0, g_1, h_1)$ and the trapdoor $\tau := x$.

- $\mathsf{Sen}(\sigma)$:

  1. Choose $s_0, t_0, s_1, t_1 \leftarrow \mathbb{Z}_p^*$.
  2. Output $\mathsf{msg}_{\mathrm{S}} := (g_0^{s_0} h_0^{t_0}, g_1^{s_1} h_1^{t_1})$ and secret randomness $\omega := (s_0, t_0, s_1, t_1)$.

- $\mathsf{Rec}(\sigma, b)$:

  1. Choose $r \leftarrow \mathbb{Z}_p^*$.
  2. Output $\mathsf{msg}_{\mathrm{R}} := (g_b^r, h_b^r)$ and the secret randomness $\rho_b := (r, b)$.

- $\mathsf{out}_{\mathrm{S}}(\sigma, \omega, \mathsf{msg}_{\mathrm{R}})$:

  1. Parse $\mathsf{msg}_{\mathrm{R}}$ as (g, h) and ω as (s_0, t_0, s_1, t_1).
  2. Output k_0 and k_1 to be the first bit of $g^{s_0} h^{t_0}$ and $g^{s_1} h^{t_1}$ respectively.

- $\mathsf{out}_{\mathrm{R}}(\sigma, \rho_b, \mathsf{msg}_{\mathrm{S}})$:

  1. Parse $\mathsf{msg}_{\mathrm{S}}$ as $(\overline{g}_0, \overline{g}_1)$ and ρ_b as (r, b).
  2. Output the first bit of $(\overline{g}_b)^r$.

Figure 3: Non-Interactive Oblivious Transfer from DDH

3.1.3 NIOT from Quadratic Residuocity

In this section we present a construction of non-interactive oblivious transfer from the quadratic residuocity (QR) assumption. We will begin by reviewing the assumption, then describe the construction, and finally prove its correctness and security.

Notations For a positive integer N, we use $\mathcal{J}(N)$ to denote the set $\{x \in \mathbb{Z}/N\mathbb{Z} : \left(\frac{x}{N}\right) = 1\}$, where $\left(\frac{x}{N}\right)$ is the Jacobi symbol of x in $\mathbb{Z}/N\mathbb{Z}$. We use $\mathcal{QR}(N)$ to denote the set of quadratic residues in $\mathcal{J}(N)$. The security of our scheme is based on the following computational assumption.

Definition 3.5 (Quadratic Residuocity (QR) Assumption [GM82]) *Let* $\mathsf{QRgen}(\cdot)$ *be a PPT algorithm that generates two equal size primes* p, q *and* $N = pq$. *The following two distributions are computationally indistinguishable:*

$$\left\{(p,q,N) \leftarrow \mathsf{QRgen}(1^\lambda); V \leftarrow \mathcal{QR}(N) : (N,V)\right\} \overset{c}{\approx} \left\{(p,q,N) \leftarrow \mathsf{QRgen}(1^\lambda); V \leftarrow \mathcal{J}(N) \setminus \mathcal{QR}(N) : (N,V)\right\}$$

In the construction and the proof of security, we make use of the the notion IBE compatible algorithm proved in [BGH07].

Definition 3.6 ([BGH07]) *Let* $\mathcal{Q}$ *be a deterministic algorithm that takes as input* (N, S, R) *where* $N \in \mathbb{Z}^+$ *and* $R, S \in \mathbb{Z}/N\mathbb{Z}$. *The algorithm outputs two polynomials* $f, g \in \mathbb{Z}/N\mathbb{Z}[x]$. *We say that* $\mathcal{Q}$ *is IBE-compatible if the following two conditions hold:*

1. *(Condition 1) If* S *and* R *are quadratic residues then* $f(s)g(r)$ *is a quadratic residue for all square roots* r *of* R *and* s *of* S.
2. *(Condition 2) If* S *is a quadratic residue then* $f(s)f(-s)R$ *is a quadratic residue for all square roots* s *of* S.

Boneh et al. [BGH07] showed a concrete instantiation of such an IBE-compatible algorithm.

Theorem 3.7 *Assuming the Quadratic Residuocity assumption, there exists a construction of non-interactive oblivious transfer.*

The Construction. We give the construction of non-interactive oblivious transfer in Figure 4.

Correctness. We start with the correctness proof. Notice that if $b = 0$ then msg_R is a quadratic residue and otherwise, $u \cdot \mathsf{msg}_R$ is a quadratic residue. Let us first consider the case where msg_R is a quadratic residue. In that case, Condition 1 in Lemma 3.6 implies that $\left(\frac{f(s)}{N}\right) = \left(\frac{g(r)}{N}\right)$. Hence, $k_0' = k_0$. A similar argument can be used to show that if $u \cdot \mathsf{msg}_R$ is a quadratic residue then $k_1' = k_1$.

CRS Indistinguishability. The CRS indistinguishability property follows directly from quadratic residuocity assumption.

```
• $\mathsf{K}_{\mathrm{R}}(1^\lambda)$ :
    1. $(p,q,N) \leftarrow \mathsf{QRgen}(1^\lambda)$..
    2. Sample a random $u \leftarrow \mathcal{J}(N) \setminus \mathcal{QR}(N)$.
    3. Output $\sigma := (N,u), \tau := (p,q)$.

• $\mathsf{K}_{\mathrm{S}}(1^\lambda)$:
    1. $(p,q) \leftarrow \mathsf{QRgen}(1^\lambda)$.
    2. Sample a random $u \leftarrow \mathcal{QR}(N)$.
    3. Output $\sigma := (N,u), \tau := (p,q)$.

• $\mathsf{Sen}(\sigma)$:
    1. Pick a random $s \in \mathbb{Z}/N\mathbb{Z}$.
    2. $S := s^2$.
    3. Output $\mathsf{msg}_{\mathrm{S}} := S, \omega := s$.

• $\mathsf{Rec}(\sigma, b)$:
    1. Pick a random $r \in \mathbb{Z}/N\mathbb{Z}$.
    2. If $b = 0$, let $\mathsf{msg}_{\mathrm{R}} := r^2$, otherwise let $\mathsf{msg}_{\mathrm{R}} := r^2 u$.
    3. Output $\mathsf{msg}_{\mathrm{R}}$ and $\rho_b := (r, b, \mathsf{msg}_{\mathrm{R}})$.

• $\mathsf{out}_{\mathrm{S}}(\sigma, \omega, \mathsf{msg}_{\mathrm{R}})$:
    1. Parse $\omega$ as $s$, and let $S := s^2$.
    2. $(f,g) \leftarrow \mathcal{Q}(N, S, \mathsf{msg}_{\mathrm{R}})$, $(\bar{f}, \bar{g}) \leftarrow \mathcal{Q}(N, S, u \cdot \mathsf{msg}_{\mathrm{R}})$.
    3. Output $k_0 := \left(\frac{f(s)}{N}\right), k_1 := \left(\frac{\bar{f}(s)}{N}\right)$.

• $\mathsf{out}_{\mathrm{R}}(\sigma, \rho_b, \mathsf{msg}_{\mathrm{S}})$:
    1. Parse $\rho_b$ as $(r, b, \mathsf{msg}_{\mathrm{R}})$; parse $\mathsf{msg}_{\mathrm{S}}$ as $S$.
    2. If $b = 0$, let $(f,g) \leftarrow \mathcal{Q}(N, S, r^2)$ and $k'_b := \left(\frac{g(r)}{N}\right)$;
       otherwise let $(\bar{f}, \bar{g}) \leftarrow \mathcal{Q}(N, S, (ru)^2)$ and $k'_b := \left(\frac{\bar{g}(ru)}{N}\right)$.
    3. Output $k'_b$.
```

Figure 4: Non-Interactive Oblivious Transfer from QR

Sender Security. We first give the description of the extractor $\mathsf{Ext}_{\mathrm{R}}$. On input msg_R, the extractor uses the trapdoor $\tau = (p,q)$ to check if msg_R is a quadratic residue. It outputs $b' = 0$ if it is the case and 1 otherwise. We now need to show that $k_{1-b'}$ is statistically indistinguishable to random and this follows directly from the following lemma given in [BGH07].[8]

[8]The lemma in [BGH07] was shown only for $R \in \mathcal{J}(N)$. We extend it to arbitrary $R \notin \mathcal{QR}(N)$.

Lemma 3.8 ([BGH07]) *Let $N = pq$ be a QR modulus, $X \in \mathcal{QR}(N)$ and $R \notin \mathcal{QR}(N)$. Let x be a random variable uniformly chosen among the four square roots of X. Let f be a polynomial such that $f(x)f(-x)R$ is a quadratic residue for all four values of x. Then, $\left(\frac{f(x)}{N}\right)$ is uniformly distributed in $\{\pm 1\}$.*

Proof Some parts of the proof are taken verbatim from [BGH07]. Let x, x' be two square-roots of X such that $x = x' \mod p$ and $x = -x' \mod q$. Then, the four square roots of X are $\{\pm x, \pm x'\}$. By definition, we have that $\left(\frac{f(x)}{p}\right) = \left(\frac{f(x')}{p}\right)$ and $\left(\frac{f(x')}{q}\right) = \left(\frac{f(-x)}{q}\right)$. Also, from the fact that $f(x)f(-x)R$ is a quadratic residue, we have that $\left(\frac{f(x)}{p}\right)\left(\frac{f(-x)}{p}\right)\left(\frac{R}{p}\right) = 1$ and $\left(\frac{f(x)}{q}\right)\left(\frac{f(-x)}{q}\right)\left(\frac{R}{q}\right) = 1$. Since $R \notin \mathcal{QR}(N)$ either $\left(\frac{R}{p}\right) = -1$ or $\left(\frac{R}{q}\right) = -1$. We consider two cases:

- **Case-1:** $\left(\frac{R}{q}\right) = -1$. In this case, $\left(\frac{f(x)}{q}\right) = -\left(\frac{f(-x)}{q}\right) = -\left(\frac{f(x')}{q}\right)$. Thus, $\left(\frac{f(x)}{N}\right) = -\left(\frac{f(x')}{N}\right)$. Similarly, one can show that $\left(\frac{f(-x)}{N}\right) = -1\left(\frac{f(-x')}{N}\right)$. Thus, among $f(x), f(x'), f(-x), f(-x')$, the first two have different Jacobi symbols and the last two have different Jacobi symbols modulo N. Thus, $\left(\frac{f(x)}{N}\right)$ is uniformly distributed over $\{\pm 1\}$.

- **Case-2:** $\left(\frac{R}{p}\right) = -1$. In this case, $\left(\frac{f(x)}{p}\right) = -\left(\frac{f(-x)}{p}\right) = -\left(\frac{f(-x')}{p}\right)$. Thus, $\left(\frac{f(x)}{N}\right) = -\left(\frac{f(-x')}{N}\right)$. Similarly, one can show that $\left(\frac{f(x')}{N}\right) = -\left(\frac{f(-x)}{N}\right)$. Thus, among $f(x), f(-x'), f(-x), f(x')$, the first two have different Jacobi symbols and the last two have different Jacobi symbols modulo N. Thus, $\left(\frac{f(x)}{N}\right)$ is uniformly distributed over $\{\pm 1\}$.

■

Receiver Security. We first give the description of the extractor Ext_S. On input σ, τ, it uses τ to find the square root u' of u. It samples a random r and sets $\mathsf{msg}_R = r^2 u$, $\rho_0 = ru'$ and $\rho_1 = r$. It is easy to see that this extractor satisfies the receiver security definition.

3.1.4 NIOT from Learning with Errors

In this section, we will provide a construction of non-interactive oblivious transfer from Learning with Errors (LWE) assumption.

We will begin by recalling the LWE assumption. In the following description, we will use n as the security parameter.

Lemma 3.9 (Lattice Trapdoors [Ajt99, GPV08, MP12]) *There is an efficient randomized algorithm $\mathsf{TrapSamp}(1^n, 1^m, q)$ that, given any integers $n \geq 1$, $q \geq 2$, and sufficiently large $m = \Omega(n \log q)$, outputs a parity check matrix $\mathbf{A} \in \mathbb{Z}_q^{nm}$ and a trapdoor matrix $\mathbf{T} \in \mathbb{Z}^{mm}$ such that the distribution of $\mathbf{A}$ is $\mathsf{negl}(n)$-close to uniform.*

Definition 3.10 (B-Bounded Distribution) *A family of distribution $\chi := \{\chi_n\}_{n\in\mathbb{N}}$ is said to be $B = B(n) \in \mathbb{N}$-bounded for every n, $\Pr[\chi \in \{-B, -B+1, \ldots, B-1, B\}] = 1$.*

Definition 3.11 (Learning with Errors [Reg05, Pei09]) *For an integer $q = q(n) \geq 2$ and a B-bounded error distribution $\chi = \chi(n)$ over $\mathbb{Z}_q$, the learning with errors problem $\mathsf{dLWE}_{n,m,q,\chi}$ is to distinguish between the following pairs of distributions:*

$$\{\mathbf{A}, \mathbf{A}^T\mathbf{s} + \mathbf{e}\} \textit{ and } \{\mathbf{A}, \mathbf{y}\}$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, s \leftarrow \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi^m$ and $\mathbf{y} \leftarrow \mathbb{Z}_q^m$.

Theorem 3.12 *Assuming LWE, there exists a construction of non-interactive oblivious transfer.*

The Construction. We give the construction of non-interactive oblivious transfer in Figure 5. In the construction, we will fix an integer $2\sqrt{q} \leq R \leq \frac{q}{4mB}$ and denote $\mathcal{R} := [0, R-1]$.

Correctness. The receiver output k_b' is the msb of $\mathbf{r}_b^T\mathbf{A}^T\mathbf{s}_R$ and k_b output by the sender is the msb of $\mathbf{r}_b^T\mathbf{A}^T\mathbf{s}_R + \mathbf{r}_b^T\mathbf{e}_R$. Note that $|\mathbf{r}_b^T\mathbf{e}_R| \leq mRB \leq q/4$. Thus, msb of $\mathbf{r}_b^T\mathbf{A}^T\mathbf{s}_R$ is same as the msb of $\mathbf{r}_b^T\mathbf{A}^T\mathbf{s}_R + \mathbf{r}_b^T\mathbf{e}_R$.

CRS Indistinguishability. The CRS indistinguishability follows directly from the LWE assumption.

Sender Security. The proof of sender security follows almost directly from [PVW08]. We now sketch the details.

For a fixed $(\mathbf{A}, \mathbf{v}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$, following [PVW08], we define $\delta(\mathbf{A}, \mathbf{v})$ to be the statistical distance between $(\mathbf{r}^T\mathbf{A}^T, \mathbf{r}^T\mathbf{v})$ where $\mathbf{r} \leftarrow \mathcal{R}^m$ and the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. The following lemma was shown in [PVW08].

Lemma 3.13 ([PVW08, GPV08]) *Let $m \geq 2(n+1)\log q$ and let $\varepsilon = \frac{1}{q^{(n+1)/2}}$. Then, with all but negligible probability over the random choices of $(\mathbf{A}, \mathbf{u}_0, \mathbf{u}_1)$, for every $\mathsf{msg}_R \in \mathbb{Z}_q^m$, there exists $b \in \{0,1\}$ such that $\delta(\mathbf{A}, \mathsf{msg}_R + \mathbf{u}_b) \leq \varepsilon$. Furthermore, given the trapdoor matrix $\mathbf{T}$, there exists an efficient algorithm that finds such a b given msg_R.*

The sender security follows from a straightforward application of this lemma.

Receiver Security. We describe the algorithm Ext_S. This algorithm outputs $\mathsf{msg}_\mathrm{R} = \mathbf{w}$ and $\rho_b = \mathbf{s}_b$ for each $b \in \{0,1\}$. It is easy to see that this extractor satisfies the receiver security requirement.

3.2 Realizing $\mathcal{F}_{\mathrm{OTCor}}$: Honest Majority Case

We first give a brief introduction to the client-server model.

- $\mathsf{K}_{\mathrm{R}}(1^\lambda)$:
  1. $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapSamp}(1^n, 1^m, q)$.
  2. Sample $\mathbf{u}_0, \mathbf{u}_1 \leftarrow \mathbb{Z}_q^m$.
  3. Output $\sigma := (\mathbf{A}, \mathbf{u}_0, \mathbf{u}_1), \tau := \mathbf{T}$.
- $\mathsf{K}_{\mathrm{S}}(1^\lambda)$:
  1. $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapSamp}(1^n, 1^m, q)$.
  2. Sample $\mathbf{w} \leftarrow \mathbb{Z}_q^m$ and $\mathbf{s}_0, \mathbf{s}_1 \leftarrow \mathbb{Z}_q^n$.
  3. Set $\mathbf{u}_b := \mathbf{A}^T\mathbf{s}_b + \mathbf{e}_b - \mathbf{w}$ where $\mathbf{e}_b \leftarrow \chi^m$ for $b \in \{0, 1\}$.
  4. Output $\sigma := (\mathbf{A}, \mathbf{u}_0, \mathbf{u}_1), \tau := (\mathbf{T}, \mathbf{s}_0, \mathbf{s}_1)$.
- $\mathsf{Sen}(\sigma)$:
  1. Sample $\mathbf{r}_0, \mathbf{r}_1 \in \mathcal{R}^m$.
  2. Set $\mathbf{v}_b := \mathbf{r}_b^T A^T$.
  3. Output $\mathsf{msg}_{\mathrm{S}} := (\mathbf{v}_0, \mathbf{v}_1)$ and $\omega = (\mathbf{r}_0, \mathbf{r}_1)$.
- $\mathsf{Rec}(\sigma, b)$:
  1. Sample $\mathbf{s}_R \leftarrow \mathbb{Z}_q^n$.
  2. Set $\mathsf{msg}_{\mathrm{R}} := \mathbf{A}^T\mathbf{s}_R + \mathbf{e}_R - \mathbf{u}_b$ where $\mathbf{e}_R \leftarrow \chi^m$.
  3. Output $\mathsf{msg}_{\mathrm{R}}$ and $\rho_b := (\mathbf{s}_R, b)$.
- $\mathsf{out}_{\mathrm{S}}(\sigma, \omega, \mathsf{msg}_{\mathrm{R}})$:
  1. Parse ω as $(\mathbf{r}_0, \mathbf{r}_1)$.
  2. Output k_b as the msb of $\mathbf{r}_b^T(\mathsf{msg}_{\mathrm{R}} + \mathbf{u}_b)$ for $b \in \{0, 1\}$.
- $\mathsf{out}_{\mathrm{R}}(\sigma, \rho_b, \mathsf{msg}_{\mathrm{S}})$:
  1. Parse ρ_b as $(\mathbf{s}_R, b)$; parse $\mathsf{msg}_{\mathrm{S}}$ as $(\mathbf{v}_0, \mathbf{v}_1)$.
  2. Output k'_b to be the msb of $\mathbf{v}_b^T\mathbf{s}_R$.

Figure 5: Non-Interactive Oblivious Transfer from LWE

Client-Server Model. In the client-server model for computing $\mathcal{F}_{\mathrm{OTCor}}$, there are two clients: the sender and the receiver of the $\mathcal{F}_{\mathrm{OTCor}}$ functionality. There are $2t + 1$ servers and at most t of them are corrupted. In the two-round protocol, a single message is sent from both the sender and the receiver to the $2t + 1$ servers (via a private channel) and a single message is sent from all the $2t + 1$ servers to the receiver (again via a private channel) which enables the receiver to obtain the output of the function.

Ishai et al. [IKP10] gave a two-round protocol for computing degree-2 functionalities in the client-server model. We observe that this protocol in fact implies as protocol for securely computing

$\mathcal{F}_{\mathrm{OTCor}}$ in the client-server model where the majority of the servers are honest and the adversary is semi-honest. We state the theorem here and for completeness we give a proof in Appendix B.

