k-Round MPC from k-Round OT via Garbled Interactive Circuits

Fabrice Benhamouda[*] Huijia Lin[†]

Abstract

We present new constructions of *round-efficient*, or even *round-optimal*, Multi-Party Computation (MPC) protocols from Oblivious Transfer (OT) protocols. Our constructions establish a *tight* connection between MPC and OT: In the setting of semi-honest security, for any $k \geq 2$, k-round semi-honest OT is *necessary and complete* for k-round semi-honest MPC. In the round-optimal case of $k = 2$, we obtain 2-round semi-honest MPC from 2-round semi-honest OT, resolving the round complexity of semi-honest MPC assuming weak and necessary assumption. In comparison, previous 2-round constructions rely on either the heavy machinery of indistinguishability obfuscation or witness encryption, or the algebraic structure of bilinear pairing groups. More generally, for an arbitrary number of rounds k, all previous constructions of k-round semi-honest MPC require at least OT with k' rounds for $k' \leq \lfloor k/2 \rfloor$.

In the setting of malicious security, we show: For any $k \geq 5$, k-round malicious OT is *necessary and complete* for k-round malicious MPC. In fact, OT satisfying a weaker notion of *delayed-semi-malicious* security suffices. In the common reference string model, for any $k \geq 2$, we obtain k-round malicious Universal Composable (UC) protocols from any k-round semi-malicious OT and non-interactive zero-knowledge. Previous 5-round protocols in the plain model, and 2-round protocols in the common reference string model all require algebraic assumptions such as DDH or LWE.

At the core of our constructions is a new framework for *garbling interactive circuits*. Roughly speaking, it allows for garbling interactive machines that participates in interactions of a special form. The garbled machine can emulate the original interactions receiving messages sent in the *clear* (without being encoded using secrets), and reveals only the transcript of the interactions, provided that the transcript is *computationally uniquely defined*. We show that garbled interactive circuits for the purpose of constructing MPC can be implemented using OT. Along the way, we also propose a new primitive of *witness selector* that strengthens witness encryption, and a new notion of *zero-knowledge functional commitments*.

[*] `fabrice.benhamouda@normalesup.org`, IBM Research, Yorktown Heights, US

[†] `rachel.lin@cs.ucsb.edu`, University of California, Santa Barbara, US

Contents

1 Introduction

A *Multi-Party Computation (MPC) protocol* allows m mutually distrustful parties to securely compute a functionality $f(\bar{x})$ of their corresponding private inputs $\bar{x} = x_1, ..., x_m$, such that party P_i receives the i-th component of $f(\bar{x})$. The *semi-honest security* guarantees that *honest-but-curious* parties who follow the specification of the protocol learn nothing more than their prescribed outputs. The stronger *malicious security* guarantees that even malicious parties who may deviate from the protocol, cannot learn more information nor manipulate the outputs of the honest parties. MPC protocols for computing general functionalities are central primitives in cryptography and have been studied extensively. An important question is: "*how many rounds of interactions do general MPC protocols need, and under what assumptions?*"

The round complexity of *2-Party Computation* (2PC) was resolved more than three decades ago: Yao [Yao82, Yao86] gave a construction of general semi-honest 2PC protocols that have only *two rounds* of interaction (where parties have access to a simultaneous broadcast channel[1]), using garbled circuits and a 2-message semi-honest Oblivious Transfer (OT) protocol. The round complexity is optimal, as any one-round protocol is trivially broken. Moreover, the underlying assumption of 2-message semi-honest OT is weak and necessary[2].

In contrast, constructing round-efficient MPC protocols turned out to be more challenging. The first general construction [GMW87] requires a high number of rounds, $O(d)$, proportional to the depth d of the computation. Later, Beaver, Micali, and Rogaway (BMR) reduced the round complexity to a constant using garbled circuits [BMR90]. However, the *exact* round complexity of MPC remained open until recently. By relying on specific algebraic assumptions, a recent line of works constructed *i)* 2-round MPC protocols relying on trusted infrastructure (e.g., a common reference string) assuming LWE [AJL+12, MW16, CM15, BP16, PS16] or DDH [BGI16, BGI17, BGI+18], and *ii)* 2-round protocols in the plain model from indistinguishability obfuscation or witness encryption with NIZK [GGHR14, GP15, CGP15, DKR15, GLS15], or bilinear groups [GS17]. However, all these constructions heavily exploit the algebraic structures of the underlying assumptions, or rely on the heavy machinery of obfuscation or witness encryption.

The state-of-the-art for malicious security is similar. Garg, Mukherjee, Pandey, Polychroniadou [GMPP16] showed that 4 round is optimal for malicious MPC. So far, there are constructions of *i)* 5-round protocols from DDH [ACJ17], and *ii)* 4-round protocols from subexponentially secure DDH [ACJ17], or subexponentially secure LWE and adaptive commitments[3] [BHP17]. In general, for any number of round k, all known constructions of semi-honest or malicious MPC require at least k' round OT for $k' \leq \lfloor k/2 \rfloor$. We ask the question,

Can we have round-optimal MPC protocols from weak and necessary assumptions?

We completely resolve this question in the semi-honest setting, constructing 2-round semi-honest MPC from 2-round semi-honest OT, and make significant progress in the malicious setting, constructing 5-round malicious MPC from 5-round delayed-semi-malicious OT, a weaker primitive than

[1] Using the simultaneous broadcast channel, every party can simultaneously broadcast a message to all other parties. A malicious adversary can rush in the sense that in every round it receives the messages broadcast by honest parties first before choosing its own messages. In the 2PC setting, if both parties receive outputs, Yao's protocols need simultaneous broadcast channel.

[2] A 2-round OT protocol consists of one message from the receiver, followed by another one from the sender. It is implied by 2-round 2PC protocols using the simultaneous broadcast channel.

[3] That is, CCA commitments introduced in [CLP10].

malicious OT. Our results are obtained via a new notion of *garbling interactive circuits*. Roughly speaking, classical garbling turns a computation, given by a circuit C and an input x, into another one $(\hat{C}, \hat{x})$ that reveals only the output $C(x)$. Our new notion considers garbling a machine participating in an interaction: Let C (with potentially hardcoded input x) be an interactive machine that interacts with an oracle $\mathcal{O}$, which is a *non-deterministic algorithm* that computes its replies to C's messages, *depending on some witnesses* $\bar{w}$. Garbling interactive machine turns C into $\hat{C}$, which can emulate the interaction between C and $\mathcal{O}$, given the witnesses $\bar{w}$ in the clear (without any secret encoding). It is guaranteed that $\hat{C}$ reveals only the transcript of messages in the interaction and nothing else, provided that the transcript is *computationally uniquely defined*, that is, it is computationally hard to find two different witnesses $\bar{w}, \bar{w}'$ that lead to different transcripts.

1.1 Our Contributions

SEMI-HONEST SECURITY: We construct 2-round semi-honest MPC protocols in the plain model from 2-round semi-honest OT. Our construction can be generalized to an arbitrary number of rounds, establishing a tight connection between MPC and OT: For any k, *k-round OT is necessary and complete for k-round MPC.*[4]

Theorem 1.1 (Semi-Honest Security)**.** *For any $k \geq 2$, there is a k-*round *semi-honest MPC protocol for any functionality f, from any k-*round *semi-honest OT protocol.*

The above theorem resolves the exact round complexity of semi-honest MPC based on weak and necessary assumptions, closing the gap between the 2-party and multi-party case. In the optimal 2-round setting, by instantiating our construction with specific 2-round OT protocols, we obtain 2-round MPC protocols in the plain model from a wide range of number theoretic and algebraic assumptions, including CDH [BM90], factoring [BM90],[5] LWE [PVW08],[6] and constant-noise LPN with a sub-exponential security [GKM+00, YZ16]. This broadens the set of assumptions that round-optimal semi-honest MPC can be based on.

MALICIOUS SECURITY: Going beyond semi-honest security, we further strengthen our protocols to achieve the stronger notion of *semi-malicious security*, as a stepping stone towards *malicious security*. Semi-malicious security proposed by [AJL+12] considers semi-malicious attackers that follow the protocol specification, but may adaptively choose arbitrary inputs and random tapes for computing each of its messages. We enhance our semi-honest protocols to handle such attackers.

Theorem 1.2 (Semi-Malicious Security)**.** *For any $k \geq 2$, there is a k-round* semi-malicious *MPC protocol for any functionality f, from any k-round* semi-malicious *OT protocol.*

Previous semi-malicious protocols have 3 rounds based on LWE [AJL+12, BHP17], or 2 rounds based on bilinear maps [GS17]. We obtain the first 2-round construction from any 2-round semi-malicious OT, which is necessary and can be instantiated from a variety of assumptions, including

[4] We recall that for MPC, we suppose that parties have access to a simultaneous broadcast channel. Furthermore a k-round OT with simultaneous broadcast channel can be transformed into a k-round OT where each round consists a single message or flow either from the receiver to the sender or the other way round. This is because in the last round there is no point for the receiver to send a message to the sender.

[5] This follows from the fact that CDH in the group of quadratic residues is as hard as factoring [Shm85, McC88, BBR97].

[6] The scheme in [PVW08] uses a CRS, but in the semi-honest setting, the sender can generate the CRS and send it to the receiver.

DDH [NP01], QR, and N-th residuosity [HK12]. Furthermore, following the compilation paradigms in recent works [AJL$^+$12, BHP17, ACJ17], we immediately obtain maliciously secure Universal Composable (UC) protocols in the common reference string model [Can01, CLOS02], using non-interactive zero-knowledge (NIZK).

Corollary 1.3 (Malicious Security in the CRS Model)**.** *For any $k \geq 2$, there is a k-round* malicious UC protocol *in the common reference string model for any functionality f, from any k-round* semi-malicious *OT protocol and NIZK.*

Moving forward to malicious MPC protocols in the plain model, we show that, for any $k \geq 5$, k-round malicious MPC protocols can be built from k-round delayed-semi-malicious OT, which is implied by k-round malicious OT.

Theorem 1.4 (Malicious Security in the Plain Model)**.** *For any $k \geq 5$, there is a k-round* malicious *MPC protocol for every functionality f, from any k-round* delayed-semi-malicious *OT protocol.*

This theorem is obtained by first showing that our k-round semi-malicious MPC protocols satisfy a stronger notion of *delayed-semi-malicious* security, when instantiated with a k-round OT protocol satisfying the same notion. Here, delayed-semi-malicious security guards against a stronger variant of semi-malicious attackers, and is still significantly weaker than malicious security. For instance, delayed-semi-malicious OT provides only indistinguishability-based privacy guarantees, whereas malicious OT supports extraction of inputs and simulation. In the second step, we transform our k-round delayed-semi-malicious MPC protocols into k-round malicious MPC protocols, assuming only one-way functions. This transformation relies on specific structures of our protocols. In complement, we also present a generic transformation that starts with *any* $(k-1)$-round delayed semi-malicious MPC protocol.

Previous 5-round malicious protocols rely on LWE and adaptive commitments [BHP17], or DDH [ACJ17]. Our construction weakens the assumptions, and in particular adds factoring-based assumptions into the picture. Our result is *one-step away* from constructing round-optimal malicious MPC from weak and necessary assumptions. So far, 4-round protocols can only be based on subexponential DDH [ACJ17] or subexponential LWE and adaptive commitments [BHP17]. A clear open question is constructing 4-round malicious MPC from 4-round OT.

Garbled Interactive Circuits, and More: Along the way of constructing our MPC protocols, we develop new techniques and primitives that are of independent interest: We propose a new notion of *garbling interactive circuits*, a new primitive of *witness selector* that strengthens witness encryption [GGSW13], and a new notion of *zero-knowledge functional decommitment.* Roughly speaking,

- As mentioned above, garbling interactive machine transforms an interactive machine C talking to a *non-deterministic* oracle $\mathcal{O}(\bar{w})$ using some witnesses, into a garbled interactive machine $\hat{C}$ that upon receiving the witnesses $\bar{w}$ in the clear (*without* any secret encoding) reveals the transcript of the interaction between C and $\mathcal{O}(\bar{w})$ and nothing else, provided that the transcript is *computationally uniquely defined.*

- Witness selector strengthens witness encryption [GGSW13] in the dimension that hiding holds when it is *computationally* hard to find a witness that enables decryption, as opposed to when no such witnesses exist.

- Finally, we enhance standard (computationally binding and computationally hiding) commitment schemes with the capability of partially opening a commitment c to the output $f(v)$ of a function f evaluated on the committed value v, where the commitment and partial decommitment reveal nothing more than the output $f(v)$.

To construct 2-round MPC, we use garbled interactive circuits and functional commitments to collapse rounds of any multi-round MPC protocols down to 2, and implement garbled interactive circuits using witness selector and classical garbled circuits. Our technique generalizes the novel ideas in recent works on constructing laconic OT from DDH [CDG+17], identity based encryption from CDH or factoring [DG17,BLSV17], and 2-round MPC from bilinear pairing [GS17]. These works can be rephrased as implementing special-purpose garbled interactive circuits from standard assumptions, and applying them for their specific applications. In this work, we implement the garbled interactive circuits, witness selector, and functional commitments needed for our constructions of MPC, from OT. The generality of our notions gives a unified view of the techniques in this and prior works.

1.2 Organization

We start with an overview of our techniques in Section 2. Then, after some classical preliminaries in Section 3, we formally define garbled interactive circuit schemes in Section 4. In Section 5, we build 2-round semi-honest MPC protocols from any semi-honest MPC protocols and (zero-knowledge) functional commitment scheme with an associated garbled interactive circuit scheme. In Section 6, we define witness selector schemes and show that they imply garbled interactive circuit schemes. We conclude the construction of 2-round semi-honest MPC protocols from 2-round OT by building a functional commitment scheme with witness selector from any 2-round OT in Section 7. We generalize our constructions to the semi-malicious setting in Section 8 and to the multi-round setting in Section 9. Finally, we show how to construct malicious MPC protocols in Section 10.

2 Overview

Garg et. al. [GGHR14] introduced a generic approach for collapsing any MPC protocol down to 2 rounds, using indistinguishability obfuscation [BGI+01, GGH+13]. Later, Gordon, Liu, and Shi [GLS15] showed how to perform round collapsing using garbled circuits, witness encryption, and NIZK. Very recently, Garg and Srinivasan [GS17] further showed how to do collapse rounds using *garbled protocols*, which can be implemented from bilinear pairing groups. In this work, we perform round collapsing using our new notion of *garbled interactive circuits*; this notion is general and enables us to weaken the assumption to 2-round OT. (See Section 2.8 for a more detailed comparison with prior works.) Below, we give an overview of our construction in the 2-round setting; construction in the multi-round setting is similar.

2.1 Round-Collapsing via Obfuscation

The basic idea is natural and simple: To construct 2-round MPC protocols for a function f, take any multi-round MPC protocols for f, referred to as the *inner MPC protocols*, such as, the Goldreich-Micali-Wigderson protocol [GMW87], and try to eliminate interaction. Garg, Gentry, Halevi, and Raykova (GGHR) [GGHR14] showed how to do this using indistinguishability obfuscation. The idea is to let each player P_i obfuscate their *next-step circuit* $\mathsf{Next}_i(x_i, r_i, \star)$ in an execution of the inner

MPC protocol Π for computing f, where $\mathsf{Next}_i(x_i, r_i, \star)$ has P_i's private input x_i and random tape r_i hardcoded, and produces P_i's next message m_i^ℓ in round ℓ, on input the messages $\bar{m}^{<\ell} = \{m_j^{\ell'}\}_{j,\ell'<\ell}$ broadcast by all parties in the previous rounds,

$$\mathsf{Next}_i(x_i, r_i, \bar{m}^{<\ell}) = m_i^\ell \ . \tag{1}$$

Given all obfuscated circuits $\{\mathrm{i}O(\mathsf{Next}(x_i, r_i, \star)_j)\}$, each party P_i can emulate the execution of Π *in its head*, eliminating interaction completely.

The above idea achieves functionality, but not security. In fact, attackers, given the obfuscated next-step circuits of honest parties, can evaluate the residual function $f(\{x_i\}_{\text{honest } i}, \star)$ with the inputs of honest parties hardcoded, or even evaluate honest parties' next-step circuits on arbitrary "invalid" messages. To avoid this, the protocol requires each party to commit to its input and random tape in the first round, $c_i \xleftarrow{R} \mathsf{COM}(x_i, r_i)$. Then, in the second round, each party obfuscates an *augmented next-step circuit* $\mathsf{AugNext}_i$ that takes additionally a NIZK proof $\pi_j^{\ell'}$ for each message $m_j^{\ell'}$ it receives, and verifies the proof $\pi_j^{\ell'}$ that $m_j^{\ell'}$ is generated honestly from inputs and random tapes committed in c_j (it aborts otherwise). This way, only the *unique* sequence of honestly generated messages is accepted by honest parties' obfuscated circuits. In the security proof, by the security of indistinguishability obfuscation and NIZK, this unique sequence can even be hardcoded into honest parties' obfuscated circuits, enabling simulation using the simulators of the inner MPC protocols.

2.2 Garbled Interactive Circuits

The fact that it suffices and is necessary that the honest parties' obfuscated circuits only allow for a single meaningful "execution path" (determined by the unique sequence of honest messages), suggests that we should rather use garbling instead of obfuscation for hiding honest parties' next-step circuits. However, the challenge is that the next-step circuits Next_i are not plain circuits: They are *interactive* in the sense that they takes inputs (i.e., MPC messages) generated by other parties that cannot be fixed at time of garbling. To overcome the challenge, we formalize the MPC players as interactive circuits, and propose a new notion called *Garbled Interactive Circuits (GIC)*.

Interactive Circuits: The interaction with an interactive circuit is captured via a *non-deterministic* (poly-size) oracle $\mathcal{O}$ that on inputs a *query* q and some *witness* w returns an *answer* $a = \mathcal{O}(q, w)$ (or $\perp$ if w is not accepting). (Note that $\mathcal{O}$ is non-deterministic in the sense that without a valid witness, one cannot evaluate $\mathcal{O}$.) An interactive circuit $\mathrm{i}C$ consists of a list of L next-step circuits $\{\mathrm{i}C^\ell\}_{\ell\in[L]}$. Its execution with oracle $\mathcal{O}$ on input a list of witnesses $\bar{w} = \{\bar{w}^\ell\}$ proceeds in L iterations as depicted in Fig. 1: In round ℓ, $\mathrm{i}C$ on input the state $st^{\ell-1}$ output in the previous round, as well as the answers $\bar{a}^{\ell-1} = \{a_k^{\ell-1}\}$ from $\mathcal{O}$ to queries $\bar{q}^{\ell-1} = \{q_k^{\ell-1}\}$ produced in the previous round, outputs the new state st^ℓ and queries $\bar{q}^\ell = \{q_k^\ell\}$, and a (round) output o^ℓ.

$$\forall \ell, \qquad \mathrm{i}C(st^{\ell-1}, \bar{a}^{\ell-1}) = (st^\ell, \bar{q}^\ell, o^\ell) \ , \text{ where } \forall k, \ a_k^{\ell-1} = \mathcal{O}(q_k^{\ell-1}, w_k^{\ell-1}) \ .$$

The *output* of the execution is the list of round outputs $\bar{o} = \{o^\ell\}_\ell$, and the *transcript* of the execution is the list of all queries, answers, and outputs $\mathsf{trans}(\mathrm{i}C, \bar{w}) = \{(\bar{q}^\ell, \bar{a}^\ell, o^\ell)\}_\ell$. In the case that any oracle answer is $a_k^\ell = \perp$, the execution is considered invalid. For simplicity of this high-level overview, we consider only valid executions and valid transcript; see Section 4 for more details.