Theorem 3.14 *There is a protocol that securely realizes $\mathcal{F}_{\mathrm{OTCor}}$ in the client-server model against adversaries who can corrupt an arbitrary number of clients and a minority of the servers.*

4 Two-round Semi-Honest MPC in the $\mathcal{F}_{\mathrm{OTCor}}$ Model

In this section, we give our construction of two-round MPC against semi-honest adversaries in the $\mathcal{F}_{\mathrm{OTCor}}$ model when the adversary is allowed to corrupt an arbitrary subset of the parties. The results we obtain against semi-honest adversaries are as follows (all our results are in the $\mathcal{F}_{\mathrm{OTCor}}$ model):

1. We first give a perfectly secure, two-round protocol for constant-size functionalities.
2. Next, using s result in [BGI+18] and the protocol from Step 1, we will give a protocol with perfectly (resp. statistical) secure, two-round protocol for functionalities with perfect (resp. statistical) randomized encodings with constant degree. Following [AIK04], we will denote the class of functions with perfectly (resp. statistically) secure constant degree randomized encodings as PREN (resp. SREN). Applebaum et al. [AIK04] showed that some of the natural complexity classes such as NC^1 and mod-2 branching programs $\oplus L/\mathsf{poly}$ are contained in PREN. A complexity class that is in SREN but not known to be in PREN is NL.
3. Next, using the result in [BMR90] and the protocol from Step 1, we will give a protocol for all circuits making black-box use of a pseudorandom generator.

4.1 Protocols for Constant-Size Functionalities

For a constant n, let $f : \{0,1\}^n \rightarrow \{0,1\}$ be a function with constant circuit size.[9] For each $i \in [n]$, the party P_i has input bit x_i and the parties want to securely compute $f(x_1, \ldots, x_n)$.[10] We give perfectly secure, two-round protocols for computing f both in the dishonest majority setting in the $\mathcal{F}_{\mathrm{OTCor}}$ hybrid model.

To construct a two-round protocol in the dishonest majority setting, we will use the same high level idea of Garg and Srinivasan [GS18]. To be more precise, we will take an arbitrary round protocol that securely computes the function f and compress it to two-rounds. However, to construct a perfectly secure protocol we will make the following changes to the round-collapsing compiler of [GS18],

1. All the executions of two-round oblivious transfer used by the round-collapsing compiler in [GS18] are replaced with perfectly secure, two-round oblivious transfer from OT correlations.

[9]For simplicity, we restrict ourselves to functions that output a single bit. We note that all our results can be generalized to functions with multiple bits with efficiency growing linearly with this number. We also assume that all the parties get the output of this functionality. We can also generalize our result for the case where some specific parties get the output.

[10]Again, for simplicity we restrict ourselves to parties with a single input bit and our results naturally generalize to parties with multiple bits as input.

2. The garbled circuits used in [GS18] compiler are replaced with perfectly secure, decomposable randomized encodings for NC^0 circuits (cf. Definition 2.2).

3. The underlying multi-round protocol that we want to round-compress might use cryptographic operations (which is necessary in the dishonest majority setting) and this creates the following two problems: (i) we can no longer argue perfect/statistical security, (ii) a subtle but a more important problem is that the compiler in [GS18] makes use of the code of the underlying protocol and hence if the underlying protocol involves cryptographic operations then the resultant two-round protocol makes non-black box use of cryptographic primitives. To solve the first problem, we will only round-compress perfect/statistical protocols in the OT-hybrid model (e.g., [GMW87, Kil88, IPS08]). Notice that any protocol in the OT-hybrid model can be reduced information theoretically to a protocol in the $\mathcal{F}_{\mathrm{OTCor}}$ functionality. To make the operations performed by all the parties information theoretic, we will generate OT correlations and make these correlations as part of the party's input. For example, consider two parties P_1 and P_2 who wish to do an OT in some round of the underlying protocol. Now, P_1 and P_2 will use the OT correlations from their input to perform an information theoretic OT.

The rest of the subsection is organized as follows. We will first recall the notion of conforming protocols from [GS18]. Intuitively, conforming protocols are MPC protocols with some additional structure. [GS18] showed that any MPC protocol can be transformed to a conforming protocol (with some efficiency loss). We give a generalization of the notion of conforming protocols to work in $\mathcal{F}_{\mathrm{OTCor}}$ model. Then, we will describe our construction of two-round MPC in the $\mathcal{F}_{\mathrm{OTCor}}$ hybrid model.

Conforming Protocol. We will now recall the notion of conforming protocols from [GS18]. We introduce an additional parameter s such that in each round of the conforming protocol, a single party computes s NAND gates and broadcasts the output of these NAND gates to every party. We note that in the formulation of [GS18], the parameter s was set to 1. We introduce this parameter for better concrete efficiency.

Consider a n-party deterministic[11] MPC protocol Φ between parties $P_1, \ldots, P_n$ with inputs $x_1, \ldots, x_n$, respectively. For each $i \in [n]$, we let $x_i \in \{0,1\}^m$ denote the input of party P_i (x_i's also include the randomness used in the protocol and hence they are m bits long). A conforming protocol Φ in the $\mathcal{F}_{\mathrm{OTCor}}$ is defined by functions pre, post, and a OT correlations generation phase and computations steps or what we call *actions* $\phi_1, \cdots \phi_T$. The protocol Φ proceeds in four stages: the OT correlations generation phase, the pre-processing stage, the computation stage and the output stage.

- **OT correlations generator:** For every instance of the OT to be performed in the protocol, interact with the $\mathcal{F}_{\mathrm{OTCor}}$ functionality to generate OT correlations.

- **Pre-processing phase**: For each $i \in [n]$, party P_i computes

$$(z_i, v_i) \leftarrow \mathsf{pre}(i, x_i)$$

  where pre is a randomized algorithm and the input x_i is now augmented with the OT correlations generated in the previous step. The algorithm pre takes as input the index i of the

[11]Randomized protocols can be handled by including the randomness used by a party as part of its input.

party, its input x_i and outputs $z_i \in \{0,1\}^{\ell/n}$ and $v_i \in \{0,1\}^{\ell}$ (where ℓ is a parameter of the protocol). Finally, P_i retains v_i as the secret information and broadcasts z_i to every other party. We require that $v_{i,k} = 0$ for all $k \in [\ell] \setminus \{(i-1)\ell/n + 1, \ldots, i\ell/n\}$.

- **Computation phase**: For each $i \in [n]$, party P_i sets

$$\mathsf{st}_i := (z_1 \| \cdots \| z_n).$$

  Next, for each $t \in \{1 \cdots T\}$ parties proceed as follows:

  1. Parse action ϕ_t as $(i, (a_1, b_1, c_1), \ldots, (a_s, b_s, c_s))$ where $i \in [n]$ and $a_j, b_j, c_j \in [\ell]$ for all $j \in [s]$.
  2. Party P_i computes s NAND gates as

$$\mathsf{st}_{i,c_j} = \mathsf{NAND}(\mathsf{st}_{i,a_j} \oplus v_{i,a_j}, \mathsf{st}_{i,b_j} \oplus v_{i,b_j}) \oplus v_{i,c_j}$$

     for all $j \in [s]$ and broadcasts $\{\mathsf{st}_{i,c_j}\}_{j \in [s]}$ to every other party.
  3. Every party P_k for $k \neq i$ updates st_{k,c_j} for all $j \in [s]$ to the bits received from P_i.

  We require that for all $t, t' \in [T]$ such that $t \neq t'$, if $\phi_t = (\cdot, (\cdot, \cdot, c_1), \ldots, (\cdot, \cdot, c_s))$ and $\phi_{t'} = (\cdot, (\cdot, \cdot, c'_1), \ldots, (\cdot, \cdot, c'_s))$ then $\{c_j\} \cap \{c'_j\} = \varnothing$. We use $A_i \subset [T]$ to denote the set of rounds in which the party P_i sends a message. Namely, $A_i = \{t \in T \mid \phi_t = (i, (\cdot, \cdot, \cdot), \ldots, (\cdot, \cdot, \cdot))\}$.

- **Output phase**: For each $i \in [n]$, party P_i outputs $\mathsf{post}(i, \mathsf{st}_i, v_i)$.

We now show the following lemma which is a generalization of the lemma proved in [GS18].

Lemma 4.1 *For $s = 1$, any MPC protocol Π in the OT hybrid model can be transformed into a conforming protocol Φ in the $\mathcal{F}_{\mathrm{OTCor}}$ model while inheriting the correctness and the security of the original protocol. Furthermore, there exists a choice of s such that the number of rounds of the resulting conforming protocol is $O(n \cdot d_{\max} \cdot r)$ where $d_{\max}$ is the maximum depth of the boolean circuit computing the next message function of any party and r is the number of rounds of the original protocol Π.*

We prove the lemma in Appendix C.

Remark 4.2 *We note that if the i-th party's output is public then the algorithm* post *need not take v_i as input.*

Compiled Protocol. We describe the compiled protocol in Figure 6 and give an informal overview below.

Overview. Our construction involves a pre-preprocessing phase followed by the two-rounds of interaction (described in Figure 6) and a local evaluation phase (described below). In the preprocessing phase, the parties interact with the $\mathcal{F}_{\mathrm{OTCor}}$ functionality to generate two sets of OT correlations. The first set of OT correlations are generated to execute the two-round oblivious transfer used in the compiler of Garg and Srinivasan [GS18]. The second set of OT correlations are to be hardwired as part of the input in the conforming protocols so that the operations done by each party in the conforming protocol are information theoretic. To obtain perfect security, we also use a decomposable randomized encoding in place of garbled circuits. Apart from these changes, our two-round protocol is exactly same as in [GS18].

Evaluation. To compute the output of the protocol, each party P_i does the following:

1. For each $k \in [n]$, let $\widehat{x}^{k,1}$ be the input encoding received from P_k at the end of round 2.
2. **for** each t from 1 to T do:
   (a) Parse ϕ_t as $(i^*, (a_1, b_1, c_1), \dots, (a_s, b_s, c_s))$.
   (b) Compute $(\{(\xi_j, \omega_j)\}_{j\in[s]}, \widehat{x}^{i^*,t+1}) := \mathsf{Dec}(\widetilde{f}^{i,t}, \widehat{x}^{i,t})$.
   (c) Set $\mathsf{st}_{i,c_j} := \xi_j$.
   (d) **for** each $k \neq i^*$ do:
      i. Compute $(\{\mathsf{ots}^2_j\}_{j\in[s]}, \{\widehat{x}^{k,t+1}_h\}_{h\in[\ell]\setminus\{c_j\}_{j\in[s]}}) := \mathsf{Dec}(\widetilde{f}^{i,t}, \widehat{x}^{i,t})$.
      ii. For every $j \in [s]$:
         A. Parse ots^2_j as (Y_0, Y_1) and ω_j as $\{\gamma^k_j\}_{k\in[n]\setminus\{i^*\}}$.
         B. Recover $\widehat{x}^{k,t+1}_{c_j} := Y_{\xi_j} \oplus \gamma^k_j$.
      iii. Set $\widehat{x}^{k,t+1} := \{\widehat{x}^{k,t+1}_h\}_{h\in[\ell]}$.
3. Compute the output as $\mathsf{post}(i, \mathsf{st}_i, v_i)$.

Asymptotic cost. Since the function f is constant size, the number of rounds of the underlying protocol and the maximum depth of the next message functions are constant (e.g., if we use [GMW87] as the underlying protocol). As a result of Lemma 4.1, the number of rounds of the conforming protocol is also a constant since k is a constant. Hence, the asymptotic cost of our protocol is a constant (though concretely it grows as $2^{O(T)}$ where T is the number of rounds of the conforming protocol).

Security. The only changes that we make when compared to the protocol in [GS18] is that we use information theoretic, two-round oblivious transfer (based on OT correlations) and perfectly secure DRE in place of garbled circuits. For completeness, we show the proof of the following theorem in Appendix D.

Theorem 4.3 *For every constant size function f, the protocol in Figure 6 perfectly computes f against a semi-honest adversaries who might corrupt an arbitrary subset of the parties.*

Extensions. We will now describe two-extensions to the protocol in Figure 6.

- **f need not be known until the second round.** We will now describe how to augment the protocol so that the function f to be computed need not be known until the beginning of the second round and only the size of these functions need to be known before the first round. Let us assume for simplicity that, $|f| = m'$. We define a $(k + m'k)$-party functionality C that takes x_i from party P_i for every $i \in [k]$ and takes a bit $y_{i\ell}$ from party $P_{i\ell}$ for each $i \in [k]$ and $\ell \in [m']$ and does the following: it checks if for each $i, i' \in [n]$ and $\ell \in [m']$, $y_{i,\ell} \overset{?}{=} y_{i',\ell}$; if yes, it interprets $y_{1,1}, \dots, y_{1,m'}$ as the function f and computes an universal circuit $U(x_1, \dots, x_k, f)$ that outputs $f(x_1, \dots, x_k)$. With this functionality, let us now see how to change the two-round protocol so that the parties need not know f until the beginning of the second-round.

Let Φ be an n-party conforming semi-honest MPC protocol (with T rounds in the computation phase) and $\widehat{f}$ be a DRE (See Definition 2.2).

Pre-processing Phase: On input the number of parties n, the number of functions s, the size of each of these functions and the size of each party's input m, the party P_i does the following:

1. For each $j \in [s]$ and $\alpha, \beta \in \{0,1\}$:
   (a) For each $t \in A_i$ (recall the definition of A_i from the description of conforming protocol), send $((t,j,\alpha,\beta), \mathbf{receiver}, i, r_{t,j,\alpha,\beta})$ (where $r_{t,j,\alpha,\beta}$ is chosen randomly) and for each $t \in [T] \setminus A_i$, send $((t,j,\alpha,\beta), \mathbf{sender}, i)$ to $\mathcal{F}_{\mathrm{OTCor}}$ functionality.
   (b) Receive $\omega_{t,j,\alpha,\beta} = \{\gamma^k_{t,j,\alpha,\beta}\}_{k\in[n]\setminus\{i\}}$ for each $t \in A_i$ and $(\gamma^0_{t,j,\alpha,\beta}, \gamma^1_{t,j,\alpha,\beta})$ if $t \in [T] \setminus A_i$ from $\mathcal{F}_{\mathrm{OTCor}}$.
2. Execute the OT correlations generation phase of the conforming protocol Φ.

Round-1: Each party P_i does the following:

1. Compute $(z_i, v_i) \leftarrow \mathsf{pre}(i, x_i)$.
2. For each $t \in A_i$, for each $j \in [s]$ and $\alpha, \beta \in \{0,1\}$, compute

$$\mathsf{ots}^1_{t,j,\alpha,\beta} \leftarrow \big(v_{i,c_j} \oplus \mathsf{NAND}(v_{i,a_j} \oplus \alpha, v_{i,b_j} \oplus \beta)\big) \oplus r_{t,j,\alpha,\beta},$$

   where $\phi_t = (i, (a_1,b_1,c_1), \ldots, (a_s,b_s,c_s))$.
3. Send $\Big(z_i, \{\mathsf{ots}^1_{t,j,\alpha,\beta}\}_{t\in A_i, j\in[s], \alpha,\beta\in\{0,1\}}\Big)$ to every other party.

Round-2: In the second round, each party P_i does the following:

1. Set $\mathsf{st}_i := (z_1\|\ldots\|z_i\|\ldots\|z_n)$.
2. Set $\mathbf{a}^{i,T+1}_{k,0} = \mathbf{a}^{i,T+1}_{k,1} = \perp$ for all $k \in [\ell]$.
3. **for** each t from T down to 1,
   (a) Parse ϕ_t as $(i^*, (a_1,b_1,c_1), \ldots, (a_s,b_s,c_s))$.
   (b) If $i = i^*$ then
      i. Let $f^{i,t}$ be a NC^0 function that takes st as input, updates st_{c_j} as per the action for every $j \in [s]$ and outputs $\omega_{t,j,\mathsf{st}_{a_j},\mathsf{st}_{b_j}}$ for every $j \in [s]$ along with $\mathbf{a}^{i,t+1}_{k,\mathsf{st}_k}$ for every $k \in [\ell]$.
   (c) If $i \neq i^*$ then for every $\alpha, \beta \in \{0,1\}$,
      i. Compute $\mathsf{ots}^2_{t,j,\alpha,\beta} := (\mathbf{a}^{i,t+1}_{c_j,0} \oplus X_0, \mathbf{a}^{i,t+1}_{c_j,1} \oplus X_1)$ where $X_b = \gamma^{b\oplus \mathsf{ots}^1_{t,j,\alpha,\beta}}_{t,j,\alpha,\beta}$ for every $j \in [s]$.
      ii. Let $f^{i,t}$ be a NC^0 function that takes st as input and outputs $\mathbf{a}^{i,t+1}_{k,\mathsf{st}_k}$ for all $k \in [\ell] \setminus \{c_j\}$ and $\mathsf{ots}^2_{t,j,\mathsf{st}_{a_j},\mathsf{st}_{b_j}}$ for every $j \in [s]$.
   (d) Compute $(\widetilde{f}^{i,t}, \{(\mathbf{a}^{i,t}_{k,0}, \mathbf{a}^{i,t}_{k,1})\}_{k\in[\ell]}) \leftarrow \widehat{f}^{i,t}(;r)$.
4. Send $\big(\{\widetilde{f}^{i,t}\}_{t\in[T]}, \{\mathbf{a}^{i,1}_{k,\mathsf{st}_k}\}_{k\in[\ell]}\big)$ to every other party.

Figure 6: Two-round MPC for constant size functions in the $\mathcal{F}_{\mathrm{OTCor}}$ hybrid model

We will use an underlying conforming protocol that securely computes the constant size circuit C. In the compiled protocol, we will let each party P_i to additionally emulate the parties $\{P_{i\ell}\}_{\ell \in [m']}$. To be more precise, in the first round of the protocol, for each $\ell \in [m']$, the party P_i sends two first round messages on behalf of party $P_{i\ell}$; the first message assuming the bit $y_{i\ell} = 0$ and the second message assuming the bit $y_{i\ell'} = 1$. In the beginning of the second round, all the parties know the description of the functions f and hence can choose the first round message corresponding to the correct value of $y_{i\ell}$ and ignore the other message. Based on the chosen messages, the parties generate the second round message in the compiled protocol.

- **Extension to the Client-Server setting.** We now describe an extension of our two-round protocol to the client-server setting. In the client server setting, there are n-input clients who holds the inputs, m servers who do not have any input and one output client. The input clients send a single message to each of the m servers and the servers send a single message to the output client and the output client learns the output of the function based on the server's message. We will assume that any number of clients can be corrupted but there is at least one server who is uncorrupted. We will transform our 2-round protocol in the $\mathcal{F}_{\mathrm{OTCor}}$ model to one in the client-server model. Later, in Section 5, we give a general transformation from any two-round MPC protocol with security against semi-honest adversaries who might corrupt an arbitrary subset of the parties to a protocol in the client-server model. However, this general transformation might make non-black-box use of cryptography but the transformation we give here is specific to protocol in Figure 6 and is information theoretic.

  1. The i-th input client computes the first round message $\left(z_i, \{\mathsf{ots}^1_{t,j,\alpha,\beta}\}_{t \in A_i, j \in [s], \alpha,\beta \in \{0,1\}}\right)$ of our two-round protocol and sends it to each of the servers.
  2. In addition to the protocols first round message, the client will generate a randomized encoding of NC^0 circuits $\overline{f}^{i,t}$ for every $t \in [T]$, and sends these randomized encodings along with an additive secret share of the input encoding $(\mathbf{a}^{i,1}_0, \mathbf{a}^{i,1}_1)$ to the servers. Let us now describe the functionality computed by $\overline{f}^{i,t}$. The functionality takes in the first round messages of all parties and reconstructs st_i. If $t \in A_i$, then it computes the same function as that of $f_{i,t}$ (described in Figure 6). If $t \notin A_i$, it will use $\mathsf{ots}^1_{t,j,\alpha,\beta}$ (obtained from the first round messages of the parties) and will generate $\mathsf{ots}^2_{t,j,\alpha,\beta}$ exactly as described in the protocol. Then, it computes the same functionality as that of $f^{i,t}$.
  3. The servers on receiving the first round messages from all the input clients, choose the secret share of the input encodings corresponding to the first round messages from all the clients and sends the chosen secret shares to the output client.
  4. The output client reconstructs the input encodings from the shares and decodes the randomized encodings exactly as given the evaluation procedure of our two-round protocol to obtain the output.

4.2 Protocols for PREN and SREN

In this subsection, we will use the protocols described in Section 4.1 to construct protocols for functions in PREN and SREN. We first define the $\mathsf{dMULTPlus}$ function below.

$$\mathsf{dMULTPlus}((x_1, z_1), \ldots, (x_d, z_d)) = x_1 \cdot \ldots \cdot x_d + \sum_{i=1}^{d} z_i.$$

We recall the following lemma from [BGI+18].

Lemma 4.4 ([BGI+18]) *Let $g : \{0,1\}^n \to \{0,1\}$ be a constant degree function i.e., there exists a constant d such that $g(x_1, \ldots, x_n) = \sum a^{\ell}_{i_1 \ldots i_d} x_{i_1} x_{i_2} \ldots x_{i_d}$. There exists a perfectly secure, two-round protocol in the presence of secure channels between every pair of parties for computing g against semi-honest adversary (corrupting an arbitrary subset of parties) in the $\mathcal{F}_{\mathsf{dMULTPlus}}$ hybrid model. The efficiency of the protocol is $O(m + n^2)$ where m is the number of monomials in g.*

We obtain the following corollary of our Theorem 4.3.

Corollary 4.5 *There exists a perfectly secure, two-round protocol for realizing $\mathcal{F}_{\mathsf{dMULTPlus}}$ functionality against semi-honest adversary (corrupting an arbitrary subset of parties) in the $\mathcal{F}_{\mathrm{OTCor}}$ hybrid model. The efficiency of the protocol is $2^{\mathsf{poly}(d)}$.*

Combining Lemma 4.4 and Corollary 4.5 and the observation that $\mathcal{F}_{\mathrm{OTCor}}$ implies secure channels, we get the following lemma.

Lemma 4.6 *Let $g : \{0,1\}^n \to \{0,1\}$ be a constant degree function i.e., there exists a constant d such that $g(x_1, \ldots, x_n) = \sum a^{\ell}_{i_1 \ldots i_d} x_{i_1} x_{i_2} \ldots x_{i_d}$. There exists a perfectly secure, two-round protocol for computing g against semi-honest adversary (corrupting an arbitrary subset of parties) in the $\mathcal{F}_{\mathrm{OTCor}}$ hybrid model. The efficiency of the protocol is $O(m + n^2)$ where m is the number of monomials in g.*

We now show our main theorem regarding securely computing functions in PREN and SREN.

Theorem 4.7 *Every $f : \{0,1\}^n \to \{0,1\}$ in PREN (resp. SREN) has an efficient, perfectly secure (resp., statistically secure) two-round protocol in the $\mathcal{F}_{\mathrm{OTCor}}$ model against a semi-honest adversary corrupting an arbitrary subset of parties. The computational cost incurred by each party is $O(m+n^2)$ where m is the size of the randomized encoding for f.*

Proof Let $\widehat{f} : \{0,1\}^n \times \{0,1\}^\rho \to \{0,1\}$ be the randomized encoding of the function f. Each party P_i chooses r_i uniformly at random from $\{0,1\}^\rho$ and the parties wish to securely compute the functionality $\widehat{f}(x_1, \ldots, x_n; r_1 \oplus r_2 \ldots \oplus r_n)$ (i.e., the input of party P_i is set as (x_i, r_i)).

Let $\widehat{f}(x_1, \ldots, x_n; r_1 \oplus r_2 \ldots \oplus r_n) = \sum a^{\ell}_{i_1 i_2 \ldots i_d} v_{i_1} v_{i_2} \ldots v_{i_d}$ where each v_{i_d} is either some input bit x_j or a bit of some random string r_j. We will use the protocol from Lemma 4.6 to securely compute $\widehat{f}$.

It now follows from the privacy of randomized encodings and the security of the protocol for computing $\widehat{f}$ the above protocol securely computes f against semi-honest corruptions. ■

Remark 4.8 *For simplicity, in Theorem 4.7, we considered a setting where each party holds a single bit as input and the output of the function f is also a single bit. This can be naturally generalized to a setting wherein each party holds a string as input and the number of outputs of the functions is greater than 1.*

We obtain the following corollary from Theorem 4.7.

Corollary 4.9 *There is a perfectly (resp. statistical) secure two-round protocol for branching programs (resp. non-deterministic branching programs) in the $\mathcal{F}_{\mathrm{OTCor}}$ model against a semi-honest adversary corrupting an arbitrary subset of parties.*

4.3 Protocols for Circuits

In this subsection, we will use the protocols described in Section 4.1 and make black-box use of a PRG to obtain secure protocols for computing circuits. Without loss of generality, we will restrict ourselves to circuits with fan-in 2 NAND gates. The high level idea is to use the protocol in Section 4.1 to compute the BMR garbling of a gate [BMR90]. To obtain the labels for executing the BMR garbled circuit, we run the BMR online phase in parallel.

BMR Garbling. We will now recall the semantics of a BMR garbled gate. The BMR garbling for a NAND gate g that takes wires a and b as input and the output wire is c is a set of values $\{\widetilde{G}^i_{r_1,r_2}\}_{r_1,r_2\in\{0,1\},i\in[n]}$, where

$$\widetilde{G}^j_{r_1,r_2} = \left(\bigoplus_{i=1}^n F_{k^i_{a,r_1}}(g,j,r_1,r_2) \oplus F_{k^i_{b,r_2}}(g,j,r_1,r_2)\right) \oplus k^j_{c,0} \oplus (\chi_{r_1,r_2} \wedge (k^j_{c,1} \oplus k^j_{c,0}))$$

where $\chi_{r_1,r_2} = ((\bigoplus_{i=1}^n \lambda_{i,a} \oplus r_1) \cdot (\bigoplus_{i=1}^n \lambda_{i,b} \oplus r_2) \oplus 1) \oplus (\bigoplus_{i=1}^n \lambda_{i,c})$. Here, F is a PRF, $k^i_{x,r}$ where $x \in \{a,b,c,\}$ and $r \in \{0,1\}$ is a PRF key, $\lambda_{i,x}$ for $x \in \{a,b,c,\}$ are bits.[12] The PRF keys $k^i_{x,r}$ and the bits $\lambda_{i,x}$ are chosen by each party before the first round of the protocol.

We notice that each output bit of $\{\widetilde{G}^i_{r_1,r_2}\}_{r_1,r_2\in\{0,1\},i\in[n]}$ is a constant degree (precisely, a degree 3 functionality). We will use the protocol in Lemma 4.6 to securely compute each output bit of $\{\widetilde{G}^i_{r_1,r_2}\}_{r_1,r_2\in\{0,1\},i\in[n]}$.[13]

Online Phase of BMR. We now describe the two-round BMR online phase.

1. For every wire w, which is the input wire of a party P_i, the other parties P_j will set $\lambda_{j,w} = 0$. The party P_i will compute $\alpha_w = \lambda_{i,w} \oplus x_w$ and broadcast it to all other parties.
2. For every α_w obtained, the party P_i will broadcast k^i_{w,α_w} to every other party.

Asymptotic Cost. The cost of computing every bit of $\widetilde{G}^i_{r_1,r_2}$ is $O(n^2)$ since the number of monomials in $\widetilde{G}^i_{r_1,r_2}$ is $O(n^2)$. So the overall complexity of our protocol is $O(n^3|C|\lambda)$. This gives a factor of n improvement over the cost in [GS18].

Using the above protocol for computing the BMR garbled gate in parallel with the online phase, we obtain the following theorem:

Theorem 4.10 *There is a computationally secure two-round protocol for any circuit C in the $\mathcal{F}_{\mathrm{OTCor}}$ model against a semi-honest adversary corrupting an arbitrary subset of parties, where the protocol makes a black-box use of a PRG. The computational cost incurred by each party is dominated by $O(n^3|C|)$ invocations of a length-doubling PRG.*

[12] For simplicity we consider a PRF. But all our results also work with a length doubling pseudorandom generator.

[13] Here, the parties will compute the PRF outputs locally and give these as inputs to the protocol

We the following two corollaries by realizing $\mathcal{F}_{\text{OTCor}}$ under DDH/QR or LWE in the strong-PKI model.

Corollary 4.11 (DDH/QR) *There is a computationally secure, two-round protocol for any circuit C in the strong-PKI model against a semi-honest adversary corrupting an arbitrary subset of parties, where the protocol makes a black-box use of a PRG and black-box use of a DDH/QR hard group.*

Corollary 4.12 (LWE) *Under the LWE assumption, there is a computationally secure, two-round protocol for any circuit C in the strong-PKI model against a semi-honest adversary corrupting an arbitrary subset of parties, where the protocol makes a black-box use of a PRG.*

4.4 Concretely Efficient Protocols in the Dishonest Majority Setting

In this subsection, we will describe a concretely efficient two-round, MPC protocols (against static, semi-honest adversaries corrupting a majority of the parties) for computing arbitrary circuits in the $\mathcal{F}_{\text{OTCor}}$ hybrid model.

It follows from Sections 4.2 and 4.3, that it is sufficient to focus on the constant size functionality $\mathsf{3MULTPlus}$ which is a three-party functionality defined as follows:

$$\mathsf{3MULTPlus}((x_1, z_1), (x_2, z_2), (x_3, z_3)) = x_1 \cdot x_2 \cdot x_3 + z_1 + z_2 + z_3$$

We first describe a multi-round protocol for computing $\mathsf{3MULTPlus}$ and then compress it to two-rounds using the protocol compiler given in Section 4.1 using a specialized garbling gadget that we will describe later.

Multi-round Protocol. The multi-round protocol we will be using is a variant of the one outlined in [ACJ17] for computing an additive secret sharing of $x_1 \cdot x_2 \cdot x_3$. We will assume the existence of a two-message OT protocol (which is implied by OT correlations).

1. In the first round of the protocol, the party P_1 sends two OT_1 messages to P_2 encoding the bits x_1 and 0 respectively. The party P_3 also sends two OT_1 messages to the first and the second party respectively and both these messages encode its input x_3.
2. In the second round, the party P_2 sends two OT_2 messages to first party. For the first OT_1 message that encodes x_1, it sends an OT_2 message that encodes $(0 \cdot x_2 + r_2, 1 \cdot x_2 + r_2)$ where r_2 is a randomly chosen. For the second OT_1 message, it samples a random y and sends an OT_2 message that encodes (y, y). The two strings encoded in the OT_2 message to the third party are $(0 \cdot r_2 + z_2 + y, 1 \cdot r_2 + z_2 + y)$.
3. In the third round, the P_1 recovers the message $u = x_1 \cdot x_2 + r_2$ from the OT_2 message of the second party and sends an OT_2 message to the third party with the two strings that are encoded are $(0 \cdot u + z_1 + y, 1 \cdot u + z_1 + y)$.
4. In the last round of the protocol, the P_3 recovers the message $v = x_3 r_2 + z_2 + y$ and $w = x_1 x_2 x_3 + x_3 r_2 + z_1 + y$ from the OT_2 messages sent from P_2 and P_1 respectively. It broadcasts $v + w + z_3$ to all the parties.

We will use the following garbling gadget from [HIJ+16] as our randomized encoding. Later, we will explain the concept of computation tables which are used to represent the next message function of a party.

Garbling Gadget. Let $f : \{0,1\}^k \to \{0,1\}^m$ be a function. The garbled function $\widehat{f}$ consists of 2^k rows with the contents in the $(\alpha_1, \alpha_2, \ldots, \alpha_k)$-th row being

$$f(\alpha_1 \oplus r_1, \alpha_2 \oplus r_2, \ldots, \alpha_k \oplus r_k) \oplus s^1_{\alpha_1} \oplus s^2_{\alpha_1 \| \alpha_2} \oplus \ldots s^k_{\alpha_1 \| \ldots \| \alpha_k}$$

where $r_1, \ldots, r_k$ are randomly chosen bits and $\{s^i_e\}_{i \in [k], e \in \{0,1\}^k}$ are chosen randomly from $\{0,1\}^m$. The garbled input $\widehat{x}$ consists of $(x_1 \oplus r_1, \ldots, x_k \oplus r_k)$ along with $\{s^i_{e \| x_i}\}_{i \in [k], e \in \{0,1\}^{i-1}}$. To decode, we just output the contents of the $(x_1 \oplus r_1, \ldots, x_k \oplus r_k)$-th row of the garbled function unmasked with the bits $s^1_{x_1 \oplus r_1} \oplus s^2_{(x_1 \oplus r_1) | (x_2 \oplus r_2)} \oplus \ldots s^k_{(x_1 \oplus r_1) \| \ldots \| (x_n \oplus r_n)}$.

Computation Tables. Instead of representing a party's next message function as boolean circuits, we will represent them as computation tables which are simply the truth table of the next message function of the party.

2-round Protocol. We now describe our two-round protocol for computing the $\mathsf{3MULTPlus}$ functionality.

- **Round-1:** In the first round of the protocol, the parties do the following:
  - P_1's round-1 messages are as follows:
    - It sends the two OT_1 messages encoding x_1 and 0 respectively to P_2.
    - It prepares the computation table corresponding to the message it has to send to P_3 in the third round of the original protocol.
      - P_1's message to P_3 depends on the four bits it receives from P_2 in the second round of the protocol (these four bits correspond to the two OT_2 messages) and the OT_1 message that it receives from P_3. Let i-th bit of the message from P_1 to P_3 for $i \in [2]$ be described by the function $f^{1 \to 3}_i()$ that takes as input the messages received from P_2 and P_3 in the second and the first round respectively.
    - For each $i \in [2]$ and $\alpha_1, \ldots, \alpha_5 \in \{0,1\}$, P_1 generates first round OT messages $\mathsf{ots}^{1 \to 3}_{i, \alpha_1, \ldots, \alpha_5}$ where the bit encoded is $f^{1 \to 3}_i(\alpha_1, \ldots, \alpha_5)$. P_1 broadcasts these messages.
  - P_2's round-1 messages are as follows:
    - P_2 first prepares its computation table for the messages it has to send in the second round of the original protocol.
      - P_2's messages to P_1 depends on the two OT_1 messages that it receives from P_1. P_2's message to P_1 in the second round consists of 4 bits corresponding to the two OT_2 messages. Let i-th bit of the message from P_2 to P_1 for $i \in [4]$ be described by the function $f^{2 \to 1}_i(\cdot, \cdot)$ that takes the messages received from P_1 as input.
      - Similarly, P_2's message to P_3 depends on the OT_1 message it receives from P_3 and P_2's message to P_3 consists of the two bits corresponding to the the OT_2 message. Let the i-th bit of the message from P_2 to P_3 for $i \in [2]$ be described by the function $f^{2 \to 3}_i(\cdot)$ that takes the message received from P_3 as input.
    - For each $i \in [4]$ and $\alpha, \beta \in \{0,1\}$, P_2 generates the first round OT messages $\mathsf{ots}^{2 \to 1}_{i, \alpha, \beta}$ where the bit encoded is $f^{2 \to 1}_i(\alpha, \beta)$. Similarly, for each $i \in [2]$ and $\alpha \in \{0,1\}$, P_2 generates first round OT messages $\mathsf{ots}^{2 \to 3}_{i, \alpha}$ where the bit encoded is $f^{2 \to 3}_i(\alpha)$. P_2 broadcasts these messages.

- P_3's messages in Round-1 are as follows:
  - It generates two OT_1 messages that both encode its input x_3 and sends them to P_2 and P_1 respectively.

- **Round-2:** In the following, we will denote (a_1, a_2) to be the two OT_1 messages that P_1 sends to P_2 in the first round of the original protocol, a_3 and b to be the OT_1 messages that P_3 sends to P_2 and P_1 in the first round of the original protocol. In round-2 of the protocol, the parties do the following:
  - P_1's message in Round-2 are as follows:
    - It uses the garbling gadget to garble a function f that takes 4 bits $(\alpha_1, \ldots, \alpha_4)$ as inputs and outputs the randomness used for generating $\{\mathsf{ots}^{1\to3}_{i,\alpha_1,\ldots,\alpha_4,b}\}_{i\in[2]}$. For each $i \in [4]$, let $(\widehat{x}_{i,0}, \widehat{x}_{i,1})$ be the two garbled inputs corresponding to the bits 0 and 1 for the above garbling. For each $i \in [4]$, P_1 generates OT_2 messages $\overline{\mathsf{ots}}^{2\to1}_{i,a_1,a_2}$ (with respect to the $\mathsf{ots}^{2\to1}_{i,a_1,a_2}$) where the messages encoded are $(\widehat{x}_{i,0}, \widehat{x}_{i,1})$. It broadcasts the garbled function and $\{\overline{\mathsf{ots}}^{2\to1}_{i,a_1,a_2}\}$ to all the parties.
  - P_2's messages in Round-2 are as follows:
    - For each $i \in [4]$, P_2 broadcasts the randomness used for generating $\mathsf{ots}^{2\to1}_{i,a_1,a_2}$.
    - For each $i \in [2]$, P_2 broadcasts the randomness used for generating $\mathsf{ots}^{2\to3}_{i,a}$.
  - P_3's messages in Round-2 are as follows:
    - It uses the garbling gadget to garble the last round message function of P_3 of the original protocol. That is, the function f_1 takes in the two bits received from P_2 in the second round, and the two bits received from P_1 in the third round and outputs the last round message of P_3. For each $i \in [4]$, let $(\widehat{x}_{i,0}, \widehat{x}_{i,1})$ be the two garbled inputs corresponding to the bits 0 and 1 for the above garbling. For each $i \in [2]$, P_3 generates OT_2 messages $\overline{\mathsf{ots}}^{2\to3}_{i,a}$ (with respect to the $\mathsf{ots}^{2\to1}_{i,a}$) where the messages encoded are $(\widehat{x}_{i,0}, \widehat{x}_{i,1})$.
    - For $i \in [3,4]$ and $\alpha_1, \ldots, \alpha_4 \in \{0,1\}$, P_3 generates OT_2 messages $\overline{\mathsf{ots}}^{1\to3}_{i-2,\alpha_1,\ldots,\alpha_4,b}$ (with respect to the $\mathsf{ots}^{1\to3}_{i-2,\alpha_1,\ldots,\alpha_4,b}$) where the messages encoded are $(\widehat{x}_{i,0}, \widehat{x}_{i,1})$. It again uses the garbling gadget to garble a function f_2 that takes 4 bits $(\alpha_1, \ldots, \alpha_4)$ as inputs and outputs the $\{\overline{\mathsf{ots}}^{1\to3}_{i+2,\alpha_1,\ldots,\alpha_4,b}\}_{i\in[2]}$. Let $(\widehat{y}_{i,0}, \widehat{y}_{i,1})$ be the two garbled inputs corresponding to the bits 0 and 1 for the garbled function $\widehat{f_2}$. For each $i \in [4]$, P_1 generates OT_2 messages $\overline{\mathsf{ots}}^{2\to1}_{i,a_1,a_2}$ (with respect to the $\mathsf{ots}^{2\to1}_{i,a_1,a_2}$) where the messages encoded are $(\widehat{y}_{i,0}, \widehat{y}_{i,1})$.
    - P_3 broadcasts $\widehat{f_1}, \widehat{f_2}, \{\overline{\mathsf{ots}}^{2\to3}_{i,a}\}_{i\in[2]}$ to all the parties.

- **Local Evaluation.** The parties use the randomness broadcasted by P_2 to recover the input encodings for evaluating $\widehat{f}$ and $\widehat{f_1}$. Later, they use the randomness output by evaluation of $\widehat{f}$ to recover the input encodings for evaluating $\widehat{f_2}$. The evaluation of $\widehat{f_2}$ gives the desired output.