Garbled Interactive Circuit Scheme: A Garbled Interactive Circuit (GIC) scheme GiC allows us to garble an interactive circuit $\widehat{\mathrm{i}C} \xleftarrow{R} \mathsf{GiC.Garble}(\mathrm{i}C)$, s.t.

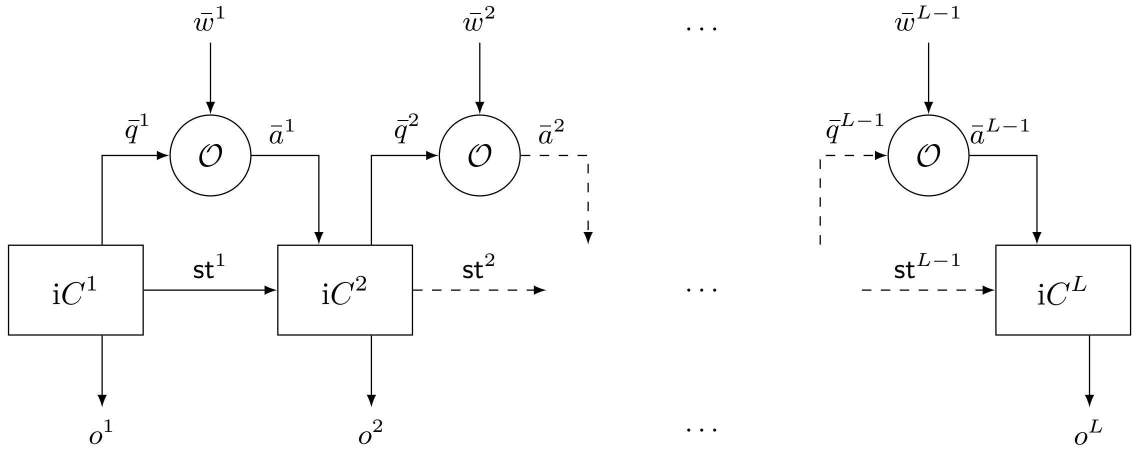


Figure 1: Execution of an interactive circuit $\mathrm{i}C$ with witnesses $\bar{w}$

Correctness: We can evaluate $\widehat{\mathrm{i}C}$ with the oracle $\mathcal{O}$ and a list $\bar{w}$ of witnesses (*in the clear*) to obtain each round output $o^\ell = \mathsf{GiC.Eval}(\mathrm{i}C, \bar{w}^{<\ell})$. This significantly differs from classical garbling techniques where inputs of the computation must be encoded using secrets (such as, mapping them to corresponding input keys or labels).

Simulation Security for Unique Transcripts Distribution: Security guarantees that $\widehat{\mathrm{i}C}$ reveals only the transcript of execution, including all outputs, queries, and answers, and nothing else, that is, it can be simulated by $\widetilde{\mathrm{i}C} \xleftarrow{R} \mathsf{GiC.Sim}(\mathsf{trans})$, provided that there is a *unique* transcript of execution.

The requirement on unique transcript is necessary, otherwise, security is ill-defined as there may exist different transcripts produced by using different witnesses, and the simulator cannot hardcode them all. Furthermore, garbled interactive circuit schemes are meant to be different from obfuscation and hides only a single execution path. To formalize this, there are two options:

- Statistically Unique Transcript. The easier option is requiring simulation security only for interactive circuits $\mathrm{i}C$ that have unique transcript no matter what witnesses are used, that is, for all $\bar{w}, \bar{w}'$, $\mathsf{trans}(\mathrm{i}C, \mathcal{O}, \bar{w}) = \mathsf{trans}(\mathrm{i}C, \mathcal{O}, \bar{w}')$. This is, however, a strong requirement.
- (Default:) Computationally Unique Transcript. The more general option is considering a distribution $\mathrm{i}\mathcal{D}$ over $(\mathrm{i}C, \bar{w})$ that has computationally unique transcripts, in the sense that given $(\mathrm{i}C, \bar{w})$, it is hard to find $\bar{w}'$ that leads to a different valid transcript, $\mathsf{trans}(\mathrm{i}C, \mathcal{O}, \bar{w}) \neq \mathsf{trans}(\mathrm{i}C, \mathcal{O}, \bar{w}')$.[7]

GIC for a computational or statistical unique-transcript distribution ensures:

$$\left\{\mathsf{GiC.Garble}(\mathrm{i}C) \ : \ (\mathrm{i}C, \bar{w}) \xleftarrow{R} \mathrm{i}\mathcal{D}\right\} \approx$$
$$\left\{\mathsf{GiC.Sim}(\mathsf{trans}(\mathrm{i}C, \mathcal{O}, \bar{w})) \ : \ (\mathrm{i}C, \bar{w}) \xleftarrow{R} \mathrm{i}\mathcal{D}\right\}$$

[7] The distribution may output some additional auxiliary information, and it is hard to find witnesses that lead to a different valid transcript even given the auxiliary information. See Section 4 for more details.

Looking ahead, our 2-round MPC protocols from 2-round semi-honest oblivious transfer crucially rely on the stronger notion of GIC for computationally unique transcripts. If using GIC for statistically unique transcripts, we would need a 2-round OT protocol where the receiver's message statistically binds its input bit, which is not a necessary assumption for constructing 2-round semi-honest MPC protocols.

2.3 Constructing GIC from Witness Selector

We start with the warm-up case of building GIC for statistically unique transcripts by combining plain garbled circuits and witness encryption. Witness Encryption (WE) proposed by Garg, Gentry, Sahai, and Waters [GGSW13], enables one to encrypt a message under an instance x of an NP language $\mathcal{L}$ to obtain a ciphertext $\mathsf{ct} \overset{R}{\leftarrow} \mathsf{WE.Enc}(\mathsf{x}, \mathsf{M})$; later this ciphertext can be decrypted using any witness w of x, $\mathsf{M} = \mathsf{WE.Dec}(\mathsf{ct}, \mathsf{w})$. The idea of combining garbled circuits and witness encryption has already appeared in three recent works by Gordon, Liu, and Shi [GLS15], Cho et al. [CDG+17], and Döttling and Garg [DG17]. Our garbled interactive circuit scheme can be viewed as a generalization of their ideas for capturing the full power of this combination. As we explain shortly, to handle computationally unique transcripts, we need to rely on a new primitive called *Witness Selector*, which strengthens WE.[8]

Warm-Up: GIC for Statistically Unique Transcript from WE: To garble an interactive circuit $\mathrm{i}C = \{\mathrm{i}C^\ell\}_\ell$, a natural first attempt is garbling each next-step circuit $\mathrm{i}C^\ell$ as a plain circuit, yielding L garbled circuits $\{\widehat{\mathrm{i}C}^\ell, \mathsf{key}^\ell\}_\ell$, where each input wire of $\widehat{\mathrm{i}C}^\ell$ has two keys, $(\mathsf{key}^\ell[k,0], \mathsf{key}^\ell[k,1])$, one for this input bit being 0 and one for 1. The difficulty is that, to evaluate $\widehat{\mathrm{i}C}^\ell$, the evaluator must obtain keys corresponding to the honestly generated state $st^{\ell-1}$ and answers $\bar{a}^{\ell-1}$ produced in the previous round; denote these keys as $\mathsf{key}^\ell[st^{\ell-1}]$ and $\mathsf{key}^\ell[\bar{a}^{\ell-1}]$.[9] We show how to enable this by modifying the garbled circuits $\{\widehat{\mathrm{i}C}^\ell\}$ as follows.

- The first idea is embedding all keys key^ℓ for one garbled circuit $\widehat{\mathrm{i}C}^\ell$ in the previous one $\widehat{\mathrm{i}C}^{\ell-1}$, so that, $\widehat{\mathrm{i}C}^{\ell-1}$ can output directly the keys $\mathsf{key}^\ell[st^{\ell-1}]$ for the state $st^{\ell-1}$ it produces. This idea, however, does not apply for selecting keys for answers $\bar{a}^{\ell-1}$, as $\widehat{\mathrm{i}C}^{\ell-1}$ only computes queries $\bar{q}^{\ell-1}$ but not answers as it does not necessarily know the corresponding witnesses $\bar{w}^{\ell-1}$.

- The second idea is using WE as a "translator." To illustrate the idea, assume that there is a single query $q^{\ell-1}$ and it has a Boolean answer $a^{\ell-1}$. In this case, let $\widehat{\mathrm{i}C}^{\ell-1}$ output a pair of WE ciphertexts $(\mathsf{ct}_0, \mathsf{ct}_1)$, where ct_b encrypts the key $\mathsf{key}^\ell[k, b]$ for the answer $a^{\ell-1}$ being b, under the statement x_b that the oracle outputs b, $\mathcal{O}(q^{\ell-1}, w'_b) = b$, for some witness w'_b. Now, the evaluator after evaluating $\widehat{\mathrm{i}C}^{\ell-1}$ obtains $\mathsf{ct}_0, \mathsf{ct}_1$. Using the witness w^ℓ it receives as input, it can decrypt the WE ciphertext $\mathsf{ct}^{\ell-1}_{a^{\ell-1}}$ for $a^{\ell-1} = \mathcal{O}(q^{\ell-1}, w^{\ell-1})$, obtaining the right key $\mathsf{key}^\ell[a^{\ell-1}]$ for evaluating the next garbled circuit.

To show security, it boils down to argue that for each garbled circuit $\widehat{\mathrm{i}C}^\ell$, only one key for each input wire is revealed. The security of $\widehat{\mathrm{i}C}^{\ell-1}$ ensures that only keys $\mathsf{key}^\ell[st^{\ell-1}]$ for the right state is revealed. On the other hand, to argue that only keys $\mathsf{key}^\ell[k, a^{\ell-1}]$ for the right answers are revealed,

[8] We mention that the work of Döttling and Garg [DG17] defined what is called *chameleon encryption scheme*, which can be viewed as a special case of our witness selector for a specific language.

[9] This is a slight abuse of notation, where $st^{\ell-1}$ and $\bar{a}^{\ell-1}$ denote both their actual values and the indices of the corresponding input wires.

it crucially relies on the fact that the transcript including the answer is statistically unique. Thus, the ciphertext $\mathsf{ct}_{1-a^{\ell-1}}$ is encrypted under a false statement, and by security of WE, the label $\mathsf{key}^{\ell}[k, 1-a^{\ell-1}]$ is hidden. We emphasize that if the transcript were only computationally unique, both WE ciphertexts $\mathsf{ct}_0, \mathsf{ct}_1$ would potentially be encrypted under true statements — as there may exist two witnesses w_0, w_1 that make the oracle output 0 and 1, $\mathcal{O}(q^{\ell-1}, w_0) = 0$, $\mathcal{O}(q^{\ell-1}, w_1) = 1$, even though it is computationally hard to find them — and the security of WE would be vacuous.

General Case: GIC from Witness Selector: To handle computationally unique transcripts, WE is not the right tool. We propose a new primitive called *Witness Selective* (WS), which strengthens WE in two ways:

Correctness: WS is defined for a non-deterministic oracle $\mathcal{O}$. One can encrypt a set of keys $\mathsf{key} = \{\mathsf{key}[k,b]\}_{k\in[l],b\in\{0,1\}}$ under a query q, $\mathsf{ct} \leftarrow \mathsf{WS.Enc}(q, \mathsf{key})$, which can later be decrypted using a witness w revealing the keys selected according to the output $a = \mathcal{O}(q, w)$, that is, $\{\mathsf{key}[k, a_k]\}_k = \mathsf{WS.Dec}(\mathsf{ct}, w)$.

Semantic Security for Unique Answers: The security guarantee is that the WS ciphertext ct hides all the keys $\mathsf{key}[k, 1-a_k]$, provided that a is the *computationally unique answer*. Clearly, if it were easy to find two witnesses w, w' such that, $(a = \mathcal{O}(q,w)) \neq (a' = \mathcal{O}(q,w'))$, the aforementioned semantic security cannot hold. Therefore, similarly to GIC, security is only required to hold for a distribution $\mathrm{w}\mathcal{D}$ over (q, w) that has computationally unique answers in the sense that given (q, w), it is hard to find w' that makes $\mathcal{O}$ output a different valid answer. Then,

$$\Big\{\mathsf{WS.Enc}(q, \mathsf{key}) \;:\; (q,w) \xleftarrow{R} \mathrm{w}\mathcal{D}\Big\} \approx$$
$$\Big\{\mathsf{WS.Enc}(q, \mathsf{key}) \;:\; (q,w) \xleftarrow{R} \mathrm{w}\mathcal{D};\; a = \mathcal{O}(q,w);\; \forall k,\; \mathsf{key}[k, 1-a_k] = 0\Big\}\ .$$

We can construct general GIC scheme for computationally unique transcript by replacing WE in the warm-up construction with WS. Slightly more precisely, each garbled circuit $\widehat{\mathrm{i}C}^{\ell-1}$ outputs a WS ciphertext ct encrypting keys $\{\mathsf{key}[k,b]\}$ for all wires corresponding to the oracle answer $a^{\ell-1}$, under the query $q^{\ell-1}$ (if there are multiple queries, simply generate one WS ciphertext for each query); then, the evaluator can use the witness $w^{\ell-1}$ to decrypt and obtain keys $\{\mathsf{key}[k, a_k^{\ell-1}]\}$ selected according to the oracle answer $a^{\ell-1} = \mathcal{O}(q^{\ell-1}, w^{\ell-1})$. Since the oracle answer (as a part of the transcript) is computationally unique, semantic security of WS ensures that the other keys $\{\mathsf{key}[k, 1-a_k^{\ell-1}]\}$ remain hidden, and hence we can invoke the security of the garbled circuits to argue the security of GIC.

Relation between WS, WE, and Extractable WE: As discussed above, WS is stronger than WE. For instance, one can use WS to encrypt a set of keys key under a query $q = (h, y = h(v))$ for a randomly sampled collision-resistant hash function h. With respect to the de-hashing oracle $\mathcal{O}(q, v')$ that outputs v' if $y = h(v')$, a WS ciphertext reveals only keys $\{\mathsf{key}[k, v_k]\}$ selected by v, and hides others. In contrast, WE provides no security in this case. On the other hand, WS is weaker than the notion of extractable WE [GKP+13]. Roughly speaking, extractable WE guarantees that for every attacker A, there is an extractor E, such that, if A can decrypt a ciphertext encrypted under statement x, then E can output a witness of x. Extractable WE implies WS, and is strictly stronger as it requires knowledge extraction.

We note that so far there is no construction of general-purpose WE, let alone WS or extractable WE, from standard assumptions. This is also not the goal of this work. Instead, we show below how to construct special-purpose WS that suffices to construct 2-round MPC protocols.

2.4 Round-Collapsing via Garbled Interactive Circuits

We now revisit the round-collapsing approach, by replacing obfuscation with garbled interactive circuits. First, we observe that each player P_i in the inner MPC protocol can be viewed as an interactive circuit $\{P_i^\ell\}$, interacting with an oracle $\mathcal{O}$ representing the other parties $\{P_j\}$, as described in Fig. 2.

P_i as an interactive circuit $\{P_i^\ell\}$

- The non-deterministic oracle $\mathcal{O}$ (representing all other parties) receives queries of form $q_j^\ell = (c_j, G_j^\ell)$, consisting of P_j's commitment and its next-step circuit with all messages in the first $\ell - 1$ rounds hardcoded, $G_j^\ell(\star, \star) = \mathsf{Next}_j(\star, \star, \bar{m}^{<\ell})$. On input such a query and a witness $w_j^\ell = (m_j^\ell, \pi_j^\ell)$, $\mathcal{O}$ computes:

$$a_j^\ell = \mathcal{O}(q_j^\ell, (m_j^\ell, \pi_j^\ell)) = \begin{cases} m_j^\ell & \text{if } \pi_j^\ell \text{ proves that the values } (x_j, r_j) \\ & \text{committed in } c_j \text{ satisfy } m_j^\ell = G_j^\ell(x_j, r_j) \\ \perp & \text{otherwise} \end{cases} .$$

- P_i^ℓ proceeds similarly as Next_i in Eq. (1) (page 8), except that, it additionally outputs one query $q_j^\ell = (c_j, G_j^\ell)$ for each player P_j's message m_j^ℓ, and a proof π_i^ℓ that its next message is indeed m_i^ℓ. (The proof system is described later.)

$$P_i^\ell(x_i, r_i, \overbrace{\bar{m}^{<\ell-1}}^{st^{\ell-1}}, \overbrace{\{m_j^{\ell-1}\}_j}^{\bar{a}^{\ell-1}}) = (\overbrace{\bar{m}^{<\ell}}^{st^{\ell}}, \overbrace{\{q_j^\ell\}_j}^{\bar{q}^{\ell}}, \overbrace{(m_i^\ell, \pi_i^\ell)}^{o^{\ell}}) ,$$

Figure 2: Each player P_i can be formalized as an interactive circuit $P_i = \{P_i^\ell\}$.

The important details are: In each round ℓ, P_i^ℓ obtains through the oracle $\mathcal{O}$ all messages $\bar{m}^{\ell-1} = \{m_j^{\ell-1}\}_j$ output in the previous round, and additionally, it outputs a proof π_i^ℓ that the message m_i^ℓ it outputs is generated honestly from its input x_i and random tape r_i committed in c_i. The message and proof are exactly the witness $w_i^\ell = (m_i^\ell, \pi_i^\ell)$ for the query q_i^ℓ that players P_j^ℓ make in round ℓ to the oracle $\mathcal{O}$ for obtaining P_i's message $a_i^\ell = m_i^\ell$ for the next round.

Our 2-Round MPC Protocol: Therefore, we can use a GIC scheme to garble the interactive circuit representing each player P_i to collapse round:

1. In the first round of MPC, each P_i broadcasts a commitment c_i to its input x_i and random tape r_i, and
2. in the second round, each P_i sends the garbled interactive circuit $\widehat{P}_i \xleftarrow{R} \mathsf{GiC.Garble}(\{P_i^\ell\})$, and
3. each P_i emulates the execution of inner MPC in its head, by evaluating all $\{\widehat{P}_j\}$ round by round: In round ℓ, it evaluates $o_j^\ell = (m_j^\ell, \pi_j^\ell) = \mathsf{GiC.Eval}(\widehat{P}_j, \bar{w}^{<\ell})$, using the outputs obtained

in previous rounds as witnesses, $w^{<\ell} = o^{<\ell} = \{(m_k^{\ell'}, \pi_k^{\ell'})\}_{k,\ell'<\ell}$. P_i obtains its output when the inner MPC execution completes.

We observe that the transcript of execution of each $\{P_i^\ell\}$ is indeed *computationally unique*, as the commitments $\{c_j\}$ have unique committed values $\{x_j, r_j\}$ by the computational binding property, and lead to unique next messages $\{m_j^\ell\}$, by the soundness of proofs $\{\pi_j^\ell\}$. Therefore, the GIC scheme guarantees that the garbled interactive circuits reveals only their outputs, queries, and answers, summing up to all commitments $\{c_j\}$, inner MPC messages $\{m_j^\ell\}$, and proofs $\{\pi_j^\ell\}$, all of which can be made simulatable.

First Attempt of Instantiation: The MPC messages can be simulated by the simulator of the inner MPC protocol. To make commitments and proofs simulatable, the easiest way is using a standard non-interactive commitment scheme and a NIZK system, which however 1) requires a common reference string, and 2) makes the task of instantiating the associated WS scheme difficult. Recall that to instantiate the GIC scheme, we need a WS scheme for the oracle $\mathcal{O}$ described above, which internally verifies proofs. To solve this, we resort to a *zero-knowledge* Functional Commitment (FC) scheme that has a built-in special-purpose proof system. By minimizing the security requirements on this commitment, we manage to construct it, together with an associated WS scheme, from 2-message semi-honest OT (which is a necessary assumption). This gives 2-round MPC protocols in the plain model from 2-message semi-honest OT.