Concrete Communication Cost. The concrete communication cost for the protocol is 1752 bits. The communication cost of generating an entry of the garbled circuit is $1752 \times n^3$ which is about 135 times the communication cost of the BMR protocol.

Security. The security of the protocol follows directly from Theorem 4.3.

5 Transformation to a Protocol in the Client-Server Model

In this section, we give a general transformation from a two-round MPC protocol with security against semi-honest adversaries to a protocol in the client-server model. We first recall the client-server model and then give the transformation.

Client-Server Model. In the client-server model, there are n input clients, m servers and a single output client. The input clients hold the inputs $x_1, \ldots, x_n$ and the servers and the output clients do not hold any input. The input clients send a single message to the servers (via private channels) and the servers send a single message to the output client who computes the output of the function. An adversary is allowed to corrupt any number of input and output clients but we assume that there exists at least one server who is not corrupted by the adversary. We will denote a protocol working with n clients and m servers as (n, m) MPC protocol.

We note that the existing two-round MPC results [GGHR14, GLS15, MW16, GS17, GS18, BL18] do not directly extend to this setting as the input clients generate both of the messages in these protocols. We now describe a general transformation from any two-round MPC protocol to a protocol in the client-server model.

Theorem 5.1 *Assuming the existence of a pseudorandom generator, for any $m \geq 2$, there exists a transformation from any n-party two-round, MPC protocol with security against semi-honest adversaries corrupting an arbitrary subset of parties to a (n, m) MPC protocol against semi-honest adversaries.*

Sketch of Proof (Informal) Let the two-round MPC protocol be abstractly defined as follows: in round-1, i-th party computes $(\mathsf{msg}_{i,1}, \mathsf{st}_i) \leftarrow \mathsf{MPC}_1(i, x_i)$ and sends $\mathsf{msg}_{i,1}$ it to all the parties. In the second round, the i-th party computes $\mathsf{msg}_{i,2} \leftarrow \mathsf{MPC}_2(i, x_i, \mathsf{st}_i, \{\mathsf{msg}_{j,1}\}_{j\in[n]})$ and sends $\mathsf{msg}_{i,2}$ to all the parties. The parties can compute the output from these messages. We start with the description of the (n, m) MPC protocol.

1. The i-th input client computes $\mathsf{msg}_{i,1}, \mathsf{st}_i \leftarrow \mathsf{MPC}_1(i, x_i)$ and sends $\mathsf{msg}_{i,1}$ to all the m-servers. Additionally, it computes a garbling of a circuit C_i that takes as input $\{\mathsf{msg}_{j,1}\}_{j\in[n]}$ and outputs $\mathsf{msg}_{i,2}$. It sends the garbled circuit $\widetilde{C}_i$ to all the servers along with an additive secret sharing of the input labels. That is, the k-th server receives the k-th secret share of both the input labels for each input wire of C_i.

2. The k-th server on receiving inputs from all the clients, chooses for each $i \in [n]$, the secret share of the label corresponding to $\{\mathsf{msg}_{j,1}\}_{j\in[n]}$ and sends them along with $\{\mathsf{msg}_{j,1}\}_{j\in[n]}, \{\widetilde{C}_j\}_{j\in[n]}$ to the output client.

3. The output client reconstructs the labels from the secret shares and evaluates $\widetilde{C}_i$ for each $i \in [n]$ to obtain $\mathsf{msg}_{i,2}$. It then computes the output of the function from $\{\mathsf{msg}_{i,1}, \mathsf{msg}_{i,2}\}_{i\in[n]}$.

Simulator. For each input client corrupted by the adversary, S will use the underlying simulator for MPC to generate their random tapes and for the uncorrupted clients, it uses the underlying simulator to generate the messages in both the rounds of the protocol. To each corrupted server, S forwards the first round message and a simulated garbled circuit that outputs the second round message on behalf of each uncorrupted client. Additionally, instead of additively secret sharing both labels, it just sends random shares to the corrupted servers. For each simulated garbled circuit generated, S chooses the shares that correctly reconstructs to the simulated input label on behalf of the uncorrupted servers. It then forwards these shares to the output client.

Proof of Indistinguishability. To show the indistinguishability of the simulated distribution to the real distribution, we first change all the garbled circuits sent from the uncorrupted input clients to be simulated and then we change the messages in both the rounds to be generated by the simulator of the underlying protocol. ■

6 Two-round MPC in the Honest Majority Setting

In the case of an honest majority, we will obtain a two-round protocol in the plain model (with every pair of parties being connected via a secure channel) for securely computing functionalities represented by polynomial sized branching programs against semi-honest adversaries. For the protocol to be efficient, we restrict the number of parties in the protocol to be a constant.

Conforming Protocols in the Honest Majority Setting. We give a generalization of the conforming protocol in [GS18] to work in a setting where each pair of parties are connected via a secure channel. The original compiler given in [GS18] fails in this case as it was designed only for protocols which make use of a broadcast channel.

Let $f : \{0,1\}^{nm} \to \{0,1\}$ be a function. Consider a n-party deterministic[14] MPC protocol Φ between parties $P_1, \ldots, P_n$ with inputs $x_1, \ldots, x_n$ that computes f.[15] For each $i \in [n]$, we let $x_i \in \{0,1\}^m$ denote the input of party P_i. A conforming protocol Φ is defined by functions pre, post, and computations steps or what we call *actions* $\phi_1, \cdots \phi_T$. The protocol Φ proceeds in three stages: the pre-processing stage, the computation stage and the output stage.

- **Pre-processing phase**: For each $i \in [n]$,
  - The party P_i computes
  $$(z_i, v_i) \leftarrow \mathsf{pre}(1^\lambda, i, x_i)$$
  where pre is a randomized algorithm. The algorithm pre takes as input the index i of the party, its input x_i and outputs $z_i \in \{0,1\}^{\ell/n}$ and $v_i \in \{0,1\}^\ell$ (where ℓ is a parameter of the protocol). Finally, P_i retains v_i as the secret information and broadcasts z_i to every other party. We require that $v_{i,k} = 0$ for all $k \in [\ell] \setminus \{(i-1)\ell/n + 1, \ldots, i\ell/n\}$.
  - For every $t \in [T]$, parse the action ϕ_t as $(i^*, k, (a_1, b_1, c_1), \ldots, (a_s, b_s, c_s))$. If $i = i^*$ and $k \neq \perp$, then send $v_{i,c_1}, v_{i,c_2}, \ldots, v_{i,c_s}$ to the party P_k via private channels. Party P_k updates $v_{k,c_j} = v_{i,c_j}$ for every $j \in [s]$.

[14] Randomized protocols can be handled by including the randomness used by a party as part of its input.

[15] For simplicity, we restrict the output of the function to be a single bit. We can naturally extend them to multiple bits.

- **Computation phase**: For each $i \in [n]$, party P_i sets

$$\mathsf{st}_i := (z_1 \| \cdots \| z_n).$$

Next, for each $t \in \{1 \cdots T\}$ parties proceed as follows:

1. Parse action ϕ_t as $(i, k, (a_1, b_1, c_1), \ldots, (a_s, b_s, c_s))$ where $i \in [n]$ and $a_j, b_j, c_j \in [\ell]$ for all $j \in [s]$.
2. Party P_i computes s NAND gates as

$$\mathsf{st}_{i,c_j} = \mathsf{NAND}(\mathsf{st}_{i,a_j} \oplus v_{i,a_j}, \mathsf{st}_{i,b_j} \oplus v_{i,b_j}) \oplus v_{i,c_j}$$

for all $j \in [s]$ and broadcasts $\{\mathsf{st}_{i,c_j}\}_{j \in [s]}$ to every other party.
3. Every party P_k for $k \neq i$ updates st_{k,c_j} for all $j \in [s]$ to the bits received from P_i.

We require that for all $t, t' \in [T]$ such that $t \neq t'$, if $\phi_t = (\cdot, (\cdot, \cdot, c_1), \ldots, (\cdot, \cdot, c_s))$ and $\phi_{t'} = (\cdot, (\cdot, \cdot, c'_1), \ldots, (\cdot, \cdot, c'_s))$ then $\{c_j\} \cap \{c'_j\} = \varnothing$. We use $A_i \subset [T]$ to denote the set of rounds in which the party P_i sends a message. Namely, $A_i = \{t \in T \mid \phi_t = (i, (\cdot, \cdot, \cdot), \ldots, (\cdot, \cdot, \cdot))\}$.

- **Output phase**: For each $i \in [n]$, party P_i outputs $\mathsf{post}(i, \mathsf{st}_i, v_i)$.

We prove the following lemma in Appendix E.

Lemma 6.1 *There exists a choice of s such that any protocol Π using secure channels can be transformed into a conforming protocol Φ (over secure channels) inheriting the correctness and the security properties of Π and the number of rounds of Φ is $O(n^2 \cdot d_{\max} \cdot r)$. Here, $d_{\max}$ is the maximum depth of the boolean circuit computing the next message function of any party and r is the number of rounds of the original protocol Π.*

Construction. In the construction, we will use a OT protocol that is secure when a majority of the parties are honest (refer Section 3.2). We will abstractly describe the protocol as follows: in the first round, the receiver computes a message $\mathsf{OT}_1(b)$ to obtain n-shares and sends the i-th share to party P_i via a secure channel. Similarly, the sender chooses two-random messages m_0 and m_1 and runs $\mathsf{OT}_2(m_0, m_1)$ to obtain n-shares and sends the i-th share to party P_i via a secure channel. Each party P_i computes a linear function on the shares to obtain ω_i. Given $\{\omega_i\}_{i \in [n]}$, the receiver runs the reconstruction algorithm OT_3 on them to obtain m_b. We give the formal description of the construction in Figure 7.

Evaluation. To compute the output of the protocol, each party P_i does the following:

1. For each $k \in [n]$, let $\widehat{x}^{k,1}$ be the input encoding received from P_k at the end of round 2.
2. **for** each t from 1 to T do:
   (a) Parse ϕ_t as $(i^*, k, (a_1, b_1, c_1), \ldots, (a_s, b_s, c_s))$.
   (b) Compute $(\{(\xi_j, \omega_j^{i^*})\}_{j \in [s]}, \widehat{x}^{i^*,t+1}) := \mathsf{Dec}(\widetilde{f}^{i^*,t}, \widehat{x}^{i^*,t})$.
   (c) Set $\mathsf{st}_{i,c_j} := \xi_j$ for each $j \in [s]$.

(d) **for** each $k \neq i^*$ do:

   i. Compute $(\{\mathsf{ots}^2_j\}_{j\in[s]}, \{\omega^k_j\}_{j\in[s]}, \{\widehat{x}^{k,t+1}_h\}_{h\in[\ell]\setminus\{c_j\}_{j\in[s]}}) := \mathsf{Dec}(\widetilde{f}^{i,t}, \widehat{x}^{i,t})$.

   ii. For each $j \in [s]$,

      A. Parse ots^2_j as (Y_0, Y_1) and compute $\{\gamma^k_j\}_{k\in[n]\setminus\{i^*\}}$ as $\mathsf{OT}_3(\{\omega^k_j\}_{k\in[n]})$.

      B. Recover $\widehat{x}^{k,t+1}_{c_j} := Y_{\xi_j} \oplus \gamma^k_j$.

   iii. Set $\widehat{x}^{k,t+1} := \{\widehat{x}^{k,t+1}_h\}_{h\in[\ell]}$.

3. Compute the output as $\mathsf{post}(i, \mathsf{st}_i, v_i)$.

Asymptotic Cost. We start with the constant round (to be more precise, 3-round) protocol with secure channels (e.g., [IK00]) for securely computing f represented by a polynomial sized branching program. We will compile this protocol via Lemma 6.1 to a conforming protocol. Notice that the protocol in [IK00] works over a finite field $\mathbb{F}$ where $|\mathbb{F}| \geq n$. Plugging [IK00] protocol into the compiler in Lemma 6.1, we get a conforming protocol with $\ell = O(n^2 m \log|\mathbb{F}|)$ where m is at most quadratic in the branching program size of f. In every round of the [IK00] protocol, every party computes a linear function on the private state and the messages received so far. Since the boolean circuit for multiplying two finite field numbers is in NC^1, the maximum depth of the circuit computed by a party in every round is $O(\log\log\mathbb{F})$. Hence, from Lemma 6.1, the total number of rounds of the conforming protocol is $O(n^2 \cdot \log\log\mathbb{F})$. Hence, the asymptotic cost of the protocol described in Figure 7 is $\ell^{O(n^2\cdot\log\log\mathbb{F})}$ which is $\mathsf{poly}(\ell)$ if n is a constant.

Security. The security of the protocol is argued via a generalization of the security proof in [GS18]. We state and prove the following theorem.

Theorem 6.2 *For a constant number of parties, the protocol given in Figure 7 is a perfectly secure, two-round protocol for computing for branching programs over secure point-to-point channels with security against a semi-honest adversary corrupting a strict minority of the parties.*

We prove this theorem in Appendix F.

7 Two-round Malicious MPC in the $\mathcal{F}_{\mathrm{OTCor}}$ Model

In this section, we give a construction of two-round malicious MPC for securely computing arbitrary circuits in the $\mathcal{F}_{\mathrm{OTCor}}$ model. Our construction makes black-box use of a pseudorandom generator. The construction is exactly same as the one described in the work of Garg and Srinivasan [GS18] except that we will be using $\mathcal{F}_{\mathrm{OTCor}}$ to generate the OT correlations. Recall that for proving the security of their construction, Garg and Srinivasan needed an additional property called as equivocal receiver security from their oblivious transfer. We show that the OTs generated via $\mathcal{F}_{\mathrm{OTCor}}$ are trivially equivocal receiver secure.

Construction. We will use take an arbitrary round, conforming protocol (cf. Section 4.1) secure against malicious adversaries and will compress it to two-rounds using the compiler of Garg and Srinivasan [GS18]. The formal description of our construction is given in Figure 8.

Let Φ be an n-party conforming semi-honest MPC protocol with an honest majority and $\widehat{f}$ be a DRE. Each pair of parties are connected via a secure channel.

Round-1: Each party P_i does the following:

1. Compute $(z_i, v_i) \leftarrow \mathsf{pre}(1^\lambda, i, x_i)$ where x_i is the augmented input that includes the randomness for the original protocol.
2. For every $t \in [T]$, parse the action ϕ_t as $(i^*, k, (a_1, b_1, c_1), \ldots, (a_s, b_s, c_s))$. If $i = i^*$ and $k \neq \perp$, then send $v_{i,c_1}, v_{i,c_2}, \ldots, v_{i,c_s}$ to the party P_k via private channels. Party P_k updates $v_{k,c_j} = v_{i,c_j}$ for every $j \in [s]$.
3. For each $j \in [s]$ and $\alpha, \beta \in \{0, 1\}$,
   (a) For each $t \in A_i$,
      i. Choose $r_{t,j,\alpha,\beta}$ randomly and compute $\mathsf{OT}_1(r_{t,j,\alpha,\beta})$ and send the corresponding shares to each party via the private channel.
      ii. Compute
      $$\mathsf{ots}^1_{t,j,\alpha,\beta} \leftarrow \big(v_{i,c_j} \oplus \mathsf{NAND}(v_{i,a_j} \oplus \alpha, v_{i,b_j} \oplus \beta)\big) \oplus r_{t,j,\alpha,\beta},$$
      where $\phi_t = (i, k, (a_1, b_1, c_1), \ldots, (a_s, b_s, c_s))$.
   (b) For each $t \notin A_i$,
      i. Choose $(\gamma^0_{t,j,\alpha,\beta}, \gamma^1_{t,j,\alpha,\beta})$ randomly and compute OT_2 on these messages and send the corresponding shares to each party via a private channel.
4. Send $\Big(z_i, \{\mathsf{ots}^1_{t,j,\alpha,\beta}\}_{t \in A_i, j \in [s], \alpha,\beta \in \{0,1\}}\Big)$ to every other party.

Round-2: In the second round, each party P_i does the following:

1. Set $\mathsf{st}_i := (z_1 \| \ldots \| z_i \| \ldots \| z_n)$.
2. Set $\mathbf{a}^{i,T+1}_{k,0} = \mathbf{a}^{i,T+1}_{k,1} = \perp$ for all $k \in [\ell]$.
3. For each $t \in [T]$, $j \in [s]$ and $\alpha, \beta \in \{0, 1\}$, let $\omega^{i,k}_{t,j,\alpha,\beta}$ be the share corresponding to linear computation on the OT between i^* and k for every $k \in [n] \setminus \{i^*\}$. We will let $\omega^i_{t,j,\alpha,\beta} = \{\omega^{i,k}_{t,j,\alpha,\beta}\}_{k \in [n] \setminus \{i^*\}}$.
4. **for** each t from T down to 1,
   (a) Parse ϕ_t as $(i^*, k, (a_1, b_1, c_1), \ldots, (a_s, b_s, c_s))$.
   (b) If $i = i^*$ then
      i. Let $f^{i,t}$ be a NC^0 function that takes st as input, updates st_{c_j} as per the action and outputs $\omega^i_{t,j,\mathsf{st}_{a_j},\mathsf{st}_{b_j}}$ for every $j \in [s]$ along with $\mathbf{a}^{i,t+1}_{k,\mathsf{st}_k}$ for every $k \in [\ell]$.
   (c) If $i \neq i^*$ then for every $\alpha, \beta \in \{0, 1\}$,
      i. Compute $\mathsf{ots}^2_{t,j,\alpha,\beta} := (\mathbf{a}^{i,t+1}_{c_j,0} + X_0, \mathbf{a}^{i,t+1}_{c_j,1} + X_1)$ where $X_b = \gamma^{b \oplus \mathsf{ots}^1_{t,j,\alpha,\beta}}_{t,j,\alpha,\beta}$.
      ii. Let $f^{i,t}$ be a NC^0 function that takes st as input and outputs $\mathbf{a}^{i,t+1}_{k,\mathsf{st}_k}$ for all $k \in [\ell] \setminus \{c_j\}_{j \in [s]}$, $\omega^i_{t,j,\mathsf{st}_{a_j},\mathsf{st}_{b_j}}$ and $\mathsf{ots}^2_{t,j,\mathsf{st}_{a_j},\mathsf{st}_{b_j}}$ for every $j \in [s]$.
   (d) Compute $(\widetilde{f}^{i,t}, \{(\mathbf{a}^{i,t}_{k,0}, \mathbf{a}^{i,t}_{k,1})\}_{k \in [\ell]}) \leftarrow \widehat{f}^{i,t}(; r)$.
5. Send $\big(\{\widetilde{f}^{i,t}\}_{t \in [T]}, \{\mathbf{a}^{i,1}_{k,\mathsf{st}_k}\}_{k \in [\ell]}\big)$ to every other party.

Figure 7: Two-round MPC for with an honest majority

Evaluation. The evaluation procedure is exactly same as the one described in Section 4.1.

Security. In order, to prove security, we show that the two-round, information theoretic OT, using OT correlations has equivocal receiver security. Once we prove this, we can directly use the result of Garg and Srinivasan [GS18] to argue security of our protocol. Below, we first recall the notion of equivocal receiver security.

Let (c, r_c) be the correlation with the receiver and (r_0, r_1) be the correlation with the sender. The two-round OT using these correlations can be abstractly defined as follows (where the receiver's input is a bit b and the sender's input is (m_0, m_1)): the receiver computes $(\mathsf{ots}_1, \omega) = \mathsf{OT}_1(b, c)$ and sends ots_1 to the sender and the sender sends $\mathsf{ots}_2 = \mathsf{OT}_2(\mathsf{ots}_1, (m_0, m_1), (r_0, r_1))$ to the receiver. The receiver computes $m_b = \mathsf{OT}_3(\mathsf{ots}_2, \omega)$. In our actual construction, $\mathsf{ots}_1 = b \oplus c$, $\omega = (b, r_c)$ and $\mathsf{ots}_2 = (m_0 \oplus r_{\mathsf{ots}_1}, m_1 \oplus r_{1 \oplus \mathsf{ots}_1})$.

Definition 7.1 (Equivocal Receiver Security) *We say that a two-round oblivious transfer has equivocal receiver security if there exists a PPT simulator Sim_{Eq} such that the for any $\beta \in \{0,1\}$:*

$$\left\{((\mathsf{ots}_1, \omega_\beta)) : (\mathsf{ots}_1, \omega_0, \omega_1) \leftarrow \mathsf{Sim}_{Eq}(1^\lambda)\right\} \overset{s}{\approx} \{(\mathsf{OT}_1(\beta, c))\}.$$

We now give the description of the Sim_{Eq}. While generating the OT correlations, Sim_{Eq} intercepts the message (r_0, r_1) that an adversarial sender sends to the $\mathcal{F}_{\mathrm{OTCor}}$ functionality. Sim_{Eq} samples ots_1 randomly and outputs ω_0 as $(0, r_{\mathsf{ots}_1})$ and ω_1 as $(1, r_{1 \oplus \mathsf{ots}_1})$. It is easy to see that the distribution of $(\mathsf{ots}_1, \omega_\beta)$ generated by Sim_{Eq} is identical to the real world distribution.

We now state our result which follows directly from [GS18].

Theorem 7.2 *The protocol given in Figure 8 is computationally secure two-round protocol for circuits in the $\mathcal{F}_{\mathrm{OTCor}}$ model against a malicious adversary corrupting an arbitrary subset of parties, where the protocol makes a black-box use of a PRG.*

References

[ABT18] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Perfect secure computation in two rounds. To appear in TCC, 2018. `https://eprint.iacr.org/2018/894`.

[ACGJ18] Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Round-optimal secure multiparty computation with honest majority. LNCS, pages 395–424, Santa Barbara, CA, USA, 2018. Springer, Heidelberg, Germany.

[ACJ17] Prabhanjan Ananth, Arka Rai Choudhuri, and Abhishek Jain. A new approach to round-optimal secure multiparty computation. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 468–499, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *45th FOCS*, pages 166–175, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press.

Let Φ be an n-party conforming semi-honest MPC protocol and $(\mathsf{Garble}, \mathsf{Eval})$ be a garbling scheme for circuits.

Pre-processing Phase: On input the number of parties n, the number of functions s, the size of each of these functions and the size of each party's input m, the party P_i does the following:

1. For each $j \in [s]$ and $\alpha, \beta \in \{0,1\}$:
   (a) For each $t \in A_i$, send $((t,j,\alpha,\beta), \textbf{receiver}, i, r_{t,j,\alpha,\beta})$ (where $r_{t,j,\alpha,\beta}$ is chosen randomly) and for each $t \in [T] \setminus A_i$, send $((t,j,\alpha,\beta), \textbf{sender}, i)$ to $\mathcal{F}_{\mathrm{OTCor}}$ functionality.
   (b) Receive $\omega_{t,j,\alpha,\beta} = \{\gamma^k_{t,j,\alpha,\beta}\}_{k\in[n]\setminus\{i\}}$ for each $t \in A_i$ and $(\gamma^0_{t,j,\alpha,\beta}, \gamma^1_{t,j,\alpha,\beta})$ if $t \in [T] \setminus A_i$ from $\mathcal{F}_{\mathrm{OTCor}}$.
2. Execute the OT correlations generation phase for the underlying conforming protocol Φ.

Round-1: Each party P_i does the following:

1. Compute $(z_i, v_i) \leftarrow \mathsf{pre}(1^\lambda, i, x_i)$.
2. For each $t \in A_i$, for each $j \in [s]$ and $\alpha, \beta \in \{0,1\}$, compute

$$\mathsf{ots}^1_{t,j,\alpha,\beta} \leftarrow \big(v_{i,c_j} \oplus \mathsf{NAND}(v_{i,a_j} \oplus \alpha, v_{i,b_j} \oplus \beta)\big) \oplus r_{t,j,\alpha,\beta},$$

   where $\phi_t = (i, (a_1,b_1,c_1), \ldots, (a_s,b_s,c_s))$.
3. Send $\Big(z_i, \{\mathsf{ots}^1_{t,j,\alpha,\beta}\}_{t\in A_i, j\in[s], \alpha,\beta\in\{0,1\}}\Big)$ to every other party.

Round-2: In the second round, each party P_i does the following:

1. Set $\mathsf{st}_i := (z_1 \| \ldots \| z_i \| \ldots \| z_n)$.
2. Set $\mathsf{lab}^{i,T+1} := \{\mathsf{lab}^{i,T+1}_{k,0}, \mathsf{lab}^{i,T+1}_{k,1}\}_{k\in[\ell]}$ where for each $k \in [\ell]$ and $b \in \{0,1\}$, $\mathsf{lab}^{i,T+1}_{k,b} := 0^\lambda$.
3. **for** each t from T down to 1,
   (a) Parse ϕ_t as $(i^*, (a_1,b_1,c_1), \ldots, (a_s,b_s,c_s))$.
   (b) If $i = i^*$ then
      i. Let $f^{i,t}$ be a NC^0 function that takes st as input, updates st_{c_j} as per the action and outputs $\omega_{t,j,\mathsf{st}_{a_j},\mathsf{st}_{b_j}}$ for every $j \in [s]$ along with $\mathsf{lab}^{i,t+1}_{k,\mathsf{st}_k}$ for every $k \in [\ell]$.
   (c) If $i \neq i^*$ then for every $\alpha, \beta \in \{0,1\}$,
      i. Compute $\mathsf{ots}^2_{t,j,\alpha,\beta} := (\mathsf{lab}^{i,t+1}_{c_j,0} \oplus X_0, \mathsf{lab}^{i,t+1}_{c_j,1} \oplus X_1)$ where $X_b = \gamma^{b\oplus \mathsf{ots}^1_{t,j,\alpha,\beta}}_{t,j,\alpha,\beta}$.
      ii. Let $f^{i,t}$ be a NC^0 function that takes st as input and outputs $\mathsf{lab}^{i,t+1}_{k,\mathsf{st}_k}$ for all $k \in [\ell] \setminus \{c_j\}$ and $\mathsf{ots}^2_{t,j,\mathsf{st}_{a_j},\mathsf{st}_{b_j}}$ for every $j \in [s]$.
   (d) Compute $(\widetilde{f}^{i,t}, \{\mathsf{lab}^{i,t}_k\}_{k\in[\ell]}) \leftarrow \widehat{f}^{i,t}(;r)$.
4. Send $\big(\{\widehat{f}^{i,t}\}_{t\in[T]}, \{\mathsf{lab}^{i,1}_{k,\mathsf{st}_{i,k}}\}_{k\in[\ell]}$ to every other party.

Figure 8: Two-round MPC against Malicious Adversaries in $\mathcal{F}_{\mathrm{OTCor}}$ hybrid model

[Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP 99*, volume 1644 of *LNCS*, pages 1–9, Prague, Czech Republic, July 11–15, 1999. Springer, Heidelberg, Germany.

[AJW11] Gilad Asharov, Abhishek Jain, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. *IACR Cryptology ePrint Archive*, 2011:613, 2011.

[AL11] Gilad Asharov and Yehuda Lindell. Utility dependence in correct and fair rational secret sharing. *Journal of Cryptology*, 24(1):157–202, January 2011.

[ALSZ13] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13*, pages 535–548, Berlin, Germany, November 4–8, 2013. ACM Press.

[ALSZ15] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions with security for malicious adversaries. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 673–701, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[BB89] Judit Bar-Ilan and Donald Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing, Edmonton, Alberta, Canada, August 14-16, 1989*, pages 201–209, 1989.

[BCG$^+$17] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. Homomorphic secret sharing: Optimizations and applications. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 17*, pages 2105–2122, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

[BCL$^+$05] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 361–377, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany.

[BCS96] Gilles Brassard, Claude Crépeau, and Miklos Santha. Oblivious transfers and intersecting codes. *IEEE Trans. Information Theory*, 42(6):1769–1780, 1996.

[BCW03] Gilles Brassard, Claude Crépeau, and Stefan Wolf. Oblivious transfers and privacy amplification. *Journal of Cryptology*, 16(4):219–237, September 2003.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 479–488, 1996.

[BGG$^+$18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. To appear in Crypto, 2018. `https://eprint.iacr.org/2017/956`.

[BGH07] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *48th FOCS*, pages 647–657, Providence, RI, USA, October 20–23, 2007. IEEE Computer Society Press.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 509–539, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[BGI17] Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 163–193, Paris, France, May 8–12, 2017. Springer, Heidelberg, Germany.

[BGI$^+$18] Elette Boyle, Niv Gilboa, Yuval Ishai, Huijia Lin, and Stefano Tessaro. Foundations of homomorphic secret sharing. In *ITCS 2018*, pages 21:1–21:21, January 2018.

[BL18] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pages 500–532, 2018.

[BM90] Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and applications. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 547–557, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513, Baltimore, MD, USA, May 14–16, 1990. ACM Press.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press.

[CEMY09] Seung Geol Choi, Ariel Elbaz, Tal Malkin, and Moti Yung. Secure multi-party computation minimizing online rounds. In *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, pages 268–286, 2009.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503, Montréal, Québec, Canada, May 19–21, 2002. ACM Press.

[CLP10] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *51st FOCS*, pages 541–550, Las Vegas, NV, USA, October 23–26, 2010. IEEE Computer Society Press.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 93–122, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[FGJI17] Nelly Fazio, Rosario Gennaro, Tahereh Jafarikhah, and William E. Skeith III. Homomorphic secret sharing from paillier encryption. In *Provable Security - 11th International Conference, ProvSec 2017, Xi'an, China, October 23-25, 2017, Proceedings*, pages 381–399, 2017.

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 74–94, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany.

[GLS15] S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 63–82, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th ACM STOC*, pages 365–377, San Francisco, CA, USA, May 5–7, 1982. ACM Press.

[GMMM18] Sanjam Garg, Mohammad Mahmoody, Daniel Masny, and Izaak Meckler. On the round complexity of OT extension. LNCS, pages 545–574, Santa Barbara, CA, USA, 2018. Springer, Heidelberg, Germany.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

[GS17] Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round MPC from bilinear maps. In *58th FOCS*, pages 588–599. IEEE Computer Society Press, 2017.

[GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pages 468–499, 2018.

[HIJ$^+$16] Shai Halevi, Yuval Ishai, Abhishek Jain, Eyal Kushilevitz, and Tal Rabin. Secure multiparty computation with general interaction patterns. In Madhu Sudan, editor, *ITCS 2016*, pages 157–168, Cambridge, MA, USA, January 14–16, 2016. ACM.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st FOCS*, pages 294–304, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press.

[IKM$^+$13] Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 600–620, 2013.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

[IKO$^+$11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 406–425. Springer, 2011.

[IKP10] Yuval Ishai, Eyal Kushilevitz, and Anat Paskin. Secure multiparty computation with minimal interaction. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 577–594, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.

[IMO18] Yuval Ishai, Manika Mittal, and Rafail Ostrovsky. On the message complexity of secure multiparty computation. In *Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part I*, pages 698–711, 2018.

[IPS08] 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, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

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

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[KOS15] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 724–741, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 333–342, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.

[PVW08] 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, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

[PW00] Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In S. Jajodia and P. Samarati, editors, *ACM CCS 00*, pages 245–254, Athens, Greece, November 1–4, 2000. ACM Press.

[Rab81] M. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.

A UC Security

In this section we briefly review UC security. For full details see [Can01]. A large part of this introduction has been taken verbatim from [CLP10].

The basic model of execution. Following [GMR88, Gol01], a protocol is represented as an interactive Turing machine (ITM), which represents the program to be run within each participant. Specifically, an ITM has three tapes that can be written to by other ITMs: the input and subroutine output tapes model the inputs from and the outputs to other programs running within the same "entity" (say, the same physical computer), and the incoming communication tapes and outgoing communication tapes model messages received from and to be sent to the network. It also has an identity tape that cannot be written to by the ITM itself. The identity tape contains the program of the ITM (in some standard encoding) plus additional identifying information specified below. Adversarial entities are also modeled as ITMs.

We distinguish between ITMs (which represent static objects, or programs) and *instances of ITMs*, or ITIs, that represent interacting processes in a running system. Specifically, an ITI is an ITM along with an identifer that distinguishes it from other ITIs in the same system. The identifier consists of two parts: A session-identifier (SID) which identifies which protocol instance the ITM belongs to, and a party identifier (PID) that distinguishes among the parties in a protocol instance. Typically the PID is also used to associate ITIs with "parties", or clusters, that represent some administrative domains or physical computers.

The model of computation consists of a number of ITIs that can write on each other's tapes in certain ways (specified in the model). The pair (SID,PID) is a unique identifier of the ITI in the system.

With one exception (discussed within) we assume that all ITMs are probabilistic polynomial time (PPT). An ITM is PPT if there exists a constant $c > 0$ such that, at any point during its run, the overall number of steps taken by M is at most n^c, where n is the overall number of bits written on the *input tape* of M in this run. (In fact, in order to guarantee that the overall protocol execution process is bounded by a polynomial, we define n as the total number of bits written to the input tape of M, *minus the overall number of bits written by M to input tapes of other ITMs.*; see [Can01].)

Security of protocols. Protocols that securely carry out a given task (or, protocol problem) are defined in three steps, as follows. First, the process of executing a protocol in an adversarial environment is formalized. Next, an "ideal process" for carrying out the task at hand is formalized. In the ideal process the parties do not communicate with each other. Instead they have access to an "ideal functionality," which is essentially an incorruptible "trusted party" that is programmed to capture the desired functionality of the task at hand. A protocol is said to securely realize an ideal functionality if the process of running the protocol amounts to "emulating" the ideal process for that ideal functionality. Below we overview the model of protocol execution (called the *real-life model*), the ideal process, and the notion of protocol emulation.

The model for protocol execution. The model of computation consists of the parties running an instance of a protocol Π, an adversary $\mathcal{A}$ that controls the communication among the parties, and an *environment* $\mathcal{Z}$ that controls the inputs to the parties and sees their outputs. We assume that all parties have a security parameter $n \in \mathbb{N}$. (We remark that this is done merely for convenience and is not essential for the model to make sense). The execution consists of a sequence of *activations*, where in each activation a single participant (either $\mathcal{Z}$, $\mathcal{A}$, or some other ITM) is activated, and may write on a tape of at most *one* other participant, subject to the rules below. Once the activation of a participant is complete (i.e., once it enters a special waiting state), the participant whose tape was written on is activated next. (If no such party exists then the environment is activated next.)

The environment is given an external input z and is the first to be activated. In its first activation, the environment invokes the adversary $\mathcal{A}$, providing it with some arbitrary input. In the context of UC security, the environment can from now on invoke (namely, provide input to) only ITMs that consist of a single instance of protocol Π. That is, all the ITMs invoked by the environment must have the same SID and the code of Π.

Once the adversary is activated, it may read its own tapes and the outgoing communication tapes of all parties. It may either deliver a message to some party by writing this message on the party's incoming communication tape or report information to $\mathcal{Z}$ by writing this information on the subroutine output tape of $\mathcal{Z}$. For simplicity of exposition, in the rest of this paper we assume authenticated communication; that is, the adversary may deliver only messages that were actually sent. (This is however not essential as shown in [BCL$^+$05].)

Once a protocol party (i.e., an ITI running Π) is activated, either due to an input given by the environment or due to a message delivered by the adversary, it follows its code and possibly writes a local output on the subroutine output tape of the environment, or an outgoing message on the adversary's incoming communication tape. Finally our adversary can decide to corrupt any honest party (in an adaptive fashion). In this case the input and the random coins used by this party are revealed to the adversary.

The protocol execution ends when the environment halts. The output of the protocol execution is the output of the environment. Without loss of generality we assume that this output consists of only a single bit.

Let $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(n, z, r)$ denote the output of the environment $\mathcal{Z}$ when interacting with parties running protocol Π on security parameter n, input z and random input $r = r_{\mathcal{Z}}, r_{\mathcal{A}}, r_1, r_2, \ldots$ as described above (z and $r_{\mathcal{Z}}$ for $\mathcal{Z}$; $r_{\mathcal{A}}$ for $\mathcal{A}$, r_i for party P_i). Let $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(n, z)$ random variable describing $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(n, z, r)$ where r is uniformly chosen. Let $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ denote the ensemble $\{\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(n, z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$.

Ideal functionalities and ideal protocols. Security of protocols is defined via comparing the protocol execution to an *ideal protocol* for carrying out the task at hand. A key ingredient in the ideal protocol is the *ideal functionality* that captures the desired functionality, or the specification, of that task. The ideal functionality is modeled as another ITM (representing a "trusted party") that interacts with the parties and the adversary. More specifically, in the ideal protocol for functionality $\mathcal{F}$ all parties simply hand their inputs to an ITI running $\mathcal{F}$. (We will simply call this ITI $\mathcal{F}$. The SID of $\mathcal{F}$ is the same as the SID of the ITIs running the ideal protocol. (the PID of $\mathcal{F}$ is null.)) In addition, $\mathcal{F}$ can interact with the adversary according to its code. Whenever $\mathcal{F}$ outputs a value to a party, the party immediately copies this value to its own output tape. We call the parties in the ideal protocol dummy parties. Let $\Pi(\mathcal{F})$ denote the ideal protocol for functionality $\mathcal{F}$.

Securely realizing an ideal functionality. We say that a protocol Π *emulates* protocol ϕ if for any adversary $\mathcal{A}$ there exists an adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$, on any input, can tell with non-negligible probability whether it is interacting with $\mathcal{A}$ and parties running Π, or it is interacting with S and parties running ϕ. This means that, from the point of view of the environment, running protocol Π is 'just as good' as interacting with ϕ. We say that Π *securely realizes* an ideal functionality $\mathcal{F}$ if it emulates the ideal protocol $\Pi(\mathcal{F})$. More precise definitions follow. A distribution ensemble is called *binary* if it consists of distributions over $\{0,1\}$.

Definition A.1 *Let Π and ϕ be protocols. We say that Π UC-emulates ϕ if for any adversary $\mathcal{A}$ there exists an adversary $\mathcal{S}$ such that for any environment $\mathcal{Z}$ that obeys the rules of interaction for UC security we have* $\mathrm{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}} \approx \mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$.

Definition A.2 *Let $\mathcal{F}$ be an ideal functionality and let Π be a protocol. We say that Π UC-realizes $\mathcal{F}$ if Π UC-emulates the ideal process $\Pi(\mathcal{F})$.*

The Common Reference String Model. In the common reference string (CRS) model [CF01, CLOS02], all parties in the system obtain from a trusted party a reference string, which is sampled according to a pre-specified distribution D. The reference string is referred to as the *CRS*. In the UC framework, this is modeled by an ideal functionality $\mathcal{F}_{CRS}^{D}$ that samples a string ρ from a pre-specified distribution D and sets ρ as the CRS. $\mathcal{F}_{CRS}^{D}$ is described in Figure 9.

Functionality $\mathcal{F}_{\mathbf{CRS}}^{\mathbf{D}}$

1. Upon activation with session id *sid* proceed as follows. Sample $\rho = D(r)$, where r denotes uniform random coins, and send $(\mathtt{crs}, sid, \rho)$ to the adversary.
2. On receiving $(\mathtt{crs}, sid)$ from some party send $(\mathtt{crs}, sid, \rho)$ to that party.

Figure 9: The Common Reference String Functionality.

General Functionality. We consider the general-UC functionality $\mathcal{F}$, which securely evaluates any polynomial-time (possibly randomize) function $f : (\{0,1\}^{\ell_{in}})^n \to (\{0,1\}^{\ell_{out}})^n$. The functionality $\mathcal{F}_f$ is parameterized with a function f and is described in Figure 10.