2.5 Functional Commitment with Witness Selector from OT

A zero-knowledge functional commitment scheme FC is computationally binding and computationally hiding, and additionally supports functional opening that is both *binding* and *zero-knowledge*. The notion of functional commitment was previously proposed by Libert, Ramanna, and Yung [LRY16] for inner product functions, and later generalized to general functions in [BGJS16]. Here, we consider a stronger property, namely a *zero-knowledge* property. On the other hand, we do not require commitments nor functional decommitments to be of size constant in the length of the committed value, and our binding property only holds against semi-honest adversaries. Functional commitments were also implicitly and informally suggested by Gorbunov, Vaikuntanathan, and Wichs in [GVW15], as a way to interpret their new primitive: Homomorphic Trapdoor Functions (HTDFs). HTDFs could be used to construct our functional commitments (but the converse is not true). However, we do not know how to construct WS associated to an FC built from the HTDF proposed in [GVW15].

Functional Opening: For a commitment $c = \mathsf{FC.Com}(v;\rho)$ and a circuit G, one can generate a *functional decommitment* d to the output of G evaluated on the committed value v, namely $m = G(v)$, using the randomness ρ of the commitment c,

$$d = \mathsf{FC.FOpen}(c, G, m, \rho), \quad \mathsf{FC.FVer}(c, G, m, d) = 1 \ .$$

We say that (m, d) is a decommitment to (c, G); here, d serves as a proof $\pi = d$ that the value committed in c evaluates to m through G in our 2-round MPC protocols.

(Semi-Honest) Functional Binding: For an honestly generated commitment $c = \mathsf{FC.Com}(v;\rho)$ with random tape ρ, it is hard to find a decommitment (m', d') to (c, G) for a different output $m' \neq m$, even given ρ. Note this is weaker than standard computational binding, as binding

is only required for honestly generated commitments. This corresponds to *distributional soundness* of the proofs.

Simulation (i.e., Zero-Knowledge): An honestly generated commitment $c \xleftarrow{R} \mathsf{FC.Com}(v;\rho)$ (with random tape ρ) and decommitment d can be simulated *together*, using only the output m, $(\tilde{c},\tilde{d}) \xleftarrow{R} \mathsf{FC.Sim}(c,G,m)$. This property is weaker than standard zero-knowledge, as the statement is from a distribution and is also simulated; only a single decommitment d can be given for each commitment, or else simulation does not work.

A WS scheme associated with FC is for the oracle $\mathcal{O}^{\mathsf{FC}}$ that on input a query (c,G) and a witness $w=(m,d)$, outputs m if (m,d) is a valid decommitment to (c,G), and $\perp$ otherwise. The functional binding property ensures that for any v,G, the distribution $\mathrm{w}\mathcal{D}_{v,G}$ of query $q=(c,G)$ and decommitment $w=(m,d)$ for honestly generated $c=\mathsf{FC.Com}(v;\rho)$, produces computationally unique oracle answer m (even given the randomness ρ as auxiliary information). Despite the fact that functional commitments are only *semi-honestly* binding and *one-time* simulatable, we show that, together with an associated WS scheme, they suffice to instantiate our 2-round MPC protocols.

FC from Garbled Circuits and OT: We show how to construct a functional commitment, and its associated WS scheme, from garbled circuits and a 2-round string 2-to-1 semi-honest OT.

OT as semi-honest binding commitment: We start with observing that any string 2-to-1 semi-honest OT gives a commitment scheme that is *semi-honest binding*; that is, given an honestly generated commitment $c=\mathsf{Com}(v;\rho)$ using a uniformly random tape ρ, it is hard to find a decommitment (v',ρ') that opens c to a different value $v'\neq v$ even given ρ. To see this, consider the *parallelized* version of 2-to-1 string OT, where $\mathrm{ot}_1=\mathrm{p}\mathsf{OT}_1(x;\rho)$ generates the first flows from OT receiver for every bit x_k, and $\mathrm{ot}_2=\mathrm{p}\mathsf{OT}_2(\mathrm{ot}_1,\{\mathsf{key}[k,b]\})$ generates the second flows from OT sender for every pair of inputs $(\mathsf{key}[k,0],\mathsf{key}[k,1])$. Combining ot_2 with the randomness ρ used for generating the first flows, one can act as the OT receiver to recover exactly one input $\mathsf{key}[k,x_k]$ at each coordinate k. We argue that the first flows $\mathrm{ot}_1=\mathrm{p}\mathsf{OT}_1(x;\rho)$ is a semi-honest commitment to x. Suppose that it is not the case and that it is easy to find a decommitment ρ' to a different value $x'\neq x$. Then a semi-honest attacker acting as OT receiver can violate the privacy of OT sender. (However, observe that $\mathrm{p}\mathsf{OT}_1(x)$ is not necessarily computationally binding, as there is no security for maliciously generated first flows of OT.)

Functional Opening: We use garbled circuits and OT (as a semi-honest binding commitment scheme) to enable functional opening. To commit to a value v, garble a universal circuit $U_v(\star)=U(v,\star)$ with v hardcoded, and commit to all its input keys $\{\mathsf{key}[k,b]\}$ using $\mathrm{p}\mathsf{OT}_1$:

$$\mathsf{FC.Com}(v;\rho)=c=(\widehat{U}_v,\mathrm{ot}_1)\ ,\ \text{where}\ \ \mathrm{ot}_1[k,b]=\mathrm{p}\mathsf{OT}_1(\mathsf{key}[k,b];\ \rho[k,b])\ .$$

To generate a decommitment (m,d) of (c,G), simply send the keys and randomness used for generating the OT first flows $\{\mathrm{ot}_1[k,G[k]]\}$ selected by G (more formally, $G[k]$ is the k-th bit of the description of G which is used as input to U_v):

$$\mathsf{FC.FOpen}(c,G,m,\rho)\ =\ d=\{\mathsf{key}[k,G[k]],\ \rho[k,G[k]]\}\ .$$

Verifying a decommitment $d=\{\mathsf{key}',\rho'\}$ w.r.t. (c,G,m) involves checking that the keys and randomness contained in d' generate the OT first flows selected by G, and the garbled universal circuit $\widehat{U}_v$ evaluates to m on input these keys.

$$\begin{aligned}\mathsf{FC.FVer}(c,G,m,d)=1 \qquad \text{iff} \qquad & 1)\ \forall k,\ \mathrm{ot}_1[k,G[k]]=\mathrm{p}\mathsf{OT}_1(\mathsf{key}'[k];\rho'[k])\ \text{and}\\ & 2)\ \widehat{U}_v(\mathsf{key}')=m\ .\end{aligned}$$

It is easy to see that the semi-honest binding property of pOT_1 implies the semi-honest functional binding of FC, and that a pair (c, d) can be simulated relying on the security of garbled circuits and the computational hiding property (i.e., receiver privacy) of pOT_1.

WS for FC*:* Next, to construct a WS scheme for the oracle $\mathcal{O}^{\mathsf{FC}}$ that verifies the functional decommitment of FC, we again use garbled circuits to "enforce and hide" this verification. To encrypt a set of messages $\mathsf{M}[i, b']$ under a query (c, G), our idea is to garble the following circuit V that acts as $\mathsf{FC.FVer}$ (without checking 1)), and selects messages according to the output m if verification passes,

$$V(\{\mathsf{key}'[k]\}) = \begin{cases} \{\mathsf{M}[i, m_i]\} & \text{if } \widehat{U}_v(\{\mathsf{key}'[k]\}) = m \\ \perp & \text{otherwise} \end{cases} . \tag{2}$$

Let $\widehat{V}$ be the garbled circuit, and $\{\mathsf{okey}_k[j, \beta]\}_j$ the set of keys for the input wires corresponding to $\mathsf{key}'[k]$. (For clarity, we denote keys for $\widehat{V}$ as okey.)

Given a decommitment $d = (\mathsf{key}', \rho')$, correct WS decryption should recover messages $\{\mathsf{M}[i, G(v)_i]\}$ selected according to the correct output $G(v)$ if the decommitment is valid, and $\perp$ if invalid. To enable this, what is missing is a "translation mechanism" that can achieve the following: For every k,

- Correctness: if $(\mathsf{key}'[k], \rho'[k])$ is a valid decommitment to $\mathrm{ot}_1[k, G[k]]$, it translates this pair into input keys of $\widehat{V}$ corresponding to $\mathsf{key}[k, G[k]]$, namely $\{\mathsf{okey}_k[j, \mathsf{key}[k, G[k]]_j]\}_j$.
- Security: the other keys $\{\mathsf{okey}_k[j, 1 - \mathsf{key}[k, G[k]]_j]\}_j$ are always hidden.

With such a translation mechanism, given a valid decommitment $d = \{\mathsf{key}[k, G[k]], \rho[k, G[k]]\}$, one can obtain all input keys corresponding to $\{\mathsf{key}[k, G[k]]\}$, and can evaluate $\widehat{V}$ with these keys to obtain the correct output,

$$\widehat{V}\left(\left\{\{\mathsf{okey}_k[j, \mathsf{key}[k, G[k]]_j]\}_j\right\}_k\right) = V(\{\mathsf{key}[k, G[k]]\}_k) = \{\mathsf{M}[i, G(v)_i]\}_i . \tag{3}$$

The security of the translation mechanism and garbled circuit $\widehat{V}$ guarantees that only the right messages $\{\mathsf{M}[i, G(v)_i]\}$ are revealed.

Our key observation is that the second flows of OT is exactly such a translation mechanism. For every OT first flows $\mathrm{ot}_1[k, G[k]]$ selected by G, generate the OT second flows using appropriate input keys of $\widehat{V}$ as sender's inputs,

$$\forall k, \qquad \mathrm{ot}_2[k] \overset{R}{\leftarrow} \mathsf{pOT}_2(\mathrm{ot}_1[k, G[k]],\ \{\mathsf{okey}_k[j, \beta]\}_{j,\beta}) . \tag{4}$$

Indeed, for every k, given a valid decommitment $(\mathsf{key}[k, G[k]], \rho')$ to $\mathrm{ot}_1[k, G[k]]$, one can act as an OT receiver to recover input keys $\{\mathsf{okey}_k\,[j, \mathsf{key}[k, G[k]]_j]\}_j$, achieving correct translation. On the other hand, the OT sender's security guarantees that the other keys $\{\mathsf{okey}_k\,[j, 1 - \mathsf{key}[k, G[k]]_j]\}_j$ remain hidden.

Summarizing the above ideas gives the following construction of WS for FC:

- $\mathsf{WS.Enc}((c, G), \mathsf{M})$: To encrypt M under (c, G), encryptor garbles the circuit V as in Equation (2), and generates the second OT flows as in Equation (4). The WS ciphertext is $\mathsf{ct} = (c, G, \widehat{V}, \{\mathrm{ot}_2[k]\})$.

- $\mathsf{WS.Dec}(\mathsf{ct}, d)$: To decrypt ct with a decommitment $d = \{\mathsf{key}', \rho'\}$, the decryptor first verifies that for every k $(\mathsf{key}'[k], \rho'[k])$ is a valid decommitment of $\mathrm{ot}_1[k, G[k]]$ in c; otherwise, abort. Then, for every k, it acts as an OT receiver with input $\mathsf{key}'[k]$, randomness $\rho'[k]$, and OT sender's message $\mathrm{ot}_2[k]$ to recover input keys of $\widehat{V}$ corresponding to $\mathsf{key}'[k]$. Finally, it evaluates $\widehat{V}$ with the obtained keys and output the messages output by $\widehat{V}$, as in Equation (3).

The correctness and security of the WS scheme follows directly from the correctness and security of the translation mechanism, which are in turn implied by those of OT. See Section 7 for more details.

Combining Sections 2.1 to 2.5, we get a construction of a 2-round semi-honest MPC protocol from any 2-round semi-honest OT protocol using round collapsing for an inner MPC protocol.

2.6 Extension to Semi-Malicious and Malicious Security in the CRS model

Toward achieving malicious security, we first achieve semi-malicious security. Roughly speaking, a semi-malicious party P_j generates its messages according to the protocol using arbitrarily and adaptively chosen inputs and random tapes. This is formalized by letting P_j "explain" each message m_j^ℓ it sends with a pair of input and random tape consistent with it, on a special witness tape. In the two-round setting, the challenge in simulating the view of P_j lies in simulating honest parties' first messages without knowing any secret information of P_j. This is because P_j may *rush* to see honest parties' first messages before outputting its own message, input, and random tape. (Observe that this is not an issue for semi-honest security, as the simulator learns the inputs and random tapes of corrupted parties first.)

Recall that in our 2-round protocols, each party P_i sends functional commitments c_i to its input and random tape (x_i, r_i) in the first round, which are later partially decommitted to reveal P_i's messages m in the inner MPC protocol. The simulation property of the functional commitment scheme FC ensures that the commitment and decommitment can be simulated together using just the message. However, this is insufficient for achieving semi-malicious security, as the simulator must simulate commitments in the first round with no information. To overcome this problem, we strengthen the simulatability of FC to *equivocability*, that is, simulation takes the following two steps: First, a commitment $\tilde{c}$ is simulated with no information, and later it is equivocated to open to any output m w.r.t. any circuit G. Instantiating our 2-round MPC protocols with such an *equivocal functional commitment scheme*, and other primitives that are semi-maliciously secure (e.g., using a semi-maliciously secure multi-round MPC protocol, and 2-round OT protocol), naturally "lift" semi-honest security to semi-malicious security.

With a simple idea, we can transform any *simulatable* functional commitment scheme FC into an equivocal one eFC: Let $(\mathsf{OT}_1, \mathsf{OT}_2)$ be the sender and receiver's algorithms of a 2-out-of-1 OT scheme.

- To commit to v, generate a FC commitment c to v, and then commit to each bit c_i twice using the algorithm OT_1, yielding the eFC commitment:

$$ec = \{\mathrm{ot}_1[i,0] = \mathsf{OT}_1(c_i;\ r[i,0]),\ \mathrm{ot}_1[i,1] = \mathsf{OT}_1(c_i;\ r[i,1])\}_i\ .$$

- An eFC decommitment $(ed, G(v))$ to (ec, G) contains the FC decommitment $(d, G(v))$ to (c, G), and the OT randomness $\{r[i, c_i]\}$ for generating the set of first flows $\{\mathrm{ot}_1[i, c_i]\}$ selected by c. Note that for any ec generated according to the above commitment algorithm, the revealed

OT randomness determines the commitment c, and then the FC decommitment d determines the output $G(v)$.

- Now, a commitment can be simulated by committing to both 0 and 1 in ec,

$$\widetilde{ec} = \{\mathrm{ot}_1[i,0] = \mathsf{OT}_1(0;\ r[i,0]),\ \mathrm{ot}_1[i,1] = \mathsf{OT}_1(1;\ r[i,1])\}_i\ .$$

  To decommit $\widetilde{ec}$ to output $G(v)$, first simulate the FC commitment and decommitment $(\tilde{c}, \tilde{d})$ together using $G(v)$, and then reveal the set of randomness $\{r[i, \tilde{c}_i]\}$ selected according to the simulated commitment $\tilde{c}$.

The WS scheme associated with eFC can be constructed similarly as that for FC. The above idea is conceptually simple, but leads to nested calls of $\mathrm{p}\mathsf{OT}_1$ / OT_1, as a FC commitment c already contains OT first flows. This is not a problem when using 2-round OT protocols, but does not extend to using multi-round OT. In Section 8, we present a more involved construction that avoids nested calls.

Malicious Security in the CRS Model. Given 2-round semi-maliciously secure protocols, in the CRS model, we can let each party prove using NIZK that each message is generated in a semi-malicious way (i.e., according to the protocol w.r.t. some input and random tape) as done in [AJL+12], which immediately gives Corollary 1.3 in the introduction. We refer the reader to [AJL+12] for more details.

Extension to k Rounds. Our 2-round semi-honest or semi-malicious constructions so far can be extended to k-round constructions, when replacing the underlying 2-round OT protocols with semi-honest or semi-malicious k-round OT protocols. See Section 9 for more details.

2.7 Malicious Security in the Plain Model

From General $(k-1)$-Round Delayed-Semi-Malicious MPC: We first show a new compilation that turns *any* $(k-1)$-round MPC protocol for computing f satisfying a stronger variant of semi-malicious security, called *delayed-semi-malicious security*, into a k-round malicious MPC protocol for f, assuming only one-way functions, for any $k \geq 5$. Roughly speaking, a delayed-semi-malicious party P_j acts like a semi-malicious party, except that, it only "explains" *all* its messages *once*, *before the last round* (instead of explaining each of its messages after each round). This is formalized by letting P_j output a pair of input and random tape before the last round (on its special witness tape) which is required to be consistent with all P_j's messages. We say that a protocol is *delayed-semi-malicious secure* if it is secure against such adversaries. (For technical reasons, we require the protocols to have a *universal* simulator.) We observe that our k-round semi-malicious MPC protocols, when instantiated with a k-round delayed-semi-malicious OT become secure against delayed semi-malicious attackers (and admit a universal simulator).

To "lift" delayed-semi-malicious security to malicious security *generically*, our compilation builds on techniques of [ACJ17]. To illustrate the idea, consider compiling our 2-round delayed-semi-malicious MPC protocol Φ for f into a 5-round malicious MPC protocol Π for f. The basic idea is running Φ for computing f, and restricting a malicious adversary A to act as a delayed-semi-malicious one A' by requiring A to prove using zero-knowledge proof of knowledge (ZKPOK) that its messages in each round of Φ are generated correctly according to some input and random tape.

Unlike the CRS model, ZKPOK in the plain model requires at least 4 rounds. Sequentializing the two ZKPOK leads to a *8-round* protocol. But if the ZKPOK allows for *delayed-input*, that is, only the last prover's message depends on the statement and witness, then the two ZKPOK can be partially parallelized, leading to a *5-round* protocol. In addition, in order to prevent mauling attacks, the ZKPOK must be *non-malleable*. Fortunately, Ciampi, Ostrovsky, Siniscalchi, and Visconti [COSV17] (COSV) recently constructed a 4-round delayed-input non-malleable ZKPOK protocol from one-way functions, which suffice for our purpose. When starting from a 4-round (instead of 2-round) protocol Φ, to obtain a 5-round malicious protocol Π, we cannot afford to prove correctness of each round. But, if Φ is delayed-semi-malicious secure, then it suffices to prove correctness only at the last two rounds, keeping the round complexity at 5.

Though the high-level ideas are simple, there are subtleties in the construction and proof. We cannot use the non-malleable ZKPOK in a black-box. This is because simulation of non-malleable ZKPOK uses rewindings and may render the Φ instance running in parallel insecure. In addition, the COSV non-malleable ZKPOK is only many-many non-malleable in the *synchronous* setting, but in Π, the non-malleable ZKPOKs are not completely synchronized (ending either at the second last or the last round). Therefore, we use the COSV construction in a *non-black-box* way in Π (with some simplification) as done in [ACJ17]. The specific property of COSV non-malleable ZKPOK that we rely on is that simulation requires only rewinding the second and third rounds, while (witness) extraction requires only rewinding the third and forth rounds. This means Φ would be rewound at second/third and third/fourth rounds. The security of a generic delayed-semi-malicious protocol may not hold amid such rewinding. However, if we start with a *4-round* protocol, rewindings can be circumvented if Π contains no messages of Φ in its third round. This means, in the rewindings of second/third and third/fourth rounds, the simulator can simply *replay* messages of Φ in the main thread, keeping the instance of Φ secure. See Section 10.3 for more details.

From Our Specific k-Round Delayed-Semi-Malicious MPC: The above transformation is modular and general, but comes at a price — it only gives k-round malicious MPC from $(k-1)$-round delayed-semi-malicious OT, which is not necessary. To eliminate the gap, we leverage specific structures of our k-round delayed-semi-malicious protocols, to address the rewinding issue above. To illustrate the ideas, lets again examine the $k = 5$ case.