We restrict the functions to be computed to any deterministic poly-time function with n inputs and single output. This functionality has been formally defined in Figure 11. As explained in Section **??** the same protocol can be used to obtain a protocol that UC-securely realizes the general functionality $\mathcal{F}_f$ for any function f.

Functionality $\mathcal{F}_f$

$\mathcal{F}_f$ parameterized by an (possibly randomized) n-ary function f, running with parties $\mathcal{P} = \{P_1, \dots P_n\}$ (of which some may be corrupted) and an adversary $\mathcal{S}$, proceeds as follows:

1. Each party P_i (and $\mathcal{S}$ on behalf of P_i if P_i is corrupted) sends $(\mathsf{input}, \mathsf{sid}, \mathcal{P}, P_i, x_i)$ to the functionality.
2. Upon receiving the inputs from all parties, evaluate $(y_1, \dots y_n) \leftarrow f(x_1, \dots, x_n)$. For every P_i that is corrupted send adversary $\mathcal{S}$ the message $(\mathsf{output}, \mathsf{sid}, \mathcal{P}, P_i, y_i)$.
3. On receiving $(\mathsf{generateOutput}, \mathsf{sid}, \mathcal{P}, P_i)$ from $\mathcal{S}$ the ideal functionality outputs $(\mathsf{output}, \mathsf{sid}, \mathcal{P}, P_i, y_i)$ to P_i. (And ignores the message if inputs from all parties in $\mathcal{P}$ have not been received.)
4. If all the parties in $\mathcal{P}$ are corrupted then the ideal functionality reveals the internals coins used in the computation of f.

Figure 10: General Functionality.

B Proof of Theorem 3.14

In this section, we give a two-round protocol for realizing $\mathcal{F}_{\mathrm{OTCor}}$ in the client-server model where the majority of the servers are honest and the adversary is semi-honest. We start with an informal description of the model.

Client-Server Model. In the client-server model for computing $\mathcal{F}_{\mathrm{OTCor}}$, there are two clients: the sender and the receiver of the $\mathcal{F}_{\mathrm{OTCor}}$ functionality. There are $2t+1$ servers and at most t of them are corrupted. In the two-round protocol, a single message is sent from both the sender and the receiver to the $2t+1$ servers (via a private channel) and a single message is sent from all the $2t+1$ servers to the receiver (again via a private channel) which enables the receiver to obtain the output of the function.

Construction. We give the formal description of the construction in Figure 12.

Theorem B.1 *The protocol given in Figure 12 securely realizes $\mathcal{F}_{\mathrm{OTCor}}$ in the client-server model against adversaries who can corrupt an arbitrary number of clients and a minority of the servers.*

Correctness. Notice that from the linearity of Shamir's secret sharing scheme that μ_i are polynomial shares of a degree $2t$ polynomial with the constant term being $m_0(1-b)+m_1b$. The correctness of the protocol now follows directly from the reconstruction correctness of the Shamir's secret sharing scheme.

Functionality $\mathcal{F}_f$

$\mathcal{F}_f$ parameterized by an n-ary deterministic single output function f, running with parties $\mathcal{P} = \{P_1, \ldots P_n\}$ (of which some may be corrupted) and an adversary $\mathcal{S}$, proceeds as follows:

1. Each party P_i (and $\mathcal{S}$ on behalf of P_i if P_i is corrupted) sends $(\mathsf{input}, \mathsf{sid}, \mathcal{P}, P_i, x_i)$ to the functionality.
2. Upon receiving the inputs from all parties, evaluate $y \leftarrow f(x_1, \ldots, x_n)$. Send adversary $\mathcal{S}$ the message $(\mathsf{output}, \mathsf{sid}, \mathcal{P}, y)$.
3. On receiving $(\mathsf{generateOutput}, \mathsf{sid}, \mathcal{P}, P_i)$ from $\mathcal{S}$ the ideal functionality outputs $(\mathsf{output}, \mathsf{sid}, \mathcal{P}, y)$ to P_i. (And ignores the message if inputs from all parties in $\mathcal{P}$ have not been received.)

Figure 11: General Functionality for Deterministic Single Output Functionalities.

Let $\mathbb{F}$ be a finite field with at least $2t + 1$ elements and let $\mathbf{0}$ and $\mathbf{1}$ denote the additive and the multiplicative identity of $\mathbb{F}$ respectively. The sender samples two field elements $m_0, m_1 \xleftarrow{\$} \{\mathbf{0}, \mathbf{1}\}$ and the receiver's input is $b \in \{\mathbf{0}, \mathbf{1}\}$.

Round-1: The sender and the receiver do the following:

1. The sender chooses two random degree t polynomials p_s, q_s such that the free coefficients in p_s and q_s are m_0 and m_1 respectively. It evaluates these polynomials on $2t + 1$ distinct elements in the field $\mathbb{F}$ and sends each evaluation to a single server. In addition, it sends an additive secret sharing of $\mathbf{0}$ to the servers.
2. The receiver chooses two random degree t polynomials p_r, q_r such that the free coefficients in p_r and q_r are b and $\mathbf{1}$ respectively. It evaluates the polynomial on $2t+1$ distinct elements in the field and sends each evaluation to a single server. In addition, it sends an additive secret sharing of $\mathbf{0}$ to the servers.

Server's Computation: Each server S_i would have obtained (α_i, β_i) from the sender along with an additive sharing of $\mathbf{0}$ $Z_{i,s}$ and (γ_i, δ_i) from the receiver along with an additive sharing of $\mathbf{0}$ $Z_{i,r}$ in round-1. It does the following:

1. It computes $\mu_i = \alpha_i(\delta_i - \gamma_i) + \beta_i(\gamma_i)$.
2. It multiplies μ_i with the Lagrange's reconstruction coefficient (for a degree $2t$ polynomial given $2t + 1$ evaluations) and adds it with $Z_{i,s} + Z_{i,r}$ to obtain ζ_i.

Round-2: Each server S_i sends ζ_i to the receiver and the receiver reconstructs the constant term from the polynomial shares ζ_i and outputs it.

Figure 12: Two-round Oblivious Transfer Protocol

Semi-honest Security. The security of our construction is argued in a similar way to [BOGW88, IK00, AL11]. If the sender is corrupted along with a subset of at most t servers, then it follows directly from the security of Shamir's secret sharing scheme that the corrupted servers and the sender do not learn anything about the receiver's input b. Let us now assume that the receiver is corrupted along with a set of at most t servers. We describe a simulator that perfectly simulates the view of the corrupted parties.

Simulating the interaction with $\mathcal{Z}$. For every input value for the set of corrupted parties that S receives from $\mathcal{Z}$, S writes that value to $\mathcal{A}$'s input tape. Similarly, the output of $\mathcal{A}$ is written as the output on S's output tape. The simulator chooses uniform random tapes for the receiver and the corrupted servers and starts the adversary.

Simulating the interaction with $\mathcal{A}$: For every concurrent interaction with the session identifier *sid* that $\mathcal{A}$ may start, the simulator does the following:

- In round-1, the simulator samples at most $2t$ random field elements (instead of sharing m_0, m_1 as in the honest protocol) and sends them to the corrupted servers on behalf of the sender. In addition, it samples random field elements and sends them to the corrupted servers as a secret sharing of $\mathbf{0}$.

- In round-2, the simulator learns ζ_i computed by each corrupted S_i by virtue of setting the random tapes of the sender and the corrupted servers. For the rest of the uncorrupted servers, simulator samples ζ_i randomly such that $\{\zeta_i\}_{i \in [2t+1]}$ is a valid reconstruction of the field element m_b. It sends the ζ_i on behalf of every honest server to the receiver in the second round.

It is easy to see that the view of the adversary in the real protocol is exactly distributed to the view of the adversary when interacting with the simulator.

C Transformation to a Conforming Protocol

We recall the lemma from Section 4.1.

Lemma C.1 *For $s = 1$, any MPC protocol Π in the OT hybrid model can be transformed into a conforming protocol Φ while inheriting the correctness and the security of the original protocol. Furthermore, there exists a choice of s such that the number of rounds of the resulting conforming protocol is $O(n \cdot d_{\max} \cdot \mathsf{rnd})$ where $d_{\max}$ is the maximum depth of the boolean circuit computing the next message function of any party and* rnd *is the number of rounds of the original protocol Π.*

Proof Let Π be any given MPC protocol. For $s = 1$, the lemma follows directly from [GS18] where for every instance of the OT to be executed Π, we first generate the OT correlations in the offline phase and use the OT correlations to execute OT in the computation phase. We argue the special case.

We assume without loss of generality that in each round of Π, *one* party broadcasts an output of a circuit on its initial state and the messages it has received so far from other parties. Note that this restriction can be easily enforced by increasing the round complexity of the protocol by a factor of k. Let the round complexity of such a Π be p. In every round $r \in [p]$ of Π, a party P_i computes a circuit C_p on its private state and the messages received so far and broadcasts the output of the circuit to all parties. Without loss of generality, we will assume that (i) these exists q, d such that for each $r \in [p]$, width of the circuit C_p is q and the depth of the circuit is d (ii) each C_p is composed of just NAND gates with fan-in two, and (iii) each party broadcasts a message in the same number of rounds. All three of these conditions can be met by adding dummy gates and dummy round of interaction. Looking ahead, we will set s to be equal to q.

We are now ready to describe our transformed conforming protocol Φ. The protocol Φ will have $T = pd$ rounds. We let $\ell = mn + pqd$ and $\ell' = pqd/n$ and depending on ℓ the compiled protocol Φ is as follows.

- **OT Correlations Generator.** For every instance of OT to be performed in the protocol Φ, interact with $\mathcal{F}_{\mathrm{OTCor}}$ to generate OT correlations.
- $\mathsf{pre}(i, x_i)$: Sample $r_i \leftarrow \{0,1\}^m$ and $s_i \leftarrow (\{0,1\}^{q(d-1)} \| 0^q)^{p/n}$. (Observe that s_i is a pqd/n bit random string such that in every qd block, the last q bits are set to 0.) Output $z_i := x_i \oplus r_i \| 0^{\ell'}$ (where the input x_i is augmented with the OT correlations in the previous round) and $v_i := 0^{\ell/n} \| \ldots \| r_i \| s_i \| \ldots \| 0^{\ell/n}$.
- We are now ready to describe the actions $\phi_1, \cdots \phi_T$. For each $r \in [p]$, round r in Π party is expanded into d actions in Φ — namely, actions $\{\phi_k\}_k$ where $k \in \{(r-1)d + 1 \cdots rd\}$. Let P_i be the party that computes the circuits C_p and broadcasts the q output bits in round r of Π. We now describe the ϕ_k for $k \in \{(r-1)d + 1 \cdots rd\}$. For each j, we set $\phi_k = (i, (a_1, b_1, c_1), \ldots, (a_q, b_q, c_q))$ where a_j and b_j are the locations in st_i that the j^{th} gate in the k-th layer of C_p is computed on (recall that initially st_i is set to $z_1 \| z_2 \| \ldots \| z_n$). Moreover, we assign $\{c_j\}$ to be the first set of locations in st_i among the locations $(i-1)\ell/k + m + 1$ to $i\ell/n$ that has previously not been assigned to an action.

  Recall from before that on the execution of ϕ_j, party P_i sets $\mathsf{st}_{i,c_j} := \mathsf{NAND}(\mathsf{st}_{i,a_j} \oplus v_{i,a_j}, \mathsf{st}_{i,b_j} \oplus v_{i,b_j}) \oplus v_{i,c_j}$ and broadcasts st_{i,c_j} for every $j \in [s]$ to all parties.
- $\mathsf{post}(i, \mathsf{st}_i, v_i)$: Gather the masked local state of P_i, the mask v_i and the messages sent by the other parties in Π from st_i, compute the output of Π.

Now we need to argue that Φ preserves the correctness and security properties of Π. Observe that Φ is essentially the same as the protocol Π except that in Φ some additional bits are sent. Specifically, in addition to the messages that were sent in Π, in Φ parties send z_i in the preprocessing step and $q(d-1)$ additional bits for every q bits sent in Π. Note that these additional bits sent are not used in the computation of Φ. Thus these bits do not affect the functionality of Π if dropped. This ensures that Φ inherits the correctness properties of Π. Next note that each of these bits is masked by a uniform independent bit. This ensures that Φ achieves the same security properties as the underlying properties of Π.

Finally, note that by construction for all $t, t' \in [T]$ such that $t \neq t'$, we have that if $\phi_t = (\cdot, (\cdot, \cdot, c_1), \ldots, (\cdot, \cdot, c_s))$ and $\phi_{t'} = (\cdot, (\cdot, \cdot, c'_1), \ldots, (\cdot, \cdot, c'_s))$ then $\{c_j\} \cap \{c'_j\} = \varnothing$ as required. Also, the number of rounds of the transformed protocol is $O(pd)$. ■

D Proof of Theorem 4.3

We start with the description of the simulator.

Description of the Simulator. We give the description of the ideal world adversary S that simulates the view of the real world adversary $\mathcal{A}$. S will internally use the semi-honest simulator Sim_Φ for Φ and the simulator Sim_G for the randomized encoding. Recall that $\mathcal{A}$ is static and hence the set of honest parties H is known before the execution of the protocol.

Simulating the interaction with $\mathcal{Z}$. For every input value for the set of corrupted parties that S receives from $\mathcal{Z}$, S writes that value to $\mathcal{A}$'s input tape. Similarly, the output of $\mathcal{A}$ is written as the output on S's output tape.

Simulating the interaction with $\mathcal{A}$: For every concurrent interaction with the session identifier sid that $\mathcal{A}$ may start, the simulator does the following:

- **Initialization**: S uses the inputs of the corrupted parties $\{x_i\}_{i \notin H}$ and output y of the functionality f to generate a simulated view of the adversary.[16] More formally, for each $i \in [n] \setminus H$, S sends $(\mathsf{input}, \mathsf{sid}, \{P_1 \cdots P_n\}, P_i, x_i)$ to the ideal functionality implementing f and obtains the output y. Next, it executes $\mathsf{Sim}_\Phi(1^\lambda, \{x_i\}_{i \notin H}, y)$ to obtain $\{z_i\}_{i \in H}$, the random tapes for the corrupted parties, the transcript of the computation phase denoted by $\{\mathsf{Z}_{t,j}\}_{t \in [T], j \in [s]}$ where $\mathsf{Z}_{t,j}$ is the j-th bit sent in the t^{th} round of the computation phase of Φ. S starts the real-world adversary $\mathcal{A}$ with the inputs $\{x_i\}_{i \notin H}$ and random tape generated by Sim_Φ.

  Defining st^*.

  1. Set $\mathsf{st}^* := z_1 \| \ldots \| z_n$.
  2. For $t \in \{1 \cdots T\}$
     (a) Parse $\phi_t = (i^*, k, (a_1, b_1, c_1), \ldots, (a_s, b_s, c_s))$.
     (b) Updates $\mathsf{st}^*_{c_j} = \mathsf{Z}_{t,j}$.

- **Messages from S to $\mathcal{F}_{\text{OTCor}}$:** S generates the OT messages sent to the $\mathcal{F}_{\text{OTCor}}$ functionality on behalf of honest parties as follows.

  1. For each $t \in A_i$, let $\phi_t = (i, (a_1, b_1, c_1), \ldots, (a_k, b_k, c_k))$. For each $j \in [s]$, let $\alpha^*_j = \mathsf{st}^*_{a_j}$, $\beta^*_j = \mathsf{st}^*_{b_j}$. For each $t \in A_i$, $j \in [s]$, S chooses a random bit $r_{t,j,\alpha^*_j,\beta^*_j}$ and sends it to $\mathcal{F}_{\text{OTCor}}$ functionality. For all other $\alpha \neq \alpha^*_j$ and $\beta \neq \beta^*_j$, it chooses a random bit $r_{t,j,\alpha,\beta}$ but sends 0 (instead of $r_{t,j,\alpha,\beta}$) to the $\mathcal{F}_{\text{OTCor}}$ functionality. Receive for each $\omega_{t,j,\alpha^*_j,\beta^*_j}$ from the $\mathcal{F}_{\text{OTCor}}$ functionality.
  2. For each $t \in A_i$, let $\phi_t = (i^*, (a_1, b_1, c_1), \ldots, (a_k, b_k, c_k))$. For each $j \in [s]$, let $\alpha^*_j = \mathsf{st}^*_{a_j}$, $\beta^*_j = \mathsf{st}^*_{b_j}$. For each $t \neq A_i$ and $j \in [s]$, S chooses random strings $(\gamma^0_{t,j,\alpha^*_j,\beta^*_j}, \gamma^1_{t,j,\alpha^*_j,\beta^*_j})$ and sends $(\gamma^{r_{t,j,\alpha^*_j,\beta^*_j}}_{t,j,\alpha^*_j,\beta^*_j}, \gamma^{r_{t,j,\alpha^*_j,\beta^*_j}}_{t,j,\alpha^*_j,\beta^*_j})$. For all other $\alpha \neq \alpha^*_j$ and $\beta \neq \beta^*_j$, it samples random strings $(\gamma^0_{t,j,\alpha,\beta}, \gamma^1_{t,j,\alpha,\beta})$ and sends it to the $\mathcal{F}_{\text{OTCor}}$ functionality.

[16]For simplicity of exposition, we only consider the case where every party gets the same output. The proof in the more general case where parties get different outputs follows analogously.

- **Round-1 messages from S to $\mathcal{A}$:** For each $i \in H, t \in A_i, j \in [s], \alpha, \beta \in \{0,1\}$, generate $\mathsf{ots}^1_{t,j,\alpha,\beta} \leftarrow \mathsf{Z}_t \oplus r_{t,j,\alpha,\beta}$. For each $i \in H$, S sends $(z_i, \{\mathsf{ots}^1_{t,j,\alpha,\beta}\}_{t \in A_i, j \in [s], \alpha, \beta \in \{0,1\}})$ to the adversary $\mathcal{A}$ on behalf of the honest party P_i.

- **Round-1 messages from $\mathcal{A}$ to S:** Corresponding to every $i \in [n] \setminus H$, S receives from the adversary $\mathcal{A}$ the value $(z_i, \{\mathsf{ots}^1_{t,j,\alpha,\beta}\}_{t \in A_i, j \in [s] \alpha, \beta \in \{0,1\}})$ on behalf of the corrupted party P_i.

- **Round-2 messages from S to $\mathcal{A}$:** For each $i \in H$, the simulator S generates the second round message on behalf of party P_i as follows:

  1. For each $k \in [\ell]$ set $\mathbf{a}^{i,T+1}_k := 0^\lambda$.
  2. **for** each t from T down to 1,
     (a) Parse ϕ_t as $(i^*, (a_1, b_1, c_1), \ldots, (a_s, b_s, c_s))$.
     (b) For each $j \in [s]$, set $\alpha^*_j := \mathsf{st}^*_{a_j}$, $\beta^*_j := \mathsf{st}^*_{b_j}$, and $\gamma^*_j := \mathsf{st}^*_{c_j}$.
     (c) If $i = i^*$ then compute
     $$\left(\widetilde{f}^{i,t}, \{\mathbf{a}^{i,t}_k\}_{k \in [\ell]}\right) \leftarrow \mathsf{Sim}_\mathsf{G}\left(\left(\{\gamma^*_j\}_{j \in [s]}, \{\omega_{t,j,\alpha^*_j,\beta^*_j}\}_{j \in [s]}, \{\mathbf{a}^{i,t+1}_k\}_{k \in [\ell]}\right)\right).$$
     (d) If $i \neq i^*$ then compute $\mathsf{ots}^2_{t,j,\alpha^*_j,\beta^*_j} := (\mathbf{a}^{i,t+1}_{c_j,\gamma^*_j} + X_0, \mathbf{a}^{i,t+1}_{c_j,\gamma^*_j} + X_1)$ where $X_b = \gamma_{t,j,\alpha^*_j,\beta^*_j}^{b \oplus \mathsf{ots}^1_{t,j,\alpha^*_j,\beta^*_j}}$.
     $$\left(\widetilde{f}^{i,t}, \{\mathbf{a}^{i,t}_k\}_{k \in [\ell]}\right) \leftarrow \mathsf{Sim}_\mathsf{G}\left(\left(\mathsf{ots}^2_{t,j,\alpha^*_j,\beta^*_j}, \{\mathbf{a}l^{i,t+1}_k\}_{k \in [\ell] \setminus \{h\}}\right)\right).$$
  3. Send $\left(\{\widetilde{f}^{i,t}\}_{t \in [T]}, \{\mathbf{a}^{i,1}_k\}_{k \in [\ell]}\right)$ to every other party.

- **Round-2 messages from $\mathcal{A}$ to S:** For every $i \in [n] \setminus H$, S obtains the second round message from $\mathcal{A}$ on behalf of the malicious parties. Subsequent to obtaining these messages, for each $i \in H$, S sends $(\mathsf{generateOutput}, \mathsf{sid}, \{P_1 \cdots P_n\}, P_i)$ to the ideal functionality.

D.1 Proof of Indistinguishability

We now show that no environment $\mathcal{Z}$ can distinguish whether it is interacting with a real world adversary $\mathcal{A}$ or an ideal world adversary S. We prove this via an hybrid argument with $T + 1$ hybrids.

- Hybrid_{Real}: This hybrid is the same as the real world execution. Note that this hybrid is the same as hybrid Hybrid_t below with $t = 0$.

- Hybrid_t (where $t \in \{0, \ldots T\}$): Hybrid Hybrid_t (for $t \in \{1 \cdots T\}$) is the same as hybrid Hybrid_{t-1} except we change the distribution of the OT messages (from the sender and the receiver) and the randomized encoding (from the second round) that play a role in the execution of the t^{th} round of the protocol Φ; namely, the action $\phi_t = (i^*, (a_1, b_1, c_1), \ldots, (a_s, b_s, c_s))$. We describe the changes more formally below.

  We start by executing the protocol Φ on the inputs and the random coins of the honest and the corrupted parties. This yields a transcript $\{\mathsf{Z}_{t,j}\}_{t \in [T], j \in [s]}$ of the computation phase. Since the adversary is assumed to be semi-honest the execution of the protocol Φ with $\mathcal{A}$ will be consistent with $\mathsf{Z}_{t,j}$. Let st^* be the local state of the end of execution of the protocol. Finally, let $\alpha_j^* := \mathsf{st}^*_{a_j}$, $\beta_j^* := \mathsf{st}^*_{b_j}$ and $\gamma_j^* := \mathsf{st}^*_{c_j}$ for each $j \in [s]$. In hybrid Hybrid_t we make the following changes with respect to hybrid Hybrid_{t-1}:

– If $i^* \notin H$ then skip these changes. S makes two changes in how it generates messages to $\mathcal{F}_{\mathrm{OTCor}}$ functionality on behalf of P_{i^*}. First, for each $j \in [s]$, S chooses a random bit $r_{t,j,\alpha_j^*,\beta_j^*}$ and sends it to the $\mathcal{F}_{\mathrm{OTCor}}$ functionality. For all other $\alpha \neq \alpha_j^*$ and $\beta \neq \beta_j^*$, it chooses a random bit $r_{t,j,\alpha,\beta}$ but sends 0 to the $\mathcal{F}_{\mathrm{OTCor}}$ functionality. S receives $\omega_{t,j,\alpha_j^*,\beta_j^*}$ from the $\mathcal{F}_{\mathrm{OTCor}}$ functionality. Additionally, for each $i \in H, t \in A_i, j \in [s], \alpha, \beta \in \{0,1\}$, generate $\mathsf{ots}^1_{t,j,\alpha,\beta} \leftarrow \mathsf{Z}_t \oplus r_{t,j,\alpha,\beta}$.
Second, it generates the garbled circuit

$$\left(\widetilde{f}^{i^*,t}, \{\mathbf{a}_k^{i^*,t}\}_{k\in[\ell]}\right) \leftarrow \mathsf{Sim}_{\mathsf{G}}\left(\left(\{\gamma_j^*\}_{j\in[s]}, \{\omega_{t,j,\alpha_j^*,\beta_j^*}\}_{j\in[s]}, \{\mathbf{a}^{i^*,t+1}_{k,\mathsf{st}_{i,k}}\}_{k\in[\ell]}\right)\right),$$

where $\{\mathbf{a}^{i^*,t+1}_{k,\mathsf{st}_{i,k}}\}_{k\in[\ell]}$ are the honestly generates input encodings for the randomized encoding $\widetilde{\mathsf{P}}^{i^*,t+1}$.

– S makes the following two changes in how it generates messages for other honest parties P_i (i.e., $i \in H \setminus \{i^*\}$). For each $j \in [s]$, S chooses random strings $(\gamma^0_{t,j,\alpha_j^*,\beta_j^*}, \gamma^1_{t,j,\alpha_j^*,\beta_j^*})$ and sends $(\gamma_{t,j,\alpha_j^*,\beta_j^*}^{r_{t,j,\alpha_j^*,\beta_j^*}}, \gamma_{t,j,\alpha_j^*,\beta_j^*}^{r_{t,j,\alpha_j^*,\beta_j^*}})$. For all other $\alpha \neq \alpha_j^*$ and $\beta \neq \beta_j^*$, it samples random strings $(\gamma^0_{t,j,\alpha,\beta}, \gamma^1_{t,j,\alpha,\beta})$ and sends it to the $\mathcal{F}_{\mathrm{OTCor}}$ functionality.
S does not generate four $\mathsf{ots}^2_{t,j,\alpha,\beta}$ values but just one of them; namely, S generates $\mathsf{ots}^2_{t,j,\alpha_j^*,\beta_j^*} := (\mathbf{a}^{i,t+1}_{c_j,\gamma_j^*} + X_0, \mathbf{a}^{i,t+1}_{c_j,\gamma_j^*} + X_1)$ where $X_b = \gamma_{t,j,\alpha_j^*,\beta_j^*}^{b \oplus \mathsf{ots}^1_{t,j,\alpha_j^*,\beta_j^*}}$.
Second it generates the garbled circuit

$$\left(\widetilde{f}^{i,t}, \{\mathbf{a}_k^{i,t}\}_{k\in[\ell]}\right) \leftarrow \mathsf{Sim}_{\mathsf{G}}\left(\left(\{\mathsf{ots}^2_{t,j,\alpha_j^*,\beta_j^*}\}_{j\in[s]}, \{\mathbf{a}^{i,t+1}_{k,\mathsf{st}_{i,k}}\}_{k\in[\ell]\setminus\{h\}}\right)\right),$$

where $\{\mathbf{a}^{i,t+1}_{k,\mathsf{st}_{i,k}}\}_{k\in[\ell]}$ are the honestly generated input encodings for the randomized encoding $\widetilde{f}^{i,t+1}$.

Indistinguishability between Hybrid_{t-1} and Hybrid_t is proved in Lemma F.1.

- Hybrid_{T+1}: In this hybrid we just change how the transcript Z, $\{z_i\}_{i\in H}$, random coins of malicious parties and value st^* are generated. Instead of generating these using honest party inputs we generate these values by executing the simulator Sim_Φ on input $\{x_i\}_{i\in[n]\setminus H}$ and the output y obtained from the ideal functionality.

  The indistinguishability between hybrids Hybrid_T and Hybrid_{T+1} follows directly from the semi-honest security of the protocol Φ. Finally note that Hybrid_{T+1} is same as the ideal execution (i.e., the simulator described in the previous subsection).

Lemma D.1 *Assuming semi-honest security of the two-round OT protocol against a honest majority and the perfect security of the randomized encoding, for all $t \in \{1 \ldots T\}$ hybrids* Hybrid_{t-1} *and* Hybrid_t *are identically distributed.*

Proof

The indistinguishability between hybrids Hybrid_{t-1} and Hybrid_t follows by a sequence of three sub-hybrids $\mathsf{Hybrid}_{t,1}$, $\mathsf{Hybrid}_{t,2}$, and $\mathsf{Hybrid}_{t,3}$.

- $\mathsf{Hybrid}_{t,1}$: Hybrid $\mathsf{Hybrid}_{t,1}$ is same as hybrid Hybrid_{t-1} except that S now generates the randomized encoding $\widetilde{f}^{i,t}$ for each $i \in H$ in a simulated manner (rather than generating them honestly). Specifically, S generates $\widetilde{f}^{i,t}$ as follows

  - If $i = i^*$ then
  $$\left(\widetilde{f}^{i^*,t}, \{\mathbf{a}_k^{i^*,t}\}_{k\in[\ell]}\right) \leftarrow \mathsf{Sim}_{\mathsf{G}}\left(\left(\{\gamma_j^*\}_{j\in[s]}, \{\omega_{t,j,\alpha_j^*,\beta_j^*}\}_{j\in[s]}, \{\mathbf{a}_{k,\mathsf{st}_{i,k}}^{i^*,t+1}\}_{k\in[\ell]}\right)\right),$$
  where $\{\mathbf{a}_{k,\mathsf{st}_{i,k}}^{i^*,t+1}\}_{k\in[\ell]}$ are the honestly generates input encodings for the randomized encoding $\widetilde{\mathsf{P}}^{i^*,t+1}$.
  - If $i \neq i^*$ then
  $$\left(\widetilde{f}^{i,t}, \{\mathbf{a}_k^{i,t}\}_{k\in[\ell]}\right) \leftarrow \mathsf{Sim}_{\mathsf{G}}\left(\left(\{\mathsf{ots}^2_{t,j,\alpha_j^*,\beta_j^*}\}_{j\in[s]}, \{\mathbf{a}_{k,\mathsf{st}_{i,k}}^{i,t+1}\}_{k\in[\ell]\setminus\{h\}}\right)\right),$$
  where $\{\mathbf{a}_{k,\mathsf{st}_{i,k}}^{i,t+1}\}_{k\in[\ell]}$ are the honestly generated input encodings for the randomized encoding $\widetilde{f}^{i,t+1}$.

  The indistinguishability between hybrids $\mathsf{Hybrid}_{t,1}$ and Hybrid_{t-1} follows by $|H|$ invocations of security of the randomized encoding.

- $\mathsf{Hybrid}_{t,2}$: Skip this hybrid, if $i^* \notin H$. First, for each $j \in [s]$, S chooses a random bit $r_{t,j,\alpha_j^*,\beta_j^*}$ and sends it to the $\mathcal{F}_{\text{OTCor}}$ functionality. For all other $\alpha \neq \alpha_j^*$ and $\beta \neq \beta_j^*$, it chooses a random bit $r_{t,j,\alpha,\beta}$ but sends 0 to the $\mathcal{F}_{\text{OTCor}}$ functionality. S receives $\omega_{t,j,\alpha_j^*,\beta_j^*}$ from the $\mathcal{F}_{\text{OTCor}}$ functionality. Additionally, for each $i \in H, t \in A_i, j \in [s], \alpha, \beta \in \{0,1\}$, generate $\mathsf{ots}^1_{t,j,\alpha,\beta} \leftarrow \mathsf{Z}_t \oplus r_{t,j,\alpha,\beta}$.

  Indistinguishability between hybrids $\mathsf{Hybrid}_{t,1}$ and $\mathsf{Hybrid}_{t,2}$ follows statistically since only $\omega_{t,j,\alpha_j^*,\beta_j^*}$ is needed in the protocol.

- $\mathsf{Hybrid}_{t,3}$: Skip this hybrid if there does not exist $i \neq i^*$ such that $i \in H$. In this hybrid, we change how S generates the $\mathsf{ots}^2_{t,j,\alpha,\beta}$ on behalf of every honest party P_i such that $i \in H \setminus \{i^*\}$ for all choices of $\alpha, \beta \in \{0,1\}$. More specifically, for each $j \in [s]$, S chooses random strings $(\gamma^0_{t,j,\alpha_j^*,\beta_j^*}, \gamma^1_{t,j,\alpha_j^*,\beta_j^*})$ and sends $(\gamma^{r_{t,j,\alpha_j^*,\beta_j^*}}_{t,j,\alpha_j^*,\beta_j^*}, \gamma^{r_{t,j,\alpha_j^*,\beta_j^*}}_{t,j,\alpha_j^*,\beta_j^*})$. For all other $\alpha \neq \alpha_j^*$ and $\beta \neq \beta_j^*$, it samples random strings $(\gamma^0_{t,j,\alpha,\beta}, \gamma^1_{t,j,\alpha,\beta})$ and sends it to the $\mathcal{F}_{\text{OTCor}}$ functionality.

  S does not generate four $\mathsf{ots}^2_{t,j,\alpha,\beta}$ values but just one of them; namely, S generates $\mathsf{ots}^2_{t,j,\alpha_j^*,\beta_j^*} := (\mathbf{a}^{i,t+1}_{c_j,\gamma_j^*} + X_0, \mathbf{a}^{i,t+1}_{c_j,\gamma_j^*} + X_1)$ where $X_b = \gamma^{b\oplus\mathsf{ots}^1_{t,j,\alpha_j^*,\beta_j^*}}_{t,j,\alpha_j^*,\beta_j^*}$.

  Indistinguishability between hybrids $\mathsf{Hybrid}_{t,2}$ and $\mathsf{Hybrid}_{t,3}$ follows statistically since $r_{t,j,\alpha_j^*,\beta_j^*}$ is the receiver's choice bit and $\gamma^{1\oplus r_{t,j,\alpha_j^*,\beta_j^*}}_{t,j,\alpha_j^*,\beta_j^*}$ is statistically hidden. Finally, observe that $\mathsf{Hybrid}_{t,3}$ is the same as hybrid Hybrid_t.

■

E Conforming Protocols with Secure Channels

We recall the lemma from Section 6.

Lemma E.1 *There exists a choice of s such that any protocol Π using secure channels can be transformed into a conforming protocol Φ inheriting the correctness and the security property as that of Π and the number of rounds of Φ being $O(n^2 \cdot d_{\max} \cdot \mathsf{rnd})$ where $d_{\max}$ is the maximum depth of the boolean circuit computing the next message function of any party and rnd is the number of rounds of the original protocol Π.*

Proof Let Π be any given MPC protocol with secure channels. We assume without loss of generality that in each round of Π, *one* party sends a private message to an another party and this message is an output of a circuit on its initial state and the messages it has received so far from other parties. Note that this restriction can be easily enforced by increasing the round complexity of the protocol by a factor of n^2 (because in a particular round of Π, every party might send a private message to every other party). Let the round complexity of such a Π be p. In every round $r \in [p]$ of Π, a party P_i computes a circuit C_p on its private state and the messages received so far and sends the output of the circuit privately to a party P_ℓ. Without loss of generality, we will assume that (i) these exists q, d such that for each $r \in [p]$, width of the circuit C_p is q and the depth of the circuit is d (ii) each C_p is composed of just NAND gates with fan-in two, and (iii) each party broadcasts a message in the same number of rounds. All three of these conditions can be met by adding dummy gates and dummy round of interaction. Looking ahead, we will set s to be equal to q.

We are now ready to describe our transformed conforming protocol Φ. The protocol Φ will have $T = pd$ rounds. We let $\ell = mn + pqd$ and $\ell' = pqd/n$ and depending on ℓ the compiled protocol Φ is as follows.

- We first describe the actions $\phi_1, \cdots \phi_T$. For each $r \in [p]$, round r in Π party is expanded into d actions in Φ — namely, actions $\{\phi_k\}_k$ where $k \in \{(r-1)d + 1 \cdots rd\}$. Let P_i be the party that computes the circuits C_p and sends the q output bits in round r as a private message to party $P_{k'}$. We now describe the ϕ_k for $k \in \{(r-1)d + 1 \cdots rd\}$. For each $k \neq rd$, we set $\phi_k = (i, \perp, (a_1, b_1, c_1), \ldots, (a_q, b_q, c_q))$, otherwise we set $\phi_k = (i, k', (a_1, b_1, c_1), \ldots, (a_q, b_q, c_q))$, where a_j and b_j are the locations in st_i that the j^{th} gate in the k-th layer of C_p is computed on (recall that initially st_i is set to $z_1 \| z_2 \| \ldots \| z_k$). Moreover, we assign $\{c_j\}$ to be the first set of locations in st_i among the locations $(i-1)\ell/k + m + 1$ to $i\ell/n$ that has previously not been assigned to an action.

- $\mathsf{pre}(i, x_i)$:

  1. Sample $r_i \leftarrow \{0,1\}^m$ and $s_i \leftarrow (\{0,1\}^{qd})^{p/k}$. The distribution from which s_i is sampled varies from the case of conforming protocols with a broadcast channel. Note that unlike the case here where s_i is chosen uniformly random, s_i in the case of broadcast channels has some bits fixed to 0 (refer Appendix C). Output $z_i := x_i \oplus r_i \| 0^{\ell'}$ and $v_i := 0^{\ell/n} \| \ldots \| r_i \| s_i \| \ldots \| 0^{\ell/n}$.
  2. For every $t \in [T]$, parse the action ϕ_t as $(i^*, k, (a_1, b_1, c_1), \ldots, (a_s, b_s, c_s))$. If $i = i^*$ and $k \neq \perp$, then send $v_{i,c_1}, v_{i,c_2}, \ldots, v_{i,c_s}$ to the party P_k via private channels. Party P_k updates $v_{k,c_j} = v_{i,c_j}$ for every $j \in [s]$.

- **Computation Phase.** Recall from before that on the execution of ϕ_j, party P_i sets $\mathsf{st}_{i,c_j} := \mathsf{NAND}(\mathsf{st}_{i,a_j} \oplus v_{i,a_j}, \mathsf{st}_{i,b_j} \oplus v_{i,b_j}) \oplus v_{i,c_j}$ and broadcasts st_{i,c_j} for every $j \in [s]$ to all parties.
- $\mathsf{post}(i, \mathsf{st}_i, v_i)$: Gather the masked local state of P_i, the mask v_i and the messages sent by the other parties in Π from st_i, compute the output of Π.

Now we need to argue that Φ preserves the correctness and security properties of Π. We first start with the correctness. For every round, where a party P_i sends a message to P_k, the conforming protocol Φ sends the message masked with random bits. Furthermore, these random masking bits are sent to P_k via a private channel. So essentially, Φ sends all the messages that Π sends and sends additional bits. Specifically, in addition to the masked private messages that were sent in Π, in Φ parties send z_i in the preprocessing step and $q(d-1)$ additional bits for every q bits sent in Π. Note that these additional bits sent are not used in the computation of Φ. Thus these bits do not affect the functionality of Π if dropped. This ensures that Φ inherits the correctness properties of Π. Next, for each private message sent from P_i and P_k, the message is masked by an uniformly chosen random string. This random string is available only to P_i and P_k and thus, for an external observer these messages are indistinguishable to random strings. Further, the messages (apart from the private messages) sent by a party P_i are masked with an uniform random string known only to P_i. This ensures that Φ achieves the same security properties as the underlying properties of Π.

Finally, note that by construction for all $t, t' \in [T]$ such that $t \neq t'$, we have that if $\phi_t = (\cdot, (\cdot, \cdot, c_1), \ldots, (\cdot, \cdot, c_s))$ and $\phi_{t'} = (\cdot, (\cdot, \cdot, c'_1), \ldots, (\cdot, \cdot, c'_s))$ then $\{c_j\} \cap \{c'_j\} = \varnothing$ as required. Also, the number of rounds of the transformed protocol is $O(pd)$. ■

F Proof of Theorem 6.2

We start with the description of the simulator.

Description of the Simulator. We give the description of the ideal world adversary S that simulates the view of the real world adversary $\mathcal{A}$. S will internally use the semi-honest simulator Sim_Φ for Φ and the simulator Sim_G for the randomized encoding. Recall that $\mathcal{A}$ is static and hence the set of honest parties H is known before the execution of the protocol.

Simulating the interaction with $\mathcal{Z}$. For every input value for the set of corrupted parties that S receives from $\mathcal{Z}$, S writes that value to $\mathcal{A}$'s input tape. Similarly, the output of $\mathcal{A}$ is written as the output on S's output tape.