- To handle rewindings at third/fourth rounds, we observe that at the end of fourth round, each party P_i proves using COSV non-malleable ZK that it has acted honestly in Φ according to some input and random tape (x_i, r_i). If in the malicious protocol Π, each party additionally commits to (x_i, r_i) in the first two rounds using a statistically binding commitment scheme (and prove that its messages are generated honestly using the committed value). Then, as long as the adversary cannot cheat in the non-malleable ZK proofs, its messages in the third/fourth rounds of Φ are determined by the commitments in the first two rounds. Therefore, the simulator can afford to continuously rewinding the adversary, until it *repeats* its messages in Φ in the main execution thread. In this case, the simulator can simply *replay* the honest parties' messages in Φ in the main thread.

- To handle rewindings at second/third rounds, the specific property of our protocol that we rely on is that the first 2 rounds of Φ contains only instances of OT, whose messages do not depend on parties' inputs. The latter holds because of the random self-reducibility of OT (hence, the sender and receiver can only use their inputs for generating their last messages). To avoid rewinding these OT instances in Φ, our idea is modifying the malicious protocol Π

as follows: In the first 2 rounds, for every OT instance OT_j in Φ, Π runs two independent OT instances OT_j^0 and OT_j^1. In the third round, an *random* instance $\mathsf{OT}_j^{b_j}$ for $b_j \leftarrow \{0,1\}$ is chosen to be continued, and the other $\mathsf{OT}_j^{1-b_j}$ aborted — they are referred to as the *real* and *shadow* instances. Now in a rewinding of the second/third round, to avoid rewinding the real OT instances, the simulator *replays* the OT messages in the second round, and in the third round, continues the shadow instances $\mathsf{OT}_j^{1-b_j}$ and aborts the real instances $\mathsf{OT}_j^{b_j}$. Importantly, since for every pair $(\mathsf{OT}_j^0, \mathsf{OT}_j^1)$, the choice b_j of which is real and which is shadow is random and independent, the view of the adversary in a rewinding is identical to that in the main execution thread. This guarantees that rewindings would succeed.

We remark that this idea does not apply in general. This is because to continue a random instance of a general protocol Φ in the third round, parties may need to *agree* on that instance, which requires coin-tossing. In contrast, our protocol Φ consists of many OT instances OT_j, the decision of which of $(\mathsf{OT}_j^0, \mathsf{OT}_j^1)$ to continue can be made *locally* by the party who is supposed to send the third message of OT_j in Φ.

In Section 10.4, we put the above two ideas together, which gives k-round malicious OT from k-round delayed-semi-malicious OT.

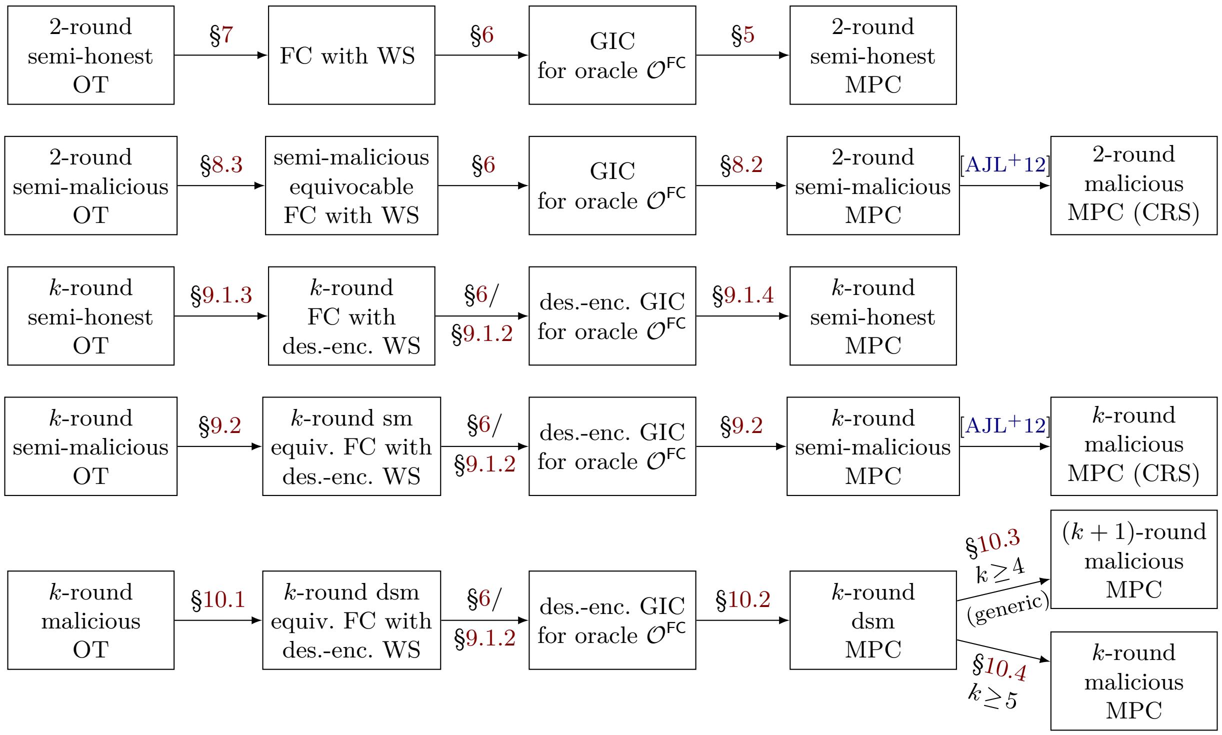
Summary of results is provided in Fig. 3.

2.8 Related Works

We compare with prior related works [GLS15, GS17, CDG+17, DG17], and briefly discuss how they can be rephrased as applying garbled interactive circuits. Discussion below is at a very high level.

- Gordon, Liu, and Shi [GLS15] constructed 2-round MPC protocols from general purpose witness encryption and NIZK. Their technique can be viewed as using garbled circuits and general-purpose witness encryption to implement garbled interactive circuits with statistically binding transcripts. As discussed above, to use the latter to collapse rounds of multi-round MPC protocols, the MPC messages must be statistically unique, and the witness encryption must be powerful enough to verify the correctness of messages in order to "translate" correct messages into appropriate keys. The former is achieved by letting players commit to their secret input and randomness using a statistically binding commitment, while the latter is achieved using general purpose witness encryption to verify NIZK proofs of the correctness of messages. As a result their assumptions are significantly stronger than ours.

- Garg and Srinivasan [GS17] showed how to instantiate the approach of Gordon, Liu, and Shi [GLS15] with a special-purpose witness encryption for a specific language, which can be implemented using bilinear pairing groups. Instead of using generic statistically binding commitments and NIZK, they employed homomorphic proof commitments proposed by Groth, Ostrovsky, and Sahai [GOS06] for constructing NIZK proofs based on bilinear maps. Such commitments are additively homomorphic, and admits zero-knowledge proofs for statements that c_1, c_2, c_3 commits to bits x_1, x_2, x_3 s.t. $x_3 = x_1$ NAND x_2. By implementing a witness encryption scheme for a specific language w.r.t. homomorphic proof commitments, they enable constructing garbled interactive circuits that can verify the correctness of a message by verifying its computation "NAND by NAND." The homomorphic proof commitments heavily


2-round semi-honest OT
§7
FC with WS
§6
GIC for oracle $\mathcal{O}^{\mathsf{FC}}$
§5
2-round semi-honest MPC
2-round semi-malicious OT
§8.3
semi-malicious equivocable FC with WS
§6
GIC for oracle $\mathcal{O}^{\mathsf{FC}}$
§8.2
2-round semi-malicious MPC
[AJL+12]
2-round malicious MPC (CRS)
k-round semi-honest OT
§9.1.3
k-round FC with des.-enc. WS
§6/ §9.1.2
des.-enc. GIC for oracle $\mathcal{O}^{\mathsf{FC}}$
§9.1.4
k-round semi-honest MPC
k-round semi-malicious OT
§9.2
k-round sm equiv. FC with des.-enc. WS
§6/ §9.1.2
des.-enc. GIC for oracle $\mathcal{O}^{\mathsf{FC}}$
§9.2
k-round semi-malicious MPC
[AJL+12]
k-round malicious MPC (CRS)
k-round malicious OT
§10.1
k-round dsm equiv. FC with des.-enc. WS
§6/ §9.1.2
des.-enc. GIC for oracle $\mathcal{O}^{\mathsf{FC}}$
§10.2
k-round dsm MPC
§10.3 $k \geq 4$ (generic)
$(k+1)$-round malicious MPC
§10.4 $k \geq 5$
k-round malicious MPC

- "des.-enc. WS/GIC" stands for "designated-encryptor WS/GIC", which is an extension of WS/GIC where the encryptor/garbler is allowed to have some extra information.
- "sm equiv. FC" stands for "semi-malicious equivocable FC".
- "dsm" stands for "delayed-semi-malicious".
- "dsm equiv. FC" stands for "delayed-semi-malicious equivocable FC".

Figure 3: Summary of results

exploits the algebraic structure of bilinear maps, whereas in this work we rely on the weaker general assumption of OT.

Garg and Srinivasan defined the beautiful notion of garbled protocols, which generalize the classical garbled circuits to the setting of protocols. Roughly speaking, each party in a protocol Π can independently and locally garble their inputs and next step functions, so that, when all input encodings, garbled next step functions and their appropriate input labels are collected, one can obtain the transcript of the protocol Π when executed with the plain inputs, and nothing else. The abstraction of garbled protocol is at a level higher than our garbled interactive circuits, which focuses on a single interactive circuits that may or may not a part of protocol. In particular, the latter can be used to implement the former. Working at a lower abstraction level has the benefit of wider applicability, as we illustrate below, garbled interactive circuits can be used to "explain" the ideas in recent constructions of laconic OT and IBE, which has nothing to do with protocols. It also provides more flexibility: Our construction in the 2-round setting, can be easily extended to the multi-round setting. Lastly, to the best of our knowledge, this is the first definition of garbling, where the information revealed is computationally, rather than, statistically unique, which allowed us to work with OT that only computationally binds receiver's inputs. This computational perspective may be interesting to other applications.

- Garg et al. [CDG+17] constructed laconic OT from DDH in the CRS model, which allows a receiver to commit to a large input D (of length M) via a short message. Subsequently, a single short message by a sender allows the receiver to learn $m_{D[i]}$, where the messages m_0, m_1 and the location $i \in [M]$ are dynamically chosen by the sender. In their construction, the receiver's message is simply the root h of a Merkle hash tree over D. To enable the receiver to recover $m_{D[i]}$, the sender essentially sends a garbled interactive circuit $\widehat{\mathrm{i}C}$ that has h, i, m_0, m_1 hardcoded inside, and interacts with a de-hashing oracle $\mathcal{O}$ that on input a hash h' returns a pre-image of h'. By interacting with $\mathcal{O}$, $\widehat{\mathrm{i}C}$ can verify that the i'th bit of D is indeed $D[i]$ and then reveal only $m_{D[i]}$ as its output.

- Döttling and Garg [DG17] constructed Identity-Based Encryption (IBE) and Hierarchical Identity-Based Encryption (HIBE) from CDH or factoring. In IBE, a secret key $\mathrm{sk}_{\mathrm{id}}$ is associated with an identity $\mathrm{id} \in \{0,1\}^n$, as well as a ciphertext $\mathrm{ct}_{\mathrm{id}}$. Decrypting a ciphertext $\mathrm{ct}_{\mathrm{id}}$ with the matching secret key $\mathrm{sk}_{\mathrm{id}}$ reveals the encrypted message m. Otherwise, the message remains hidden, even to an adversary that has obtained keys for adaptively chosen identities different from id. In an *inefficient* version of their construction, the setup algorithm samples 2^n public secret key pairs $\{\bar{\mathrm{pk}}_{\mathrm{id}}, \bar{\mathrm{sk}}_{\mathrm{id}}\}_{\mathrm{id} \in \{0,1\}^n}$ of a basic public key encryption scheme, and outputs a master public key mpk that is the root h of a Merkle hash tree over the 2^n public keys $\{\mathrm{pk}_{\mathrm{id}}\}$. A ciphertext of m with id is a garbled interactive circuit $\widehat{\mathrm{i}C}$ that has h, id, m hardcoded in and interacts with a de-hash oracle $\mathcal{O}$. By interacting with $\mathcal{O}$, $\widehat{\mathrm{i}C}$ can verify that the i'th public key is indeed $\mathrm{pk}_{\mathrm{id}}$ and outputs an encryption of m under $\mathrm{pk}_{\mathrm{id}}$. A secret key $\mathrm{sk}_{\mathrm{id}}$ contains the hash values of the nodes on the path from the root to the leaf $\bar{\mathrm{pk}}_{\mathrm{id}}$, and their siblings, together with the secret key $\bar{\mathrm{sk}}_{\mathrm{id}}$. Using the hash values, the decryptor can evaluate $\widehat{\mathrm{i}C}$, and then decrypt the output ciphertext of m using $\mathrm{sk}_{\mathrm{id}}$. So far, the setup algorithm runs in exponential time. To make the scheme efficient, Döttling and Garg used a hash function that is equivocable, so that the scheme does not need to generate the entire Merkle tree at setup, but can "equivocate" the path to each $\mathrm{pk}_{\mathrm{id}}$ at key generation time efficiently.

3 Preliminaries

The security parameter is denoted λ. We recall the notion of polynomial-size circuit classes and families, together with the notion of statistical and computational indistinguishability in Section 3.1.

For the sake of simplicity, we suppose that all circuits in a circuit class have the same input and output lengths. This can be achieved without loss of generality using appropriate paddings. We recall that for any S-size circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$, there exists a universal $\mathrm{poly}(S)$-size circuit family $\{U_\lambda\}_{\lambda \in \mathbb{N}}$ such that for any $\lambda \in \mathbb{N}$, any circuit $C \in \mathcal{C}_\lambda$ with input and output lengths n, l, and any input $x \in \{0,1\}^n$, $U_\lambda(C, x) = C(x)$.

3.1 Circuit Classes and Indistinguishability

CIRCUIT CLASSES AND FAMILIES: We recall the definitions of circuit classes and families.

Definition 3.1 (Class of S-Size Circuits)**.** Let S be a function from $\mathbb{N}$ to $\mathbb{N}$, a *S-size circuit class* is a family of sets $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ of circuits, satisfying that every circuit $C \in \mathcal{C}_\lambda$ has size at most $S(\lambda)$. A *poly-size circuit class* is a S-size circuit class for some polynomial S.

Let n and l be functions from $\mathbb{N}$ to $\mathbb{N}$. We say that $\mathcal{C}$ has input and/or output length n and l, if every circuit $C \in \mathcal{C}_\lambda$ has input and/or output length $n(\lambda)$ and $l(\lambda)$.

For the sake of simplicity, we suppose that all circuits in $\mathcal{C}_\lambda$ have the same input and output lengths. This can be achieved without loss of generality using appropriate paddings.

STATISTICAL AND COMPUTATIONAL INDISTINGUISHABILITY: A function $\mathrm{negl}\colon \mathbb{N} \to \mathbb{N}$ is negligible if for any polynomial $p\colon \mathbb{N} \to \mathbb{N}$, for any large enough $\lambda \in \mathbb{N}$, $\mathrm{negl}(\lambda) < 1/p(\lambda)$.

Definition 3.2 (Indistinguishability)**.** Let $S = \{S_\lambda\}_{\lambda \in \mathbb{N}}$ be an ensemble of subsets of $\{0,1\}^*$, where every element in set S_λ has length $\mathrm{poly}(\lambda)$. Then ensembles $X = \{X_{\lambda,w}\}_{\lambda \in \mathbb{N}, w \in S_\lambda}$ and $Y = \{Y_{\lambda,w}\}_{\lambda \in \mathbb{N}, w \in S_\lambda}$ are *statistically* (resp., *computationally*) *indistinguishable*, denoted as $X \approx_s Y$ (resp., $X \approx Y$), if for any arbitrary-size (resp., polynomial-size) circuit family $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ and any polynomial-size sequence of index $\{w_\lambda \in S\}_{\lambda \in \mathbb{N}}$, there exists a negligible function negl such that, for every $\lambda \in \mathbb{N}$,

$$|\Pr[D_\lambda(w_\lambda, X_{\lambda,w_\lambda}) = 1] - \Pr[D_\lambda(w_\lambda, Y_{\lambda,w_\lambda}) = 1]| \leq \mathrm{negl}(\lambda) \ .$$

Two statistically indistinguishable ensembles are also said to be *statistically close.*

3.2 Garbled Circuit

Definition 3.3 (Garbled Circuit)**.** Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be a poly-size circuit class with input and output lengths n and l. A *garbled circuit* scheme GC for $\mathcal{C}$ is a tuple of four polynomial-time algorithms $\mathsf{GC} = (\mathsf{GC.Gen}, \mathsf{GC.Garble}, \mathsf{GC.Eval}, \mathsf{GC.Sim})$:

Input Labels Generation: $\mathsf{key} \overset{R}{\leftarrow} \mathsf{GC.Gen}(1^\lambda)$ generates input labels $\mathsf{key} = \{\mathsf{key}[i,b]\}_{i \in [n], b \in \{0,1\}}$ (with $\mathsf{key}[i,b] \in \{0,1\}^\kappa$ being the input label corresponding to the value b of the i-th input wire) for the security parameter λ, input length n, and input label length κ;

Circuit Garbling: $\widehat{C} \overset{R}{\leftarrow} \mathsf{GC.Garble}(\mathsf{key}, C)$ garbles the circuit $C \in \mathcal{C}_\lambda$ into $\widehat{C}$;

Evaluation: $y = \mathsf{GC.Eval}(\widehat{C}, \mathsf{key}')$ evaluates the garbled circuit $\mathsf{GC.Garble}$ using input labels $\mathsf{key}' = \{\mathsf{key}'[i]\}_{i \in [n]}$ (where $\mathsf{key}'[i] \in \{0,1\}^\kappa$) and returns the output $y \in \{0,1\}^l$;

Simulation: $(\mathsf{key}', \widetilde{C}) \overset{R}{\leftarrow} \mathsf{GC.Sim}(1^\lambda, y)$ simulates input labels $\mathsf{key}' = \{\mathsf{key}'[i]\}_{i \in [n]}$ and a garbled circuit $\widetilde{C}$ for the security parameter λ and the output $y \in \{0,1\}^l$;

satisfying the following security properties:

Correctness: For any security parameter $\lambda \in \mathbb{N}$, for any circuit $C \in \mathcal{C}_\lambda$, for any input $x \in \{0,1\}^n$, for any key in the image of $\mathsf{GC.Gen}(1^\lambda)$ and any $\widehat{C}$ in the image of $\mathsf{GC.Garble}(\mathsf{key}, C)$:

$$\mathsf{GC.Eval}(\widehat{C}, \{\mathsf{key}[i, x_i]\}_{i \in [n]}) = C(x) \ .$$

Simulatability: The following two distributions are computationally indistinguishable:

$$\left\{ (\{\mathsf{key}[i, x_i]\}_{i\in[n]}, \widehat{C}) \;:\; \begin{array}{l} \mathsf{key} \xleftarrow{R} \mathsf{GC.Gen}(1^\lambda); \\ \widehat{C} \xleftarrow{R} \mathsf{GC.Garble}(\mathsf{key}, C) \end{array} \right\}_{\lambda, C\in\mathcal{C}_\lambda, x\in\{0,1\}^n},$$

$$\left\{ (\mathsf{key}', \widehat{C}) \;:\; (\mathsf{key}', C) \xleftarrow{R} \mathsf{GC.Sim}(1^\lambda, C(x)) \right\}_{\lambda, C\in\mathcal{C}_\lambda, x\in\{0,1\}^n}.$$

We recall that garbled circuit schemes can be constructed from one-way functions.

For the sake of simplicity, if $x \in \{0,1\}^n$ and $\mathsf{key} = \{\mathsf{key}[i,b]\}_{i\in[n],b\in\{0,1\}}$, we define $\mathsf{key}[x] = \{\mathsf{key}[i,x_i]\}_{i\in[n]}$. We extend this notation when the input is a tuple: for example, if $x = (u,v) \in \{0,1\}^{n_1} \times \{0,1\}^{n_2}$, we define $\mathsf{key}[u] = \{\mathsf{key}[i,u_i]\}_{i\in[n_1]}$ and $\mathsf{key}[v] = \{\mathsf{key}[n_1+i, v_i]\}_{i\in[n_2]}$. We also abuse notation and define $\mathsf{key}[[u]]$ (resp., $\mathsf{key}[[v]]$) to be the $2n_1$ (resp., $2n_2$) input labels corresponding to the input wires for u and v: $\mathsf{key}[[u]] = \{\mathsf{key}[i,b]\}_{i\in[n_1],b\in\{0,1\}}$ and $\mathsf{key}[[v]] = \{\mathsf{key}[n_1+i,b]\}_{i\in[n_2],b\in\{0,1\}}$. This notation is also used for $\mathsf{key}' = \{\mathsf{key}[i]\}_{i\in[n]}$: $\mathsf{key}'[[u]] = \{\mathsf{key}'[i]\}_{i\in[n_1]}$ and $\mathsf{key}'[[v]] = \{\mathsf{key}'[n_1+i]\}_{i\in[n_2]}$.

3.3 Multiparty Computation Protocols

We recall the definition of semi-honest multi-party computation (MPC) protocols essentially from [Gol04].

3.3.1 Syntax

Definition 3.4 (Functionality)**.** Let N be a positive integer. An N-party functionality f is a deterministic function from $\bigcup_{\kappa\in\mathbb{N}} (\{0,1\}^\kappa)^N$ to $(\{0,1\}^*)^N$.

For any $i \in [N]$, we write $f_i(\bar{x})$ the i-th element of the output tuple of f on input $\bar{x} \in \bigcup_{\kappa\in\mathbb{N}} (\{0,1\}^\kappa)^N$. For any $I \subseteq [N]$, we write $f_I(\bar{x}) = \{f_i(\bar{x})\}_{i\in I}$. Similarly, $\bar{x}_I = \{x_i\}_{i\in I}$.

We consider MPC protocols where at each round ℓ, each party P_i broadcasts a message m_i^ℓ to all the other parties.

Definition 3.5 (MPC Protocol)**.** Let N be a positive integer, $L = L(\lambda)$ a polynomial in the security parameter, and f an N-party functionality. An L-round *MPC protocol* Π for f is a tuple of two deterministic polynomial-time algorithms $\Pi = (\mathsf{Next}, \mathsf{Output})$:

Next Message: $m_i^\ell = \mathsf{Next}_i(1^\lambda, x_i, r_i, \bar{m}^{<\ell})$ is the message broadcasted by party P_i for $i \in [N]$ in round $\ell \in [L]$, on input $x_i \in \{0,1\}^\kappa$, on random tape $r_i \in \{0,1\}^R$, after receiving the messages $\bar{m}^{<\ell} = \{m_j^{\ell'}\}_{j\in[N],\ell'<\ell}$, where $m_j^{\ell'}$ is the message broadcasted by party P_j on round $\ell' \in [\ell-1]$, and where the input length κ and random tape length R are polynomial in the security parameter λ;

Output: $y_i = \mathsf{Output}_i(1^\lambda, x_i, r_i, \bar{m})$ is the output of party P_i for $i \in [N]$, on input $x_i \in \{0,1\}^\kappa$, on random tape $r_i \in \{0,1\}^R$, after receiving the messages $\bar{m} = \{m_j^\ell\}_{j\in[N],\ell\in[L]}$ as defined above;

satisfying the following property:

Correctness: For any security parameter $\lambda \in \mathbb{N}$, for any inputs $(x_1, \ldots, x_N) \in (\{0,1\}^\kappa)^N$,

$$\Pr\left[\left\{\mathsf{Output}_i(1^\lambda, x_i, r_i, \bar{m})\right\}_{i \in [N]} \neq f(x_1, \ldots, x_N) \quad : \quad \bar{r} \xleftarrow{R} (\{0,1\}^R)^N\right] = 0 \ ,$$

where $m_i^\ell = \mathsf{Next}_i(1^\lambda, x_i, r_i, \bar{m}^{<\ell})$ for $i \in [N]$ and $\ell \in [L]$.

Definition 3.6 (View and Output)**.** Let N be a positive integer. Let f be an N-party functionality. Let $\Pi = (\mathsf{Next}, \mathsf{Output})$ be an MPC protocol for f. Let $I \subseteq [N]$.

- The *view* of parties $\{P_i\}_{i \in I}$ during an execution of Π with security parameter λ, input length κ, inputs $\bar{x} = (x_1, \ldots, x_N) \in (\{0,1\}^\kappa)^N$, random tapes $\bar{r} = (r_1, \ldots, r_N) \in (\{0,1\}^R)^N$ is:

  $$\mathsf{View}_I(1^\lambda, \bar{x}, \bar{r}) = (\bar{x}_i, \bar{r}_I, \bar{m}) \ ,$$

  where $\bar{m}$ is defined as in Definition 3.5.

- The *output* of the protocol for the parties $\{P_i\}_{i \in I}$ is:

  $$\mathsf{Output}_I(\bar{x}, \bar{r}) = \{y_i\}_{i \in I} \ ,$$

  where $y_i = \mathsf{Output}_i(1^\lambda, x_i, r_i, \bar{m})$.

In the sequel, the unary representation 1^λ of the security parameter λ is often omitted from the parameters of Next, Output, and View to simplify notation.

3.3.2 Security against Semi-Honest Adversaries

Definition 3.7 (Security against Semi-Honest Adversaries)**.** Let N be a positive integer. Let f be an N-party functionality. Let Π be an MPC protocol for f. Then Π is *secure against semi-honest adversaries* if there exists a probabilistic polynomial-time algorithm Sim such that for the following two distributions are computationally indistinguishable:

$$\left\{\left(\mathsf{View}_I(1^\lambda, \bar{x}, \bar{r}),\ \mathsf{Output}_I(\bar{x}, \bar{r})\right) \ : \ \bar{r} \xleftarrow{R} (\{0,1\}^R)^N\right\}_{\lambda, I \subseteq [N], \bar{x}} \ ,$$
$$\left\{\left(\mathsf{Sim}(1^\lambda, I, \bar{x}_I, f_I(\bar{x})),\ f_I(\bar{x})\right)\right\}_{\lambda, I \subseteq [N], \bar{x}} \ .$$

3.3.3 Security against Malicious Adversaries

We now recall the notion of security against malicious adversary. We focus on the case with static corruptions and security with abortion. We also recall that we assume that parties have access to a simultaneous broadcast channel.

We first need to define the notions of ideal execution $\mathsf{Ideal}_{I,\mathsf{Sim}}(1^\lambda, \bar{x})$ against a simulator Sim simulating malicious parties $\{P_i\}_{i \in I}$ and of real execution $\mathsf{Real}_{I,A}(1^\lambda, \bar{x})$ against an adversary A playing the roles of malicious parties $\{P_i\}_{i \in I}$. Simulators Sim are defined as non-uniform *expected-poly-time* interactive Turing machines while adversaries A are defined as non-uniform poly-time interactive Turing machines.

Ideal Execution. $\mathsf{Ideal}_{I,\mathsf{Sim}}(1^\lambda, \bar{x})$ is defined by playing the following game with the simulator Sim:

1. The simulator is given I and $\bar{x}_I$.
2. The simulator chooses a vector $\bar{x}'_I = \{\bar{x}'_i\}_{i \in I}$ intuitively corresponding to the extracted inputs of the malicious parties. We set $x'_i = x_i$ for $i \in \bar{I}$, where $\bar{I} = [N] \setminus I$ corresponds to the set of honest parties. As usual, $\bar{x}' = \{\bar{x}'_i\}_{i \in [N]}$.
3. The simulator is given $f_I(\bar{x}')$.
4. The simulator can then decide to abort or proceed. If it aborts, we set $\bar{y}_{\bar{I}} = (\perp, \dots, \perp)$, otherwise, we set $\bar{y}_{\bar{I}} = f_{\bar{I}}(\bar{x}')$.
5. $\mathsf{Ideal}_{I,\mathsf{Sim}}(1^\lambda, \bar{x})$ is defined as $(\bar{y}_{\bar{I}}, z)$ where z is the output of the simulator.

Real Execution. $\mathsf{Real}_{I,A}(1^\lambda, \bar{x})$ is defined by running the MPC protocol where the adversary A controls the malicious parties $\{P_i\}_{i \in I}$ while the honest parties $\{P_i\}_{i \in \bar{I}}$ follow the protocol. It is then defined as the pair $(\bar{y}_{\bar{I}}, z)$, where $\bar{y}_{\bar{I}}$ is the vector of outputs of the honest parties while z is the output of the adversary. The adversary can be rushing: in each round, it can wait for all the messages from the honest parties before sending its own messages.

Definition 3.8 (Malicious Security)**.** Let N be a positive integer. Let f be an N-party functionality. Let Π be an MPC protocol for f. Then Π is *secure against malicious adversaries* if for any non-uniform poly-time interactive Turing machine A, there exists a non-uniform expected-poly-time interactive Turing machine $\mathsf{Sim} = \{\mathsf{Sim}_\lambda\}_{\lambda \in \mathbb{N}}$ such that:

$$\{\mathsf{Ideal}_{I,\mathsf{Sim}}(1^\lambda, \bar{x})\}_{\lambda, I, \bar{x}} \approx \{\mathsf{Real}_{I,A}(1^\lambda, \bar{x})\}_{\lambda, I, \bar{x}} \enspace .$$

3.3.4 Security against Semi-Malicious Adversaries

A semi-malicious adversary [AJL+12] A is similar to a malicious adversary, except that after each round, it has to write on a special *witness tape*, pairs (x_i, r_i) of input x_i and randomness r_i explaining all the messages of the malicious party P_i, for each $i \in I$. The witnesses given in each round do not need to be consistent, and the adversary is rushing: in each round, it can choose its message and witness (x_i, r_i) after having seen the messages of the other parties.

More formally, we define $\mathsf{Real}^{\mathsf{sm}}_{I,A}(1^\lambda, \bar{x})$ as $\mathsf{Real}_{I,A}(1^\lambda, \bar{x})$ except that if at some round ℓ one witness is invalid, then honest parties all abort (do not send any more messages) and output $\perp$.

Definition 3.9 (Semi-Malicious Security)**.** Let N be a positive integer. Let f be an N-party functionality. Let Π be an MPC protocol for f. Then Π is *secure against malicious adversaries* if for any non-uniform poly-time interactive Turing machine A (with an extra witness tape), there exists a non-uniform poly-time interactive Turing machine Sim such that:

$$\{\mathsf{Ideal}_{I,\mathsf{Sim}}(1^\lambda, \bar{x})\}_{\lambda, I, \bar{x}} \approx \{\mathsf{Real}^{\mathsf{sm}}_{I,A}(1^\lambda, \bar{x})\}_{\lambda, I, \bar{x}} \enspace .$$

3.3.5 Delayed-Semi-Malicious Security

Haitner [Hai08] introduced the notion of defensible security for constructing malicious OT from semi-honest OT in a black-box way. In his definition, a defensible adversary is one that outputs at the end of the protocol execution a "defense," which is a pair of input and randomness, and is valid if an honest player with this pair of input and randomness would produce exactly the same messages as what the adversary has sent. In other words, a defensible adversary is like a semi-malicious adversary, except that it only needs to provide a witness (as defined above) at the end of the execution. Haitner then gave an indistinguishability-based definition of OT privacy against defensible adversaries.

In this work, we consider a variant of defensible adversaries, called *delayed-semi-malicious* who are required to provide a witness in the *second last round*, and security only holds if this witness explains the messages of the corrupted players in *all* rounds. Furthermore, we define simulation-based security against these adversaries with a *universal* simulator that can simulate the view of the adversaries by interacting them as black-box in a *straight-line.* In slightly more detail,

- The real world is defined identically as the real world for semi-malicious security, except that, the adversary A is only required to provide a witness in the *second last round*, that is, round $L-1$. If the witness is invalid w.r.t. messages of the corrupted players in the first $L-1$ rounds, then honest parties all abort (do not send any more messages) and output $\perp$ after round $L-1$. In addition, if the witness is invalid w.r.t. messages of the corrupted parties in the last round L, then honest parties again output $\perp$. $\mathsf{Real}^{\mathsf{def}}_{I,A}(1^\lambda, \bar{x})$ denotes the outputs of honest players and the adversary.

- The ideal world is defined identically as the ideal world for semi-malicious security, except that, the universal simulator Sim on input $(1^\lambda, I)$ interacts with adversary A (as a black-box) in a *straight line*, and receives the witness that A outputs after round $L-1$. $\mathsf{Ideal}_{I,\mathsf{Sim}\leftrightarrow A}(1^\lambda, \bar{x})$ denotes the output of honest players and Sim.

Definition 3.10. Let N be a positive integer. Let f be an N-party functionality. Let Π be an MPC protocol for f. Then Π is *delayed-semi-maliciously secure* if there exists a non-uniform expected-poly-time interactive Turing machine Sim, such that, for every non-uniform poly-time interactive Turing machine A:

$$\{\mathsf{Ideal}_{I,\mathsf{Sim}\leftrightarrow A}(1^\lambda, \bar{x})\}_{\lambda, I} \approx \{\mathsf{Real}^{\mathsf{def}}_{I,A}(1^\lambda, \bar{x})\}_{\lambda, I, \bar{x}} \ .$$

4 Definition of Garbled Interactive Circuit Schemes

In this section, we define Garbled Interactive Circuit (GIC) schemes. An overview is provided in Section Section 2.2.

4.1 Interactive Circuits

We start by defining non-deterministic oracles and interactive circuits.

Definition 4.1 (Non-Deterministic Oracles)**.** A non-deterministic oracle $\mathcal{O}$ is a circuit that takes as input a pair of bitstrings $(q, w) \in \{0,1\}^n \times \{0,1\}^m$, called *query* and *witness* respectively, and the output is a l-bit string or a special element $\perp$, called *answer*: $\mathcal{O}(q, w) \in \{0,1\}^l \cup \{\perp\}$. A *poly-size non-deterministic oracle family* is an ensemble of *poly-size* non-deterministic oracles $\mathcal{O} = \{\mathcal{O}_\lambda\}_{\lambda \in N}$.

Definition 4.2. Let $\mathcal{O}$ be a non-deterministic oracle. An L-round interactive circuit $\mathrm{i}C = \{\mathrm{i}C^\ell\}_{\ell\in[L]}$ with oracle $\mathcal{O}$ consists of a list of L next-step circuits.

<u>Execution of $\mathrm{i}C$ with $\mathcal{O}$ on Witnesses $\bar{w}$:</u> An execution of $\mathrm{i}C$ with $\mathcal{O}$ and a list of witnesses $\bar{w} = \{\bar{w}^\ell\}_{\ell\in[L]}$ proceeds in L iterations as follows: In round $\ell \in [L]$, the next-step circuit $\mathrm{i}C^\ell$ on input the state $st^{\ell-1}$ (output in the previous round) and answers $\bar{a}^{\ell-1} = \{a_k^{\ell-1}\}_k$ (to queries $\bar{q}^{\ell-1} = \{q_k^{\ell-1}\}_k$ produced in the previous round), outputs a new state st^ℓ, queries $\bar{q}^\ell = \{q_k^\ell\}_k$, and a (round) output o^ℓ,

$$(st^\ell, \bar{q}^\ell, o^\ell) = \begin{cases} \mathrm{i}C^\ell(st^{\ell-1}, \bar{a}^{\ell-1}) & \text{if } \forall k,\ a_k^{\ell-1} = \mathcal{O}(q_k^{\ell-1}, w_k^{\ell-1}) \neq \bot \\ (\bot, \bot, \bot) & \text{otherwise} \end{cases} .$$

The execution terminates after L rounds, or whenever $\bot$ is output. By convention, st^0 and $\bar{q}^0$ are empty strings.

We say that an execution is *valid* if it terminates after L rounds without outputting $\bot$. We call the list of witnesses $\bar{w}$ the *witnesses* of the execution. The *output* of the execution is the list of round outputs, denoted as $\mathsf{out}(\mathrm{i}C, \mathcal{O}, \bar{w}) = \bar{o} = \{o^\ell\}_{\ell\in[L]}$. The *transcript* of the execution is the list of queries, answers, and outputs, denoted as $\mathsf{trans}(\mathrm{i}C, \mathcal{O}, \bar{w}) = \{\bar{q}^\ell, \bar{a}^\ell, o^\ell\}_{\ell\in[L]}$. (If the execution outputs $\bot$ in round ℓ, $\bar{q}^{\ell'} = \bar{a}^{\ell'} = o^{\ell'} = \bot$ for all $\ell' \geq \ell$.) Finally, we say that $\mathrm{i}C$ has size S if the total size of all circuits are bounded by S. In the rest of the paper, when the oracle $\mathcal{O}$ is clear from the context, we often omit it in the notations and write $\mathsf{out}(\mathrm{i}C, \bar{w})$ and $\mathsf{trans}(\mathrm{i}C, \bar{w})$.

4.2 Garbling Interactive Circuits

As mentioned above, an important difference between GIC schemes and classical garbled circuit schemes is that to evaluate a garbled (plain) circuit, one must obtain encoded inputs, whereas a garble interactive circuit can be evaluated with its oracle $\mathcal{O}$ on input an arbitrary list of witnesses, without encoding. This provides a more powerful functionality, but poses an issue on security: There may exist different lists of witnesses $\bar{w}, \bar{w}'$ that lead to executions with completely different transcripts. In this case, it is unclear how simulation can be done. To circumvent this, we only require the security of the garbling scheme to hold for distributions $\mathrm{i}\mathcal{D}$ of interactive circuits $\mathrm{i}C$ and witnesses $\bar{w}$ (with potentially some auxiliary information aux) that have *computationally unique transcripts* $\mathsf{trans}(\mathrm{i}C, \mathcal{O}, \bar{w})$, in the sense that (given aux) it is hard to find another list of witnesses $\bar{w}'$ that leads to an *inconsistent* transcript $\mathsf{trans}(\mathrm{i}C, \mathcal{O}, \bar{w})$, where inconsistency means:

Definition 4.3 (Consistent Transcripts)**.** We say that two transcripts $\{\bar{q}^\ell, \bar{a}^\ell, o^\ell\}_{\ell\in[L]}$ and $\{\bar{q}'^\ell, \bar{a}'^\ell, o'^\ell\}_{\ell\in[L]}$ are *consistent* if for every $\ell \in [L]$, $(\bar{q}^\ell, \bar{a}^\ell, o^\ell) = (\bar{q}'^\ell, \bar{a}'^\ell, o'^\ell)$, or one of them is $(\bot, \bot, \bot)$. Otherwise, we say that the two transcripts are *inconsistent*.

Note that one can always produce a list of invalid witnesses that lead to an invalid execution. Therefore, difference due to outputting $\bot$ does not count as inconsistency. Next, we formally define these distributions that produce unique transcripts.