Simulating the interaction with $\mathcal{A}$: For every concurrent interaction with the session identifier sid that $\mathcal{A}$ may start, the simulator does the following:

- **Initialization**: S uses the inputs of the corrupted parties $\{x_i\}_{i \notin H}$ and output y of the functionality f to generate a simulated view of the adversary.[17] More formally, for each $i \in [n] \setminus H$, S sends $(\mathsf{input}, \mathsf{sid}, \{P_1 \cdots P_n\}, P_i, x_i)$ to the ideal functionality implementing f and obtains the output y. Next, it executes $\mathsf{Sim}_\Phi(1^\lambda, \{x_i\}_{i \notin H}, y)$ to obtain $\{z_i\}_{i \in H}$, the

[17]For simplicity of exposition, we only consider the case where every party gets the same output. The proof in the more general case where parties get different outputs follows analogously.

random tapes for the corrupted parties, the transcript of the computation phase denoted by $\{\mathsf{Z}_{t,j}\}_{t\in[T],j\in[s]}$ where $\mathsf{Z}_{t,j}$ is the j-th bit sent in the t^{th} round of the computation phase of Φ. S starts the real-world adversary $\mathcal{A}$ with the inputs $\{z_i\}_{i\in H}$ and random tape generated by Sim_Φ.

Defining st^*.

1. Set $\mathsf{st}^* := z_1\|\ldots\|z_n$.
2. For $t \in \{1\cdots T\}$
   (a) Parse $\phi_t = (i^*, k, (a_1, b_1, c_1), \ldots, (a_s, b_s, c_s))$.
   (b) Updates $\mathsf{st}^*_{c_j} = \mathsf{Z}_{t,j}$.

- **Round-1 messages from S to $\mathcal{A}$:** Next S generates the OT messages on behalf of honest parties as follows.
  1. For each $t \in A_i$, let $\phi_t = (i, k, (a_1, b_1, c_1), \ldots, (a_k, b_k, c_k))$. For each $j \in [s]$, let $\alpha^*_j = \mathsf{st}^*_{a_j}$, $\beta^*_j = \mathsf{st}^*_{b_j}$. For each $t \in A_i$, $j \in [s]$, S chooses a random bit $r_{t,j,\alpha^*_j,\beta^*_j}$ and generates $\mathsf{OT}_1(r_{t,j,\alpha^*_j,\beta^*_j})$. For all other $\alpha \neq \alpha^*_j$ and $\beta \neq \beta^*_j$, it chooses a random bit $r_{t,j,\alpha,\beta}$ but samples some random shares and sends them to the corrupted parties. In particular, it does not generate secret sharing of the random bit $r_{t,j,\alpha,\beta}$.
  2. For each $t \in A_i$, let $\phi_t = (i^*, k, (a_1, b_1, c_1), \ldots, (a_k, b_k, c_k))$. For each $j \in [s]$, let $\alpha^*_j = \mathsf{st}^*_{a_j}$, $\beta^*_j = \mathsf{st}^*_{b_j}$. For each $t \neq A_i$ and $j \in [s]$, S chooses random strings $(\gamma^0_{t,j,\alpha^*_j,\beta^*_j}, \gamma^1_{t,j,\alpha^*_j,\beta^*_j})$ and generates $\mathsf{OT}_2((\gamma^{\mathsf{st}^*_{c_j}}_{t,j,\alpha^*_j,\beta^*_j}, \gamma^{\mathsf{st}^*_{c_j}}_{t,j,\alpha^*_j,\beta^*_j}))$. For all other $\alpha \neq \alpha^*_j$ and $\beta \neq \beta^*_j$, it samples some random shares and sends them to the corrupted parties.

  For each $i \in H, t \in A_i, j \in [s], \alpha, \beta \in \{0,1\}$, generate $\mathsf{ots}^1_{t,j,\alpha,\beta} \leftarrow \mathsf{Z}_t \oplus r_{t,j,\alpha,\beta}$. For each $i \in H$, S sends $(z_i, \{\mathsf{ots}^1_{t,j,\alpha,\beta}\}_{t\in A_i,j\in[s],\alpha,\beta\in\{0,1\}})$ to the adversary $\mathcal{A}$ on behalf of the honest party P_i.

- **Round-1 messages from $\mathcal{A}$ to S:** Corresponding to every $i \in [n] \setminus H$, S receives from the adversary $\mathcal{A}$ the value $(z_i, \{\mathsf{ots}^1_{t,j,\alpha,\beta}\}_{t\in A_i,j\in[s]\alpha,\beta\in\{0,1\}})$ on behalf of the corrupted party P_i.

- **Round-2 messages from S to $\mathcal{A}$:** For each $i \in H$, the simulator S generates the second round message on behalf of party P_i as follows:
  1. For each $k \in [\ell]$ set $\mathbf{a}^{i,T+1}_k := 0^\lambda$.
  2. **for** each t from T down to 1,
     (a) Parse ϕ_t as $(i^*, k, (a_1, b_1, c_1), \ldots, (a_s, b_s, c_s))$.
     (b) For each $j \in [s]$, set $\alpha^*_j := \mathsf{st}^*_{a_j}$, $\beta^*_j := \mathsf{st}^*_{b_j}$, and $\gamma^*_j := \mathsf{st}^*_{c_j}$.
     (c) For each $t \in [T]$, $j \in [s]$, let $\omega^{i,k}_{t,j,\alpha^*_j,\beta^*_j}$ be the share corresponding to linear computation on the OT between i^* and k for every $k \in [n] \setminus \{i^*\}$. We will let $\omega^i_{t,j,\alpha^*_j,\beta^*_j} = \{\omega^{i,k}_{t,j,\alpha^*_j,\beta^*_j}\}_{k\in[n]\setminus\{i^*\}}$.
     (d) If $i = i^*$ then compute
     $$\left(\widetilde{f}^{i,t}, \{\mathbf{a}^{i,t}_k\}_{k\in[\ell]}\right) \leftarrow \mathsf{Sim}_{\mathsf{G}}\left(\left(\{\gamma^*_j\}_{j\in[s]}, \{\omega^{i^*}_{t,j,\alpha^*_j,\beta^*_j}\}_{j\in[s]}, \{\mathbf{a}^{i,t+1}_k\}_{k\in[\ell]}\right)\right).$$

(e) If $i \neq i^*$ then compute $\mathsf{ots}^2_{t,j,\alpha^*_j,\beta^*_j} := (\mathbf{a}^{i,t+1}_{c_j,\gamma^*_j}+X_0, \mathbf{a}^{i,t+1}_{c_j,\gamma^*_j}+X_1)$ where $X_b = \gamma^{b \oplus \mathsf{ots}^1_{t,j,\alpha^*_j,\beta^*_j}}_{t,j,\alpha^*_j,\beta^*_j}$.

$$\left(\widetilde{f}^{i,t}, \{\mathbf{a}^{i,t}_k\}_{k\in[\ell]}\right) \leftarrow \mathsf{Sim}_{\mathsf{G}}\left(\left(\{\omega^{i^*}_{t,j,\alpha^*_j,\beta^*_j}\}_{j\in[s]}, \mathsf{ots}^2_{t,j,\alpha^*_j,\beta^*_j}, \{\mathbf{a}l^{i,t+1}_k\}_{k\in[\ell]\setminus\{h\}}\right)\right).$$

3. Send $\left(\{\widetilde{f}^{i,t}\}_{t\in[T]}, \{\mathbf{a}^{i,1}_k\}_{k\in[\ell]}\right)$ to every other party.

- **Round-2 messages from $\mathcal{A}$ to S:** For every $i \in [n]\setminus H$, S obtains the second round message from $\mathcal{A}$ on behalf of the malicious parties. Subsequent to obtaining these messages, for each $i \in H$, S sends $(\mathsf{generateOutput}, \mathsf{sid}, \{P_1 \cdots P_n\}, P_i)$ to the ideal functionality.

F.1 Proof of Indistinguishability

We now show that no environment $\mathcal{Z}$ can distinguish whether it is interacting with a real world adversary $\mathcal{A}$ or an ideal world adversary S. We prove this via an hybrid argument with $T+1$ hybrids.

- Hybrid_{Real}: This hybrid is the same as the real world execution.

  Note that this hybrid is the same as hybrid Hybrid_t below with $t = 0$.

- Hybrid_t (where $t \in \{0, \ldots T\}$): Hybrid Hybrid_t (for $t \in \{1 \cdots T\}$) is the same as hybrid Hybrid_{t-1} except we change the distribution of the OT messages (from the sender and the receiver) and the randomized encoding (from the second round) that play a role in the execution of the t^{th} round of the protocol Φ; namely, the action $\phi_t = (i^*, j, (a_1, b_1, c_1), \ldots, (a_s, b_s, c_s))$. We describe the changes more formally below.

  We start by executing the protocol Φ on the inputs and the random coins of the honest and the corrupted parties. This yields a transcript $\{\mathsf{Z}_{t,j}\}_{t\in[T],j\in[s]}$ of the computation phase. Since the adversary is assumed to be semi-honest the execution of the protocol Φ with $\mathcal{A}$ will be consistent with $\mathsf{Z}_{t,j}$. Let st^* be the local state of the end of execution of the protocol. Finally, let $\alpha^*_j := \mathsf{st}^*_{a_j}$, $\beta^*_j := \mathsf{st}^*_{b_j}$ and $\gamma^*_j := \mathsf{st}^*_{c_j}$ for each $j \in [s]$. In hybrid Hybrid_t we make the following changes with respect to hybrid Hybrid_{t-1}:

  – If $i^* \notin H$ then skip these changes. S makes two changes in how it generates messages on behalf of P_{i^*}. First, for each $j \in [s]$, S chooses a random bit $r_{t,j,\alpha^*_j,\beta^*_j}$ and generates $\mathsf{OT}_1(r_{t,j,\alpha^*_j,\beta^*_j})$. For all other $\alpha \neq \alpha^*_j$ and $\beta \neq \beta^*_j$, it chooses a random bit $r_{t,j,\alpha,\beta}$ but samples some random shares and sends them to the corrupted parties. In particular, it does not generate secret sharing of the random bit $r_{t,j,\alpha,\beta}$. Additionally, for each $i \in H, t \in A_i, j \in [s], \alpha, \beta \in \{0,1\}$, generate $\mathsf{ots}^1_{t,j,\alpha,\beta} \leftarrow \mathsf{Z}_t \oplus r_{t,j,\alpha,\beta}$.
  Second, it generates the garbled circuit

  $$\left(\widetilde{f}^{i^*,t}, \{\mathbf{a}^{i^*,t}_k\}_{k\in[\ell]}\right) \leftarrow \mathsf{Sim}_{\mathsf{G}}\left(\left(\{\gamma^*_j\}_{j\in[s]}, \{\omega^{i^*}_{t,j,\alpha^*_j,\beta^*_j}\}_{j\in[s]}, \{\mathbf{a}^{i^*,t+1}_{k,\mathsf{st}_{i,k}}\}_{k\in[\ell]}\right)\right),$$

  where $\{\mathbf{a}^{i^*,t+1}_{k,\mathsf{st}_{i,k}}\}_{k\in[\ell]}$ are the honestly generates input encodings for the randomized encoding $\widetilde{\mathsf{P}}^{i^*,t+1}$.

– S makes the following two changes in how it generates messages for other honest parties P_i (i.e., $i \in H \setminus \{i^*\}$). For each $j \in [s]$, S chooses random strings $(\gamma^0_{t,j,\alpha^*_j,\beta^*_j}, \gamma^1_{t,j,\alpha^*_j,\beta^*_j})$ and generates $\mathsf{OT}_2((\gamma^{\mathsf{st}^*_{c_j}}_{t,j,\alpha^*_j,\beta^*_j}, \gamma^{\mathsf{st}^*_{c_j}}_{t,j,\alpha^*_j,\beta^*_j}))$. For all other $\alpha \neq \alpha^*_j$ and $\beta \neq \beta^*_j$, it samples some random shares and sends them to the corrupted parties.
  S does not generate four $\mathsf{ots}^2_{t,j,\alpha,\beta}$ values but just one of them; namely, S generates $\mathsf{ots}^2_{t,j,\alpha^*_j,\beta^*_j} := (\mathbf{a}^{i,t+1}_{c_j,\gamma^*_j} + X_0, \mathbf{a}^{i,t+1}_{c_j,\gamma^*_j} + X_1)$ where $X_b = \gamma^{b \oplus \mathsf{ots}^1_{t,j,\alpha^*_j,\beta^*_j}}_{t,j,\alpha^*_j,\beta^*_j}$.
  Second it generates the garbled circuit

$$\left(\widetilde{f}^{i,t}, \{\mathbf{a}^{i,t}_k\}_{k\in[\ell]}\right) \leftarrow \mathsf{Sim}_{\mathsf{G}}\left(\left(\{\omega^{i^*}_{t,j,\alpha^*_j,\beta^*_j}\}_{j\in[s]}, \mathsf{ots}^2_{t,j,\alpha^*_j,\beta^*_j}, \{\mathbf{a}^{i,t+1}_{k,\mathsf{st}_{i,k}}\}_{k\in[\ell]\setminus\{h\}}\right)\right),$$

  where $\{\mathbf{a}^{i,t+1}_{k,\mathsf{st}_{i,k}}\}_{k\in[\ell]}$ are the honestly generated input encodings for the randomized encoding $\widetilde{f}^{i,t+1}$.

Indistinguishability between Hybrid_{t-1} and Hybrid_t is proved in Lemma F.1.

- Hybrid_{T+1}: In this hybrid we just change how the transcript Z, $\{z_i\}_{i\in H}$, random coins of malicious parties and value st^* are generated. Instead of generating these using honest party inputs we generate these values by executing the simulator Sim_Φ on input $\{x_i\}_{i\in[n]\setminus H}$ and the output y obtained from the ideal functionality.

  The indistinguishability between hybrids Hybrid_T and Hybrid_{T+1} follows directly from the semi-honest security of the protocol Φ. Finally note that Hybrid_{T+1} is same as the ideal execution (i.e., the simulator described in the previous subsection).

Lemma F.1 *Assuming semi-honest security of the two-round OT protocol against a honest majority and the perfect security of the randomized encoding, for all $t \in \{1 \dots T\}$ hybrids Hybrid_{t-1} and Hybrid_t are identically distributed.*

Proof
The indistinguishability between hybrids Hybrid_{t-1} and Hybrid_t follows by a sequence of three sub-hybrids $\mathsf{Hybrid}_{t,1}$, $\mathsf{Hybrid}_{t,2}$, and $\mathsf{Hybrid}_{t,3}$.

- $\mathsf{Hybrid}_{t,1}$: Hybrid $\mathsf{Hybrid}_{t,1}$ is same as hybrid Hybrid_{t-1} except that S now generates the randomized encoding $\widetilde{f}^{i,t}$ for each $i \in H$ in a simulated manner (rather than generating them honestly). Specifically, S generates $\widetilde{f}^{i,t}$ as follows

  – If $i = i^*$ then

$$\left(\widetilde{f}^{i^*,t}, \{\mathbf{a}^{i^*,t}_k\}_{k\in[\ell]}\right) \leftarrow \mathsf{Sim}_{\mathsf{G}}\left(\left(\{\gamma^*_j\}_{j\in[s]}, \{\omega^{i^*}_{t,j,\alpha^*_j,\beta^*_j}\}_{j\in[s]}, \{\mathbf{a}^{i^*,t+1}_{k,\mathsf{st}_{i,k}}\}_{k\in[\ell]}\right)\right),$$

  where $\{\mathbf{a}^{i^*,t+1}_{k,\mathsf{st}_{i,k}}\}_{k\in[\ell]}$ are the honestly generates input encodings for the randomized encoding $\widetilde{\mathsf{P}}^{i^*,t+1}$.

  – If $i \neq i^*$ then

$$\left(\widetilde{f}^{i,t}, \{\mathbf{a}^{i,t}_k\}_{k\in[\ell]}\right) \leftarrow \mathsf{Sim}_{\mathsf{G}}\left(\left(\{\omega^{i^*}_{t,j,\alpha^*_j,\beta^*_j}\}_{j\in[s]}, \mathsf{ots}^2_{t,j,\alpha^*_j,\beta^*_j}, \{\mathbf{a}^{i,t+1}_{k,\mathsf{st}_{i,k}}\}_{k\in[\ell]\setminus\{h\}}\right)\right),$$

where $\{\mathbf{a}^{i,t+1}_{k,\mathsf{st}_{i,k}}\}_{k\in[\ell]}$ are the honestly generated input encodings for the randomized encoding $\widetilde{f}^{i,t+1}$.

The indistinguishability between hybrids $\mathsf{Hybrid}_{t,1}$ and Hybrid_{t-1} follows by $|H|$ invocations of security of the randomized encoding.

- $\mathsf{Hybrid}_{t,2}$: Skip this hybrid, if $i^* \notin H$. This hybrid is same as $\mathsf{Hybrid}_{t,1}$ except that we change how S generates the Round-1 message on behalf of P_{i^*}. First, for each $j \in [s]$, S chooses a random bit $r_{t,j,\alpha^*_j,\beta^*_j}$ and generates $\mathsf{OT}_1(r_{t,j,\alpha^*_j,\beta^*_j})$. For all other $\alpha \neq \alpha^*_j$ and $\beta \neq \beta^*_j$, it chooses a random bit $r_{t,j,\alpha,\beta}$ but samples some random shares and sends them to the corrupted parties. In particular, it does not generate secret sharing of the random bit $r_{t,j,\alpha,\beta}$. Additionally, for each $i \in H, t \in A_i, j \in [s], \alpha, \beta \in \{0,1\}$, generate $\mathsf{ots}^1_{t,j,\alpha,\beta} \leftarrow \mathsf{Z}_t \oplus r_{t,j,\alpha,\beta}$.

  Indistinguishability between hybrids $\mathsf{Hybrid}_{t,1}$ and $\mathsf{Hybrid}_{t,2}$ follows directly from the perfect receiver security of the oblivious transfer.

- $\mathsf{Hybrid}_{t,3}$: Skip this hybrid if there does not exist $i \neq i^*$ such that $i \in H$. In this hybrid, we change how S generates the $\mathsf{ots}^2_{t,j,\alpha,\beta}$ on behalf of every honest party P_i such that $i \in H \setminus \{i^*\}$ for all choices of $\alpha, \beta \in \{0,1\}$. More specifically, for each $j \in [s]$, S chooses random strings $(\gamma^0_{t,j,\alpha^*_j,\beta^*_j}, \gamma^1_{t,j,\alpha^*_j,\beta^*_j})$ and generates $\mathsf{OT}_2((\gamma^{\mathsf{st}^*_{c_j}}_{t,j,\alpha^*_j,\beta^*_j}, \gamma^{\mathsf{st}^*_{c_j}}_{t,j,\alpha^*_j,\beta^*_j}))$. For all other $\alpha \neq \alpha^*_j$ and $\beta \neq \beta^*_j$, it samples some random shares and sends them to the corrupted parties.

  S does not generate four $\mathsf{ots}^2_{t,j,\alpha,\beta}$ values but just one of them; namely, S generates $\mathsf{ots}^2_{t,j,\alpha^*_j,\beta^*_j} := (\mathbf{a}^{i,t+1}_{c_j,\gamma^*_j} + X_0, \mathbf{a}^{i,t+1}_{c_j,\gamma^*_j} + X_1)$ where $X_b = \gamma^{b \oplus \mathsf{ots}^1_{t,j,\alpha^*_j,\beta^*_j}}_{t,j,\alpha^*_j,\beta^*_j}$.

  Indistinguishability between hybrids $\mathsf{Hybrid}_{t,2}$ and $\mathsf{Hybrid}_{t,3}$ follows directly from the perfect sender security of the oblivious transfer. Finally, observe that $\mathsf{Hybrid}_{t,3}$ is the same as hybrid Hybrid_t.

■