Definition 4.4 (Unique-Transcript Distribution)**.** Let $\mathcal{O} = \{\mathcal{O}_\lambda\}_{\lambda\in\mathbb{N}}$ be a non-deterministic oracle family. Let $\mathrm{i}\mathcal{D} = \{\mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}\}_{\lambda\in\mathbb{N},\mathrm{id}}$ be an ensemble of efficiently samplable distributions over tuples $(\mathrm{i}C, \bar{w}, \mathsf{aux})$. We say that $\mathrm{i}\mathcal{D}$ is a *(computationally) unique-transcript* distribution for $\mathcal{O}$, if

Valid Execution: For any $\lambda \in \mathbb{N}$, any index $\mathrm{id} \in \{0,1\}^{\mathrm{poly}(\lambda)}$, and any $(\mathrm{i}C, \bar{w}, \mathsf{aux})$ in the support of $\mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}$, the execution of $\mathrm{i}C$ with $\mathcal{O}_\lambda$ and $\bar{w}$ is valid.

Computationally Unique Transcript: For any poly-size circuit family $A = \{A_\lambda\}_\lambda$, any sequence of indices $\{\mathrm{id}_\lambda\}_\lambda$, there is a negligible function negl, such that for any λ:

$$\Pr\Big[\mathsf{trans}(\mathrm{i}C, \mathcal{O}_\lambda, \bar{w}') \text{ and } \mathsf{trans}(\mathrm{i}C, \mathcal{O}_\lambda, \bar{w}) \text{ are inconsistent } : \\ (\mathrm{i}C, \bar{w}, \mathsf{aux}) \overset{R}{\leftarrow} \mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}_\lambda};\ \bar{w}' \overset{R}{\leftarrow} A_\lambda(\mathrm{i}C, \bar{w}, \mathsf{aux})\Big] \le \mathrm{negl}(\lambda) \ .$$

It is a *statistically unique-transcript distribution* if the second property holds for any arbitrary-size circuit family $A = \{A_\lambda\}_\lambda$.

Now, we are ready to define GIC schemes.

Definition 4.5 (Garbled Interactive Circuit Schemes)**.** Let $\mathcal{O} = \{\mathcal{O}_\lambda\}_{\lambda \in \mathbb{N}}$ be a non-deterministic oracle family, and $\mathrm{i}\mathcal{D} = \{\mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}\}_{\lambda \in \mathbb{N},\mathrm{id}}$ be a unique-transcript distribution for $\mathcal{O}$. A *garbled interactive circuit* scheme for $(\mathcal{O}, \mathrm{i}\mathcal{D})$ is a tuple of three polynomial-time algorithms $\mathsf{GiC} = (\mathsf{GiC.Garble}, \mathsf{GiC.Eval}, \mathsf{GiC.Sim})$:

Garbling: $\widehat{\mathrm{i}C} \overset{R}{\leftarrow} \mathsf{GiC.Garble}(1^\lambda, \mathrm{i}C)$ garbles an interactive circuit $\mathrm{i}C$ into a garbled interactive circuit $\widehat{\mathrm{i}C}$;

Evaluation: $o^\ell = \mathsf{GiC.Eval}(\widehat{\mathrm{i}C}, \bar{w}^{<\ell})$ evaluates a garbled interactive circuit $\widehat{\mathrm{i}C}$ with a partial list of witness $\bar{w}^{<\ell}$, and outputs the ℓ-th round output o^ℓ;

Simulation: $\widetilde{\mathrm{i}C} \overset{R}{\leftarrow} \mathsf{GiC.Sim}(1^\lambda, T)$ simulates a garbled circuit $\widetilde{\mathrm{i}C}$ from a transcript T of an execution;

satisfying the following properties:

Correctness: For any $\lambda \in \mathbb{N}$, any index $\mathrm{id} \in \{0,1\}^{\mathrm{poly}(\lambda)}$, any $(\mathrm{i}C, \bar{w}, \mathsf{aux})$ in the support of $\mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}$, it holds that

$$\Pr\Big[\{\mathsf{GiC.Eval}(\widehat{\mathrm{i}C}, \bar{w}^{<\ell})\}_{\ell \in [L]} \;=\; \mathsf{out}(\mathrm{i}C, \mathcal{O}_\lambda, \bar{w}) \quad : \quad \widehat{\mathrm{i}C} \;\overset{R}{\leftarrow}\; \mathsf{GiC.Garble}(1^\lambda, \mathrm{i}C)\Big] \;=\; 1 \ ;$$

Simulatability: The following two distributions are computationally indistinguishable:

$$\left\{(\mathrm{i}C, \bar{w}, \mathsf{aux}, \widehat{\mathrm{i}C}) \;:\; \begin{array}{l} (\mathrm{i}C, \bar{w}, \mathsf{aux}) \overset{R}{\leftarrow} \mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}; \\ \widehat{\mathrm{i}C} \overset{R}{\leftarrow} \mathsf{GiC.Garble}(1^\lambda, \mathrm{i}C) \end{array}\right\}_{\lambda,\mathrm{id}} ,$$

$$\left\{(\mathrm{i}C, \bar{w}, \mathsf{aux}, \widetilde{\mathrm{i}C}) \;:\; \begin{array}{l} (\mathrm{i}C, \bar{w}, \mathsf{aux}) \overset{R}{\leftarrow} \mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}; \\ \widetilde{\mathrm{i}C} \overset{R}{\leftarrow} \mathsf{GiC.Sim}(1^\lambda, \mathsf{trans}(\mathrm{i}C, \mathcal{O}_\lambda, \bar{w})) \end{array}\right\}_{\lambda,\mathrm{id}} .$$

Remark 4.6*.* In this paper, we always consider perfect correctness for all primitives for the sake of simplicity. We could relax this notion to correctness up to a negligible error probability if, in addition, we ask that no non-uniform poly-time adversary can generate inputs and randomness which would not satisfy the correctness property, with non-negligible probability. In other words, in the case of GIC schemes, semi-maliciously generated GIC should satisfy the correctness property (except with negligible probability). This additional property is not needed for our semi-honest constructions.

Remark 4.7. Note that the above notion of garbled interactive circuit scheme is defined for a non-deterministic oracle family and an associated unique-transcript distribution. We can also consider a uniform version, where the oracle is defined to be a single non-deterministic uniform *algorithm*. The uniform version would suffice for our application of constructing 2-round MPC protocols. We however choose to define the non-uniform version since it is stronger, and fits better with the garbling.

5 2-Round Semi-Honest MPC Protocols

In this section, we present our construction of 2-round semi-honest MPC protocols. For that purpose, we first introduce the notion of functional commitment. We then show the MPC construction.

5.1 New Tool: Functional Commitment

Definition 5.1 ((Zero-Knowledge) Functional Commitment)**.** Let $\mathcal{G} = \{\mathcal{G}_\lambda\}_{\lambda\in\mathbb{N}}$ be a poly-size circuit class. A *(zero-knowledge) functional commitment* scheme FC for $\mathcal{G}$ is a tuple of four polynomial-time algorithms $\mathsf{FC} = (\mathsf{FC.Com}, \mathsf{FC.FOpen}, \mathsf{FC.FVer}, \mathsf{FC.Sim})$:

Commitment: $c = \mathsf{FC.Com}(1^\lambda, v; \rho)$ generates a commitment c of $v \in \{0,1\}^n$ using random tape $\rho \in \{0,1\}^\tau$, for the security parameter λ, where the random tape length τ is polynomial in λ;

Functional Opening: $d = \mathsf{FC.FOpen}(c, G, v, \rho)$ derives from the commitment c and the random tape ρ used to generate it, a functional decommitment d of c to $y = G(v)$ for $G \in \mathcal{G}_\lambda$;

Functional Verification: $b = \mathsf{FC.FVer}(c, G, y, d)$ outputs $b = 1$ if d is a valid functional decommitment of c to y for $G \in \mathcal{G}_\lambda$; and outputs $b = 0$ otherwise;

Simulation: $(c, d) \xleftarrow{R} \mathsf{FC.Sim}(1^\lambda, G, y)$ simulates a commitment c together with a functional decommitment d of c to y for $G \in \mathcal{G}_\lambda$;

satisfying the following properties:

Correctness: For any security parameter $\lambda \in \mathbb{N}$, for any $v \in \{0,1\}^n$, for any circuit $G \in \mathcal{G}_\lambda$, for any $\rho \in \{0,1\}^\tau$, it holds that if $c = \mathsf{FC.Com}(1^\lambda, v; \rho)$, then:

$$\mathsf{FC.FVer}(c, G, G(v), \mathsf{FC.FOpen}(c, G, v, \rho)) = 1 \ ;$$

Semi-Honest Functional Binding: For any polynomial-time circuit family $A = \{A_\lambda\}_{\lambda\in\mathbb{N}}$, there exists a negligible function negl, such that for any $\lambda \in \mathbb{N}$, for any $v \in \{0,1\}^n$, for any circuit $G \in \mathcal{G}_\lambda$:

$$\Pr\Big[\mathsf{FC.FVer}(c, G, y, d) = 1 \text{ and } y \neq G(v) \ : \\ \rho \xleftarrow{R} \{0,1\}^\tau;\ c = \mathsf{FC.Com}(1^\lambda, v; \rho);\ (y, d) \xleftarrow{R} A_\lambda(1^\lambda, c, v, \rho)\Big] \leq \mathrm{negl}(\lambda) \ ;$$

Simulatability: The following two distributions are computationally indistinguishable:

$$\left\{(c, d) \ : \ \begin{matrix} \rho \xleftarrow{R} \{0,1\}^\tau;\ c \xleftarrow{R} \mathsf{FC.Com}(1^\lambda, v; \rho); \\ d = \mathsf{FC.FOpen}(c, G, v, \rho) \end{matrix} \right\}_{\lambda,G,v} ,$$

$$\{(c, d) \ : \ (c, d) \xleftarrow{R} \mathsf{FC.Sim}(1^\lambda, G, G(v)) \ \}_{\lambda,G,v} \ .$$

Note that the simulatability property implies the standard hiding property of commitments, if each circuit class $\mathcal{G}_\lambda$ contains a constant circuit: Consider indeed any constant circuit $C(x) = \alpha$, the fact that (c, d) can be simulated from C and α implies that c hides the message committed inside.

Let us now define the non-deterministic oracle family associated to FC.

Definition 5.2. Let $\mathcal{G} = \{\mathcal{G}_\lambda\}_{\lambda\in\mathbb{N}}$ be a poly-size circuit class. Let $\mathsf{FC} = (\mathsf{FC.Com}, \mathsf{FC.FOpen}, \mathsf{FC.FVer}, \mathsf{FC.Sim})$ be a *functional commitment* scheme for $\mathcal{G}$. We define the following *associated non-deterministic oracle family* $\mathcal{O}^{\mathsf{FC}} = \{\mathcal{O}^{\mathsf{FC}}_\lambda\}_{\lambda\in\mathbb{N}}$:

$$\mathcal{O}^{\mathsf{FC}}_\lambda((c, G), (y, d)) = \begin{cases} y & \text{if } \mathsf{FC.FVer}(c, G, y, d) = 1 \\ \perp & \text{otherwise;} \end{cases}$$

5.2 Construction of 2-Round Semi-Honest MPC

Tools: Let f be an arbitrary N-party functionality.[10] To construct a 2-round semi-honest MPC protocol $\widetilde{\Pi}$ for f, we rely on the following tools:

- A semi-honestly secure L-round MPC protocol $\Pi = (\mathsf{Next}, \mathsf{Output})$ for f. We will refer to this protocol the "inner MPC protocol".

  Recall that Next is next message function that computes the message broadcasted by party P_i in round ℓ, $m_i^\ell = \mathsf{Next}_i(x_i, r_i, \bar{m}^{<\ell})$, on input x_i and random tape r_i, after receiving messages $\bar{m}^{<\ell} = \{m_j^{\ell'}\}_{j\in[N],\ell'<\ell}$ broadcasted by parties P_j on previous rounds. And Output is the output function that computes the output of party P_i, $y_i = \mathsf{Output}_i(x_i, r_i, \bar{m})$, after receiving all the messages $\bar{m} = \{m_j^\ell\}_{j\in[N],\ell\in[L]}$. The security parameter λ is an implicit parameter 1^λ of Next and Output.

- A functional commitment scheme $\mathsf{FC} = (\mathsf{FC.Com}, \mathsf{FC.FOpen}, \mathsf{FC.FVer}, \mathsf{FC.Sim})$ for the class of all S-size circuits with a sufficiently large polynomial bound S. We denote by $\mathcal{O}^{\mathsf{FC}}$ the associated non-deterministic oracle family defined in Definition 5.2.

- A garbled interactive circuit scheme $\mathsf{GiC} = (\mathsf{GiC.Garble}, \mathsf{GiC.Eval})$ for the oracle $\mathcal{O}^{\mathsf{FC}}$ and the unique-transcript distribution $\mathrm{i}\mathcal{D} = \{\mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}\}_{\lambda\in\mathbb{N},\mathrm{id}}$ that we define later.

We will show that using the constructions in Sections 6 and 7, we can construct the two last tools from 2-round (semi-honest) OT. With the above tools, our 2-round MPC protocol $\widetilde{\Pi} = (\widetilde{\mathsf{Next}}, \widetilde{\mathsf{Output}})$ for f proceed as follows:

The First Round: Each party P_i computes its first message $\tilde{m}_i^1 = \widetilde{\mathsf{Next}}_i(x_i, \tilde{r}_i, \emptyset)$, using security parameter λ, input x_i, random tape $\tilde{r}_i$, and no messages, as follows.

1. Take a sufficient long substring r_i of $\tilde{r}_i$ as the random tape for running the inner MPC protocol Π.

2. Commit L times to (x_i, r_i) using the functional commitment scheme FC: for $\ell \in [L]$, $c_i^\ell = \mathsf{FC.Com}(1^\lambda, (x_i, r_i); \rho_i^\ell)$, where all the ρ_i^ℓ's (and r_i) are non-overlapping substrings of $\tilde{r}_i$.

[10] Formal definitions of MPC protocol and N-party functionality are provided in Section 3.3.

3. Broadcast the first message $\widetilde{m}_i^1 = \{c_i^\ell\}_{\ell\in[L]}$, and keep $\{\rho_i^\ell\}_{\ell\in[L]}$ secret.

The Second Round: Each party P_i computes its second message $\widetilde{m}_i^2 = \widetilde{\mathsf{Next}}_i(x_i, \tilde{r}_i, \{\widetilde{m}_j^1\}_{j\in N})$, using all first messages $\{\widetilde{m}_j^1\}_{j\in N}$ as follows:

1. Garble the interactive circuit $\mathrm{i}C_i = \{\mathrm{i}C_i^\ell\}_{\ell\in[L]}$ defined in Fig. 4:

$$\widehat{\mathrm{i}C}_i \xleftarrow{R} \mathsf{GiC.Garble}(1^\lambda, \mathrm{i}C_i) \ .$$

2. Broadcast the second message $\widetilde{m}_i^2 = \widehat{\mathrm{i}C}_i$.

The Interactive Circuit $\mathrm{i}C_i$

Constants: 1^λ, ℓ, x_i, r_i, the ℓ-th commitments c_j^ℓ for each party P_j (part of the first message $\widetilde{m}_j^1$), and the randomness ρ_i^ℓ used in commitment c_i^ℓ.

Inputs: $(st^{\ell-1}, \bar{a}^{\ell-1})$ where for $\ell > 1$:

- The state $st^{\ell-1} = \bar{m}^{<\ell-1}$ contains the inner MPC messages of the first $\ell - 1$ rounds.
- The answers $a_j^\ell = m_j^{\ell-1}$ are the answers of the non-deterministic oracle $\mathcal{O}^{\mathsf{FC}}$ to the queries $q_j^\ell = (c_j^{\ell-1}, G_j^{\ell-1})$, for $j \in [N]$, where the circuit $G_j^{\ell-1}$ is defined by $G_j^{\ell-1}(\star,\star) = \mathsf{Next}_j(\star,\star,\bar{m}^{<\ell-1})$.

These inputs define $\bar{m}^{<\ell}$.

Procedure:

1. Define the circuit G_j^ℓ as $G_j^\ell(\star,\star) = \mathsf{Next}_j(\star,\star,\bar{m}^{<\ell})$, for $j \in [N]$.
2. Compute the ℓ-th message of P_i in the inner MPC:
$m_i^\ell = \mathsf{Next}_i\left(x_i, r_i, \bar{m}^{<\ell}\right)$.
3. Compute the associated functional decommitment of c_i^ℓ:
$d_i^\ell = \mathsf{FC.FOpen}(c_i^\ell, G_i^\ell, (x_i, r_i), \rho_i^\ell)$.
4. Compute the next queries: for every $j \in [N]$, $q_j^\ell = (c_j^\ell, G_j^\ell)$.
5. Define the next state to be $st^\ell = \bar{m}^{<\ell}$ and the output to be $o_i^\ell = (m_i^\ell, d_i^\ell)$.

Output: $(st^\ell,\ \bar{q}^\ell,\ o_i^\ell)$.

Figure 4: The interactive circuit $\mathrm{i}C_i$

The Output Function: Each party P_i computes its output $y_i = \widetilde{\mathsf{Output}}_i(x_i, \tilde{r}_i, \{\widetilde{m}_j^1, \widetilde{m}_j^2\}_{j\in[N]})$, using all first and second messages $\{\widetilde{m}_j^1, \widetilde{m}_j^2\}_{j\in N}$ as follows. Proceed in L iterations to evaluate the N garbled circuits $\{\widehat{\mathrm{i}C}_j\}_{j\in[N]}$ in parallel. Before iteration $\ell \in [L]$ starts, the following invariant holds:

Invariant: After the first $(\ell - 1)$ iterations, P_i has obtained for every $j \in [N]$ and every $\ell' < \ell$:

- the inner MPC message $m_j^{\ell'}$ generated in the ℓ'-th round by party P_j, and

- the associated functional decommitment $d_j^{\ell'}$ of $c_j^{\ell'}$ for the circuit $G_j^{\ell'}(\star,\star) = \mathsf{Next}_j(\star,\star,\bar{m}^{<\ell'})$.

We define $\bar{w}^{<\ell} = \{w_j^{\ell'}\}_{j,\ell'<\ell} = \{(m_j^{\ell'}, d_j^{\ell'})\}_{\ell'<\ell}$.

In the first round $\ell = 1$, all these messages and functional decommitments are empty. Thus, the invariant holds initially. With the above, P_i does the following in iteration ℓ: for every $j \in [N]$: $(m_j^\ell, d_j^\ell) = o_j^\ell = \mathsf{GiC.Eval}(\widehat{\mathrm{i}C}_j, \bar{w}^{<\ell})$.

After all L iterations, P_i obtains the set of all messages $\bar{m}$, and computes the output by invoking the output function of the inner MPC protocol: $y_i = \mathsf{Output}_i\,(x_i, r_i, \bar{m})$.

<u>Unique-Transcript Distribution:</u> We now define the unique-transcript distribution $\mathrm{i}\mathcal{D} = \{\mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}\}_{\lambda\in\mathbb{N},\mathrm{id}}$ (for the garbled interactive circuit $\mathrm{i}C_i$) as follows: $\mathrm{id} = (i, \bar{x}, \bar{r}, \bar{m})$ and $\mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}$ is

$$\left\{ (\mathrm{i}C_i,\ \bar{w},\ \bar{\rho} = \{\rho_j^\ell\}_{j,\ell}) \ :\ \begin{array}{l} \forall j \in [N],\ \forall \ell \in [L], \\ \qquad \rho_j^\ell \xleftarrow{R} \{0,1\}^{|\rho_j^\ell|};\ c_j^\ell = \mathsf{FC.Com}(1^\lambda, (x_j, r_j); \rho_j^\ell); \\ \qquad G_j^\ell(\star,\star) = \mathsf{Next}_j(\star,\star,\bar{m}^{<\ell}); \\ \qquad d_j^\ell = \mathsf{FC.FOpen}(c_j^\ell, G_j^\ell, (x_j, r_j), \rho_j^\ell); \\ \bar{w} = \{w_j^\ell = (m_j^\ell, d_j^\ell)\}_{j,\ell}; \\ \mathrm{i}C_i \text{ defined in Fig. 4} \end{array} \right\} .$$

The unique-transcript property follows from the semi-honest functional binding property of FC thanks to Lemmas 6.4 and 7.2.

<u>Security:</u> We have the following theorem.

Theorem 5.3. *If the inner MPC $\Pi = (\mathsf{Next}, \mathsf{Output})$ is correct and secure against semi-honest adversaries, if the functional scheme FC is correct, semi-honest functional binding, and simulatable, if the garbled interactive circuit scheme GiC is correct and simulatable, then the MPC protocol defined above is correct and secure against semi-honest adversaries.*

Proof of Theorem 5.3. Correctness is straightforward. Let us prove security against semi-honest adversaries.

We need to exhibit a polynomial-time simulator of the view of any subset $I \subseteq [N]$ of corrupted parties, namely:

$$\widetilde{\mathsf{View}}_I(1^\lambda, I, \bar{x}, \widetilde{\bar{r}}) = (\bar{x}_I, \widetilde{\bar{r}}_I, \widetilde{\bar{m}})$$

where $\widetilde{\bar{r}} = \{\tilde{r}_i\}_{i\in[N]}$ are honestly-generated random tapes of the parties. We recall that $\bar{x}_I = \{x_i\}_{i\in I}$ and $\widetilde{\bar{r}}_I = \{\tilde{r}_i\}_{i\in I}$.

The simulator first run the simulator of the inner MPC protocol and get $(\bar{r}_I, \bar{m})$. It then simulates all the messages $\widetilde{\bar{m}}$ together with the random tapes $\widetilde{\bar{r}}_I$ of the corrupted parties as follows.

<u>*First round*</u>:

- For each corrupted party P_i with $i \in I$, generate the commitments $c_i^\ell = \mathsf{FC.Com}(1^\lambda, (x_i, r_i); \rho_i^\ell)$ and the first message $\widetilde{m}_i^1 = \{c_i^\ell\}_{\ell\in[L]}$ as in the real protocol.

- For each honest party P_i with $i \notin I$, simulate the commitments:

  $$(c_i^\ell, d_i^\ell) \xleftarrow{R} \mathsf{FC.Sim}(1^\lambda, G_i^\ell, m_i^\ell)\ ,$$

  for $\ell \in [L]$ and for the circuit $\mathbb{G}_i^\ell$ defined by $G_i^{\ell-1}(\star,\star) = \mathsf{Next}_i(\star,\star,\bar{m}^{<\ell-1})$. Then set the first message $\widetilde{m}_i^1 = \{c_i^\ell\}_{\ell\in[L]}$.

Second round:

- For each corrupted party P_i with $i \in I$, generate the garbled interactive circuit $\widehat{\mathrm{i}C}_i \xleftarrow{R} \mathsf{GiC.Garble}(1^\lambda, \mathrm{i}C_i)$ and the second message $\widetilde{m}_i^2 = \widehat{\mathrm{i}C}_i$, as in the real protocol.
- For each honest party P_i with $i \notin I$, compute $\bar{q}^\ell = \{c_j^{\ell-1}, G_j^{\ell-1}\}_{j\in[N]}$, $\bar{w}^\ell = \{m_j^\ell, d_j^\ell\}_{j\in[N]}$, and $o^\ell = (m_i^\ell, d_i^\ell)$, for $\ell \in [L]$, and simulate the garbled interactive circuit:
$$\widetilde{\mathrm{i}C}_i \xleftarrow{R} \mathsf{GiC.Sim}(1^\lambda, \{\bar{q}^\ell, \bar{w}^\ell, o^\ell\}_{\ell\in[L]}) \ .$$
The second message is $\widetilde{m}_i^2 = \widetilde{\mathrm{i}C}_i$.

We now need to prove that the simulated view is indistinguishable from the real view. More formally we need to prove that the following two distributions are computationally indistinguishable:

$$\mathcal{D}_0 = \left\{ \Big(\mathsf{View}_I(1^\lambda, \bar{x}, \bar{r}),\ \mathsf{Output}_I(\bar{x}, \bar{r}) \Big) \ :\ \bar{r} \xleftarrow{R} \left(\{0,1\}^R\right)^N \right\}_{\lambda, I\subseteq[N], \bar{x}} \ ,$$
$$\mathcal{D}_1 = \left\{ \Big(\mathsf{Sim}(1^\lambda, I, \bar{x}_I, f_I(\bar{x})),\ f_I(\bar{x}) \Big) \right\}_{\lambda, I\subseteq[N], \bar{x}} \ .$$

For that, we consider a sequence of $N + N^2$ hybrids $\{\mathcal{H}_{1,i^\star}\}_{i^\star\in[N]}$ and $\{\mathcal{H}_{2,(\ell^\star,j^\star)}\}_{\ell^\star\in[L],j^\star\in[N]}$:

Hybrid $\mathcal{H}_{1,i^\star}$: This hybrid is similar to $\mathcal{D}_0$ (the real protocol), except that for the second messages of parties P_i for $i \le i^\star$ which are simulated as in $\mathcal{D}_1$: when $i \notin I$, $\widetilde{m}_i^2 = \widetilde{\mathrm{i}C}_i = \mathsf{GiC.Sim}(1^\lambda, \{\bar{q}^\ell, \bar{w}^\ell, o^\ell\}_{\ell\in[L]})$.

Let $\mathcal{H}_{1,0} = \mathcal{D}_0$. We have the following claim.

Claim 5.4. *If* GiC *is simulatable, then for any* $i^\star \in [N]$*,* $\mathcal{H}_{1,i^\star-1}$ *and* $\mathcal{H}_{1,i^\star}$ *are computationally indistinguishable.*

Proof. First, if $P_{i^\star}$ is corrupted ($i^\star \in I$), then $\mathcal{H}_{1,i^\star-1}$ and $\mathcal{H}_{1,i^\star}$ are actually the same distribution. Let us focus on the case $i^\star \notin I$.

The only difference between $\mathcal{H}_{1,i^\star-1}$ and $\mathcal{H}_{1,i^\star}$ is that $\widetilde{m}_i^2 = \widetilde{\mathrm{i}C}_i$ is simulated in the latter distribution. Thus, these two distributions are computationally indistinguishable if GiC is simulatable. □

Hybrid $\mathcal{H}_{2,(\ell^\star,j^\star)}$: We consider the lexicographic order $\prec$ (or any linear order) over the pairs $(\ell^\star, j^\star) \in [L] \times [N]$, and we define $(\ell^\star, j^\star)^-$ to be the predecessor of $(\ell^\star, j^\star)$.

The hybrid $\mathcal{H}_{2,(\ell^\star,j^\star)}$ is similar to $\mathcal{D}_1$ (the simulated protocol), except that for all $(\ell, j) \succ (\ell^\star, j^\star)$, c_j^ℓ and d_j^ℓ are generated as in the real protocol ($\mathcal{D}_0$):

$$c_j^\ell = \mathsf{FC.Com}(1^\lambda, (x_j, r_j); \rho_j^\ell);\ d_j^\ell = \mathsf{FC.FOpen}(c_j^\ell, G_j^\ell, m_j^\ell, \rho_j^\ell) \ ,$$

where ρ_j^ℓ is a uniform random tape.

Let $\mathcal{H}_{1,N} = \mathcal{H}_{2,(1,1)^-}$. We have the straightforward following claim.

Claim 5.5. *If* FC *is simulatable, then for any* $(\ell^\star, j^\star) \in [L] \times [N]$*,* $\mathcal{H}_{2,(\ell^\star,j^\star)^-}$ *and* $\mathcal{H}_{2,(\ell^\star,j^\star)}$ *are computationally indistinguishable.*

Furthermore, the only difference between $\mathcal{H}_{2,(N,N)}$ and $\mathcal{D}_1$ is that in the latter distribution, the inner MPC messages $\bar{m}$ are simulated by the inner MPC simulator. Thus, we have the following claim.

Claim 5.6. *If the inner MPC is secure against semi-honest adversaries, then* $\mathcal{H}_{2,(N,N)}$ *and* $\mathcal{D}_1$ *are computationally indistinguishable.*

We remark that we do not directly use the semi-honest functional binding property of the FC scheme, as it is implied by the simulatability property of the GIC scheme. □

6 Garbled Interactive Circuit from Witness Selector

In this section, we show how to construct GIC from another tool we call witness selector, which can be seen as generalization of witness encryption to languages defined by a non-deterministic oracle family $\mathcal{O}$. Contrary to witness encryption, each query to $\mathcal{O}$ may have multiple answers, as long as at most one can be found efficiently.

We first define the notion of computationally unique-answer distribution for $\mathcal{O}$ and the notion of witness selector for such a distribution. Then we show how to construct a garbled interactive circuit scheme for $(\mathcal{O}, \mathrm{i}\mathcal{D})$ from any witness selector for a unique-answer distribution for $\mathcal{O}$ which is *consistent* with the unique-transcript distribution $\mathrm{i}\mathcal{D}$.

6.1 Witness Selector

Definition 6.1 (Unique-Answer Distribution)**.** Let $\mathcal{O}$ be a non-deterministic oracle family. Let $\mathrm{w}\mathcal{D} = \{\mathrm{w}\mathcal{D}_{\lambda,\mathsf{id}}\}_{\lambda\in\mathbb{N},\mathsf{id}}$ be an ensemble of efficiently samplable distributions over tuples (q, w, aux). We way that $\mathrm{w}\mathcal{D}$ is a *(computationally) unique-answer distribution* for $\mathcal{O}$ if

Non-$\perp$ Answer: For any $\lambda \in \mathbb{N}$, any index $\mathsf{id} \in \{0,1\}^{\mathrm{poly}(\lambda)}$, and any (q, w, aux) in the support of $\mathrm{w}\mathcal{D}_{\lambda,\mathsf{id}}$, $\mathcal{O}_\lambda(q, w) \neq \perp$.

Computationally Unique Answer: For any poly-size circuit family $A = \{A_\lambda\}_{\lambda\in\mathbb{N}}$, for any sequence of indices $\{\mathsf{id}_\lambda\}_\lambda$, there exists a negligible function negl, such that for any $\lambda \in \mathbb{N}$:

$$\Pr\Big[\mathcal{O}_\lambda(q, w') \neq \perp \text{ and } \mathcal{O}_\lambda(q, w') \neq \mathcal{O}_\lambda(q, w) \ : \ (q, w, \mathsf{aux}) \xleftarrow{R} \mathrm{w}\mathcal{D}_{\lambda,\mathsf{id}_\lambda};\ w' \xleftarrow{R} A_\lambda(q, w, \mathsf{aux})\Big] \leq \mathrm{negl}(\lambda) \ .$$

It is a *statistically unique-answer distribution* if the second property holds for any arbitrary-size circuit family $A = \{A_\lambda\}_\lambda$.

Definition 6.2 (Witness Selector)**.** Let $\mathcal{O} = \{\mathcal{O}_\lambda\}_{\lambda\in N}$ be a non-deterministic oracle family, and $\mathrm{w}\mathcal{D} = \{\mathrm{w}\mathcal{D}_{\lambda,\mathsf{id}}\}_{\lambda\in\mathbb{N},\mathsf{id}}$ a unique-answer distribution for $\mathcal{O}$. A *witness selector* scheme for $(\mathcal{O}, \mathrm{w}\mathcal{D})$ is a tuple of two polynomial-time algorithms $\mathsf{WS} = (\mathsf{WS.Enc}, \mathsf{WS.Dec})$:

Encryption: $\mathsf{ct} \xleftarrow{R} \mathsf{WS.Enc}(1^\lambda, q, \mathsf{M})$ encrypts messages $\mathsf{M} = \{\mathsf{M}[i,b]\}_{i\in[l],b\in\{0,1\}}$ for a query q, into a ciphertext ct, where each message has the same length $|\mathsf{M}[i,b]| = \text{poly}(\lambda)$;

Decryption: $\mathsf{M}' = \mathsf{WS.Dec}(\mathsf{ct}, w)$ decrypts a ciphertext ct into messages $\mathsf{M}' = \{\mathsf{M}'[i]\}_{i\in[l]}$ using a witness w;

satisfying the following properties:

Correctness: For any security parameter $\lambda \in \mathbb{N}$, for any index id,

for any (q, w, aux) in the support of $\mathrm{w}\mathcal{D}_{\lambda,\mathsf{id}}$, for any messages $\mathsf{M} = \{\mathsf{M}[i,b]\}_{i,b}$, for $a = \mathcal{O}(q, w)$:

$$\Pr\left[\mathsf{WS.Dec}(\mathsf{WS.Enc}(1^\lambda, q, \mathsf{M}),\ w) = \{\mathsf{M}[i, a_i]\}_{i\in[l]}\right] = 1 \ ;$$

Semantic Security: The following two distributions are indistinguishable:

$$\{(q, w, \mathsf{aux}, \mathsf{WS.Enc}(1^\lambda, q, \mathsf{M})) \ : \ (q, w, \mathsf{aux}) \xleftarrow{R} \mathrm{w}\mathcal{D}_{\lambda,\mathsf{id}} \ \}_{\lambda,\mathsf{id},\mathsf{M}} \ ,$$

$$\left\{ (q, w, \mathsf{aux}, \mathsf{WS.Enc}(1^\lambda, q, \mathsf{M}')) \ : \ \begin{array}{l} (q, w, \mathsf{aux}) \xleftarrow{R} \mathrm{w}\mathcal{D}_{\lambda,\mathsf{id}}; \\ a = \mathcal{O}_\lambda(q, w); \\ \{\mathsf{M}'[i,b]\}_{i,b} = \{\mathsf{M}[i,a_i]\}_{i,b} \end{array} \right\}_{\lambda,\mathsf{id},\mathsf{M}} .$$

6.2 Garbled Interactive Circuit from Witness Selector

Let $\mathcal{O} = \{\mathcal{O}_\lambda\}_{\lambda\in\mathbb{N}}$ be a poly-size non-deterministic oracle family. Let $\mathrm{i}\mathcal{D} = \{\mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}\}_{\lambda\in\mathbb{N},\mathrm{id}}$ be an ensemble of efficiently samplable distributions over tuples $(\mathrm{i}C, \bar{w}, \mathsf{aux})$, where $\mathrm{i}C$ is an L-round interactive circuit. We suppose that $\mathrm{i}\mathcal{D}$ is a unique-transcript distribution for $\mathcal{O}$. To construct a garbled interactive circuit scheme $\mathsf{GiC} = (\mathsf{GiC.Garble}, \mathsf{GiC.Eval}, \mathsf{GiC.Sim})$ for $(\mathcal{O}, \mathrm{i}\mathcal{D})$, we rely on the following tools:

- A witness selector $\mathsf{WS} = (\mathsf{WS.Enc}, \mathsf{WS.Dec})$ for $(\mathcal{O}, \mathrm{w}\mathcal{D})$ where $\mathrm{w}\mathcal{D} = \{\mathrm{w}\mathcal{D}_{\lambda,\mathsf{id}}\}$ is a unique-answer distribution for $\mathcal{O}$, which is consistent with the unique-transcript distribution $\mathrm{i}\mathcal{D}$ (consistency is defined below in Definition 6.3).
- A garbled circuit scheme $\mathsf{GC} = (\mathsf{GC.Gen}, \mathsf{GC.Garble}, \mathsf{GC.Eval}, \mathsf{GC.Sim})$ for the class of all S-size circuits with a sufficiently large polynomial bound S.

The naive notion of consistence would be: $\mathrm{i}\mathcal{D}$ is consistent with $\mathrm{w}\mathcal{D}$ if each query q_k^ℓ and its associated witness w_k^ℓ follow the same distribution as $\mathrm{w}\mathcal{D}$. Unfortunately, this is not sufficient as the adversary may learn some auxiliary information. Instead, we require that for any ℓ and k, the distribution of $(\mathrm{i}C, \bar{w}, \mathsf{aux}) \xleftarrow{R} \mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}$ can be simulated from $(q, w, \mathsf{aux}) \xleftarrow{R} \mathrm{w}\mathcal{D}_{\lambda,\mathsf{id}'}$ (for some index id' function of id) in such a way that q_k^ℓ and w_k^ℓ match q and w. More formally, we have the following definition.

Definition 6.3 (Consistency of $\mathrm{i}\mathcal{D}$ and $\mathrm{w}\mathcal{D}$)**.** Let $\mathrm{i}\mathcal{D} = \{\mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}\}_{\lambda\in\mathbb{N},\mathrm{id}}$ be an ensemble of efficiently samplable distributions over tuples $(\mathrm{i}C, \bar{w}, \mathsf{aux})$ where $\mathrm{i}C$ is an L-round interactive circuit. Let $\mathrm{w}\mathcal{D} = \{\mathrm{w}\mathcal{D}_{\lambda\in N,\mathsf{id}'}\}$ be an ensemble of efficiently samplable distributions over tuples (q, w, aux). The distributions $\mathrm{i}\mathcal{D}$ and $\mathrm{w}\mathcal{D}$ are consistent if for any sequence of indices $\{\ell^\star\}_{\lambda,\mathrm{id}}$ and $\{k^\star\}_{\lambda,\mathrm{id}}$, there exists two probabilistic polynomial-time functions g and h, such that the following two distributions are identical:

$$\left\{ ((\mathrm{i}C, \bar{w}, \mathsf{aux}),\ (q_{k^\star}^{\ell^\star}, w_{k^\star}^{\ell^\star})) \ :\ \begin{array}{l} (\mathrm{i}C, \bar{w}, \mathsf{aux}) \overset{R}{\leftarrow} \mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}; \\ \{\bar{q}^\ell, \bar{a}^\ell, o^\ell\}_{\ell\in[L]} = \mathsf{trans}(\mathrm{i}C, \mathcal{O}, \bar{w}) \end{array} \right\}_{\lambda,\mathrm{id}} ,$$

$$\left\{ (h(q, w, \mathsf{aux}),\ (q, w)) \ :\ (q, w, \mathsf{aux}) \overset{R}{\leftarrow} \mathrm{w}\mathcal{D}_{\lambda, g(1^\lambda, \mathrm{id})} \right\}_{\lambda,\mathrm{id}} .$$

We have the following straightforward lemma.

Lemma 6.4. *If* $\mathrm{i}\mathcal{D}$ *and* $\mathrm{w}\mathcal{D}$ *are consistent and if* $\mathrm{w}\mathcal{D}$ *is unique-answer, then* $\mathrm{i}\mathcal{D}$ *is unique-transcript.*

The construction proceeds as follows:

Garbling: $\widehat{\mathrm{i}C} \overset{R}{\leftarrow} \mathsf{GiC.Garble}(1^\lambda, \mathrm{i}C)$ garbles the interactive circuit $\mathrm{i}C = \{\mathrm{i}C^\ell\}_{\ell\in[L]}$ into $\widehat{\mathrm{i}C}$ as follows: For ℓ from L to 1,

1. Generate input labels $\mathsf{key}^\ell \overset{R}{\leftarrow} \mathsf{GC.Gen}(1^\lambda)$.
2. Garble the circuit $\mathrm{i}C.\mathsf{AugNext}^\ell$ defined in Fig. 5:

$$\mathrm{i}C.\widehat{\mathsf{AugN}}\mathsf{ext}^\ell \overset{R}{\leftarrow} \mathsf{GC.Garble}(\mathsf{key}^\ell, \mathrm{i}C.\mathsf{AugNext}^\ell) \ .$$

And output $\widehat{\mathrm{i}C} = \{\mathrm{i}C.\widehat{\mathsf{AugN}}\mathsf{ext}^\ell\}_{\ell\in[L]}$.

Evaluation: $o^{\ell'} = \mathsf{GiC.Eval}(\widehat{\mathrm{i}C}, \bar{w}^{<\ell'})$ evaluates the garbled interactive circuit $\widehat{\mathrm{i}C}$ with the partial list of witnesses $\bar{w}^{<\ell'}$ as follows. For $\ell \in [\ell']$, we denote by key'^ℓ the set of input labels that we actually use to evaluate $\mathrm{i}C.\widehat{\mathsf{AugN}}\mathsf{ext}^\ell$ (i.e., it contains one label per input wire; key'^1 and key'^{L+1} are the empty strings). key'^ℓ is composed of two parts $\mathsf{key}'^\ell[[st^\ell]]$ and $\mathsf{key}'^\ell[[\bar{a}^\ell]] = \{\mathsf{key}'^\ell[[a_k^\ell]]\}_k$ corresponding to the input wires for st^ℓ and $\bar{a}^\ell$ respectively: $\mathsf{key}'^\ell = (\mathsf{key}'^\ell[[st^\ell]], \{\mathsf{key}'^\ell[[a_k^\ell]]\}_k)$. For ℓ from 1 to ℓ', the evaluator does the following:

1. Evaluate the garbled circuit $\mathrm{i}C.\widehat{\mathsf{AugN}}\mathsf{ext}^\ell$:

$$(\mathsf{key}'^{\ell+1}[[st^\ell]],\ \bar{q}^\ell,\ \bar{\mathsf{ct}}^\ell,\ o^\ell) = \mathsf{GC.Eval}(\mathrm{i}C.\widehat{\mathsf{AugN}}\mathsf{ext}^\ell, \mathsf{key}'^\ell) \ .$$

2. If $\ell < \ell'$, for each $k \in [|\bar{\mathsf{ct}}^\ell|]$, decrypt ct_k^ℓ using the witness w_k^ℓ:

$$\mathsf{key}'^{\ell+1}[[a_k^\ell]] = \mathsf{WS.Dec}(\mathsf{ct}_k^\ell, w_k^\ell) \ .$$

And output $o^{\ell'}$ (except if $\perp$ was output before).

Simulation: $\widetilde{\mathrm{i}C} \overset{R}{\leftarrow} \mathsf{GiC.Sim}(1^\lambda, T)$ simulates a garbled interactive circuit $\widetilde{\mathrm{i}C}$ from a transcript $T = \{\bar{q}^\ell, \bar{a}^\ell, o^\ell\}_{\ell\in[L]}$ as follows. As for evaluation, for $\ell \in [L]$, we denote by $\mathsf{key}'^\ell = (\mathsf{key}'^\ell[[st^\ell]], \{\mathsf{key}'^\ell[[a_k^\ell]]\}_k)$ the set of input labels that we actually use as inputs to $\mathrm{i}C.\widehat{\mathsf{AugN}}\mathsf{ext}^\ell$ (i.e., it contains one label per input wire). For ℓ from L to 1, the simulator does the following:

1. Define $\mathsf{key}^{\ell+1}$ to be such that $\mathsf{key}^{\ell+1}[i, b] = \mathsf{key}'^{\ell+1}[i]$ for all input wire i and all bits $b \in \{0, 1\}$. key'^{L+1} and key^{L+1} are empty.

2. Encrypt the labels generated for the round $\ell + 1$ corresponding to the answer $\bar{a}^\ell$, using the witness selector scheme: for each k,

$$\mathsf{ct}_k^\ell \overset{R}{\leftarrow} \mathsf{WS.Enc}(1^\lambda, \bar{q}^\ell, \mathsf{key}^{\ell+1}[[a_k^\ell]]) \ .$$

   (For $\ell = L$, $\bar{\mathsf{ct}}^\ell$ and $\mathsf{key}^{\ell+1}$ are empty.)

3. Simulate the garbling of $\widehat{\mathrm{i}C.\mathsf{AugNext}^\ell}$, using its outputs $\mathsf{key}'^{\ell+1}[[st^\ell]] = \mathsf{key}^{\ell+1}[st^\ell]$ (for $\ell = L$, this value is empty), $\bar{q}^{\ell+1}$, $\bar{\mathsf{ct}}^\ell$, and o^ℓ:

$$\widehat{\mathrm{i}C.\mathsf{AugNext}^\ell} \overset{R}{\leftarrow} \mathsf{GC.Sim}(1^\lambda, (\mathsf{key}'^{\ell+1}[[st^\ell]], \bar{q}^\ell, \bar{\mathsf{ct}}^\ell, o^\ell)) \ .$$

Correctness follows immediately from the correctness properties of the witness selector WS and of the garbled circuit scheme GC.

The Augmented Next Message Function $\mathrm{i}C.\mathsf{AugNext}^\ell$

Constants: 1^λ, ℓ, $\mathrm{i}C^\ell$, and the keys $\mathsf{key}_{i^\star}^{\ell+1}$ for the $(\ell+1)$-th garbled circuit.

Inputs: The previous state $st^{\ell-1}$ and the answers $\bar{a}^{\ell-1}$ (of the non-deterministic oracle $\mathcal{O}$ to the queries $\bar{q}^{\ell-1}$).

Procedure:

1. Compute $(st^\ell, \bar{q}^\ell, o^\ell) = \mathrm{i}C^\ell(st^{\ell-1}, \bar{a}^{\ell-1})$. If $o^\ell = \bot$, abort and output $(\bot, \bot, \bot, \bot)$. By convention, st^0 and $\bar{a}^0$ are empty strings.

2. For every $k \in [|\bar{q}^\ell|]$, generate using a hardcoded random tape:

$$\mathsf{ct}_k^\ell = \mathsf{WS.Enc}(1^\lambda, q_k^\ell, \mathsf{key}^{\ell+1}[[a_k^\ell]]) \ ,$$

   where $\mathsf{key}^{\ell+1}[[a_k^\ell]]$ is the tuple of input labels $\mathsf{key}^{\ell+1}[i, b]$ for all $b \in \{0, 1\}$ and for the input wires i corresponding to the input a_k^ℓ of $\mathrm{i}C.\mathsf{AugNext}^{\ell+1}$. Set $\bar{\mathsf{ct}}^\ell = \{\mathsf{ct}_k^\ell\}_k$. By convention, $\bar{q}^\ell$ is empty if $\ell = L$.

3. Select the input labels for the next step, corresponding to the new state st^ℓ: $\mathsf{key}^{\ell+1}[st^\ell] = \{\mathsf{key}^{\ell+1}[i, st_i^\ell]\}_i$. By convention, st^ℓ and $\mathsf{key}^{\ell+1}[st^\ell]$ are empty if $\ell = L$.

Output: $(\mathsf{key}^{\ell+1}[st^\ell]$, $\bar{q}^\ell$, $\bar{\mathsf{ct}}^\ell$, $o^\ell)$.

Figure 5: The augmented next message function $\mathrm{i}C.\mathsf{AugNext}^\ell$

Security: We have the following theorem.

Theorem 6.5. *If* GC *is correct and simulatable, if* WS *is correct and semantically secure, if* $\mathrm{w}\mathcal{D}$ *is unique-answer, and if* $\mathrm{i}\mathcal{D}$ *and* $\mathrm{w}\mathcal{D}$ *are consistent, then the garbled interactive circuit scheme* GiC *defined above is correct and simulatable.*

Proof of Theorem 6.5. Correctness is straightforward. Let us prove simulatability.

S<span style="font-variant:small-caps"></span>IMULATABILITY: We need to prove the computational indistinguishability of the following two distributions:

$$\begin{aligned}\mathcal{D}_0 =& \Big\{\widehat{\mathrm{i}C} \ : \ (\mathrm{i}C, \bar{w}, \mathsf{aux}) \xleftarrow{R} \mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}};\ \widehat{\mathrm{i}C} \xleftarrow{R} \mathsf{GiC.Garble}(1^\lambda, \mathrm{i}C)\Big\}_{\lambda,\mathrm{id}} \ , \\ \mathcal{D}_1 =& \Big\{\widetilde{\mathrm{i}C} \ : \ (\mathrm{i}C, \bar{w}, \mathsf{aux}) \xleftarrow{R} \mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}};\ \widetilde{\mathrm{i}C} \xleftarrow{R} \mathsf{GiC.Sim}(1^\lambda, \mathsf{trans}(\mathrm{i}C, \mathcal{O}_\lambda, \bar{w}))\Big\}_{\lambda,\mathrm{id}} \ .\end{aligned}$$

For that, we introduce $2L+2$ hybrid distributions $\mathcal{H}_{0,0}, \mathcal{H}_{0,1}, \mathcal{H}_{1,0}, \mathcal{H}_{1,1}, \mathcal{H}_{2,0}, \dots, \mathcal{H}_{L,1}$:

Hybrid $\mathcal{H}_{\ell,0}$: This hybrid is similar to $\mathcal{D}_1$, except that $\mathrm{i}C.\widehat{\mathsf{AugNext}}^{>\ell}$ and $\mathsf{key}^{>\ell}$ (thus in particular $\bar{\mathsf{ct}}^{>\ell}$ is not defined) are generated as in $\mathcal{D}_0$.

We have the following straightforward claim.

Claim 6.6. *$\mathcal{H}_{0,0}$ and $\mathcal{D}_0$ are the same distribution.*

Hybrid $\mathcal{H}_{\ell,1}$: For $\ell = 0$, this hybrid is the same as $\mathcal{H}_{0,0}$.

For $\ell \geq 1$, this hybrid is similar to $\mathcal{H}_{\ell,0}$, except that: $\bar{\mathsf{ct}}^\ell$ is computed as:

$$\mathsf{ct}_k^\ell \xleftarrow{R} \mathsf{WS.Enc}(1^\lambda, q_k^\ell, \mathsf{key}''^{\ell+1}[[a_k^\ell]]) \ ,$$

where $\mathsf{key}''^{\ell+1}[i,b] = \mathsf{key}[i, a_{k,j}^\ell]$ for each $b \in \{0,1\}$ and each input wire i corresponding to the j-th bit of the input a_k^ℓ in $\mathrm{i}C.\mathsf{AugNext}^\ell$. In other words, for each input wire of the answers $\bar{a}^\ell$, instead of encrypting both possible input labels with the witness selector, we encrypt twice the input label which is actually used.

Thanks to consistency between the distributions $\mathrm{i}\mathcal{D}$ and $\mathrm{w}\mathcal{D}$, the semantic security of the witness selector ensures that this hybrid is indistinguishable from the previous one.

Claim 6.7. *If $\mathrm{i}\mathcal{D}$ and $\mathrm{w}\mathcal{D}$ are consistent and if WS is semantically secure, then for any $\ell \in [L]$, $\mathcal{H}_{\ell,0}$ and $\mathcal{H}_{\ell,1}$ are computationally indistinguishable.*

Let $\mathcal{H}_{L+1,0}$ be the distribution $\mathcal{D}_1$. As the only difference between $\mathcal{H}_{\ell,1}$ and $\mathcal{H}_{\ell+1,0}$ is that in the latter hybrid, $\mathrm{i}C.\widehat{\mathsf{AugNext}}^{\ell+1}$ and $\mathsf{key}^{\ell+1}$ are simulated via $\mathsf{GC.Sim}$ instead of being generated via $\mathsf{GC.Garble}$, we have the following claim.

Claim 6.8. *If GC is simulatable, then for any $\ell \in \{0, \dots, L\}$, $\mathcal{H}_{\ell,1}$ and $\mathcal{H}_{\ell+1,0}$ are computationally indistinguishable.*

□

7 Functional Commitment with Witness Selector

In this section, we start with defining the witness selector (WS) associated with a functional commitment scheme, which suffices for constructing the GIC schemes needed for our construction of MPC protocols in Section 5. Then, we show how to construct a functional commitment scheme with witness selector from any 2-round OT scheme.

Definition 7.1. Let $\mathsf{FC} = (\mathsf{FC.Com}, \mathsf{FC.FOpen}, \mathsf{FC.FVer}, \mathsf{FC.Sim})$ be a *functional commitment* scheme for a poly-size circuit class $\mathcal{G}$, and $\mathcal{O}^{\mathsf{FC}} = \{\mathcal{O}^{\mathsf{FC}}_\lambda\}_{\lambda \in \mathbb{N}}$ the *non-deterministic oracle family* associated with FC defined in Definition 5.2.

Define the following *unique-answer distribution* associated with FC: $\mathrm{w}\mathcal{D}^{\mathsf{FC}} = \{\mathrm{w}\mathcal{D}^{\mathsf{FC}}_{\lambda,G,v}\}$ where $\lambda \in \mathbb{N}$, $G \in \mathcal{G}_\lambda$, and $v \in \{0,1\}^n$:

$$\mathrm{w}\mathcal{D}^{\mathsf{FC}}_{\lambda,G,v} = \left\{ ((c,G),\ (y,d),\ \mathsf{aux} = \rho)\ :\ \begin{array}{l} \rho \xleftarrow{R} \{0,1\}^\tau;\ c = \mathsf{FC.Com}(1^\lambda, v; \rho); \\ y = G(v);\ d = \mathsf{FC.FOpen}(c, G, v, \rho) \end{array} \right\} .$$

A *witness selector* associated to FC is a witness selector for $(\mathcal{O}^{\mathsf{FC}}, \mathrm{w}\mathcal{D}^{\mathsf{FC}})$.

The unique-answer property of $\mathrm{w}\mathcal{D}^{\mathsf{FC}}$ for $\mathcal{O}^{\mathsf{FC}}$ follows from the semi-honest functional binding property of the functional commitment FC. Furthermore, we have the following lemma.

Lemma 7.2. *The distribution* $\mathrm{i}\mathcal{D}$ *defined in Section 5.2 is consistent with* $\mathrm{w}\mathcal{D}^{\mathsf{FC}}$.

Proof of Lemma 7.2. Let us fix the security parameter λ and the index $\mathrm{id} = (i, \bar{x}, \bar{r}, \bar{m})$. Let $\ell^\star \in [L]$ and $j^\star$ be an integer. We show that we can simulate $(\mathrm{i}C_i, \bar{w}, \bar{\rho})$ from $((c,G),(y,d),\rho) \xleftarrow{R} \mathrm{w}\mathcal{D}^{\mathsf{FC}}_{\lambda,G,v}$ for some well-chosen G and v, and such that the query $q^{\ell^\star}_{j^\star}$ and its witness $w^{\ell^\star}_{j^\star}$ in the execution of $\mathrm{i}C_i$ satisfy $q^{\ell^\star}_{j^\star} = (c,G)$ and $w^{\ell^\star}_{j^\star} = (y,d)$. For that, we choose $G = G^{\ell^\star}_{j^\star}$ with $G^{\ell}_{j^\star}(\star,\star) = \mathsf{Next}_{j^\star}(\star,\star,\bar{m}^{<\ell^\star})$, and $v = (x_{j^\star}, r_{j^\star})$. Then we set $c^{\ell^\star}_{j^\star} = c$, $d^{\ell^\star}_{j^\star} = d$, and $\rho^{\ell^\star}_{j^\star} = \rho$ (which implies $y = m^{\ell^\star}_{j^\star}$). Finally, we generate $(c^\ell_j, \rho^\ell_j, G^\ell_j, d^\ell_j)$ for $(j,\ell) \neq (j^\star, \ell^\star)$ as in $\mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}$ and define $\bar{w} = \{w^\ell_j = (m^\ell_j, d^\ell_j)\}_{j,\ell}$ and $\mathrm{i}C_i$ as in Fig. 4. The resulting distribution

$$\left\{((\mathrm{i}C_i,\ \bar{w},\ \bar{\rho} = \{\rho^\ell_j\}),\ ((c,G),(y,d))\right\}$$

is the same as:

$$\left\{ ((\mathrm{i}C_i, \bar{w}, \mathsf{aux}),\ (q^{\ell^\star}_{j^\star}, w^{\ell^\star}_{j^\star}))\ :\ \begin{array}{l} (\mathrm{i}C_i, \bar{w}, \mathsf{aux}) \xleftarrow{R} \mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}; \\ \{\bar{q}^\ell, \bar{a}^\ell, o^\ell\}_{\ell \in [L]} = \mathsf{trans}(\mathrm{i}C, \mathcal{O}, \bar{w}) \end{array} \right\} .$$

This is what we wanted to prove. □

This means, given a functional commitment with an associated WS, we can instantiate the construction of GIC in Section 6, which in turn can be used to instantiate the construction of MPC protocols in Section 5.

7.1 Recall: 2-Round Oblivious Transfer

We recall the definition of 2-round OT.

Definition 7.3. A *2-round oblivious transfer (OT)* is a tuple of three polynomial-time algorithms $\mathsf{OT} = (\mathsf{OT.Send}^1, \mathsf{OT.Send}^2, \mathsf{OT.Output})$:

First Round: $\mu^1 = \mathsf{OT.Send}^1(1^\lambda, \sigma; \rho)$ generates the first flow μ^1 (from the receiver to the sender) for the selection bit $\sigma \in \{0,1\}$, the security parameter λ, and the random tape $\rho \in \{0,1\}^\tau$, where τ is polynomial in λ;

Second Round: $\mu^2 \overset{R}{\leftarrow} \mathsf{OT.Send}^2(\mu^1, x_0, x_1)$ generates the second flow (from the sender to the receiver) for the messages $(x_0, x_1) \in (\{0,1\}^k)^2$, where the message length k is polynomial in λ;

Output: $x = \mathsf{OT.Output}(\mu^2, \sigma, \rho)$ computes the output $x \in \{0,1\}^k$ of the receiver;

satisfying the following properties:

Correctness: For any security parameter $\lambda \in \mathbb{N}$, for any selection bit $\sigma \in \{0,1\}$, for any messages $(x_0, x_1) \in (\{0,1\}^k)^2$, for any $\rho \in \{0,1\}^\tau$, it holds that:

$$\Pr\Big[\mu^1 = \mathsf{OT.Send}^1(1^\lambda, \sigma; \rho);\ \mu^2 \overset{R}{\leftarrow} \mathsf{OT.Send}^2(\mu^1, x_0, x_1)\ :\ x_\sigma = \mathsf{OT.Output}(\mu^2, \sigma, \rho)\Big] = 1\ ;$$

Receiver Privacy: The following two distributions are computationally indistinguishable:

$$\{\mathsf{OT.Send}^1(1^\lambda, 0; \rho)\ :\ \rho \overset{R}{\leftarrow} \{0,1\}^\tau\}_\lambda\ ,$$
$$\{\mathsf{OT.Send}^1(1^\lambda, 1; \rho)\ :\ \rho \overset{R}{\leftarrow} \{0,1\}^\tau\}_\lambda\ ;$$

Semi-Honest Sender Privacy: The following two distributions are computationally indistinguishable:

$$\{(\rho, \mathsf{OT.Send}^2(\mu^1, x_0, x_1))\ :\ \rho \overset{R}{\leftarrow} \{0,1\}^\tau;\ \mu^1 = \mathsf{OT.Send}^1(1^\lambda, \sigma; \rho)\}_{\lambda,\sigma,x_0,x_1}\ ,$$
$$\{(\rho, \mathsf{OT.Send}^2(\mu^1, x_\sigma, x_\sigma))\ :\ \rho \overset{R}{\leftarrow} \{0,1\}^\tau;\ \mu^1 = \mathsf{OT.Send}^1(1^\lambda, \sigma; \rho)\}_{\lambda,\sigma,x_0,x_1}\ .$$

7.2 Functional Commitment with WS from 2-Round OT

Let $\mathcal{G} = \{\mathcal{G}_\lambda\}_{\lambda\in\mathbb{N}}$ be a S-size circuit class (where S is polynomial in λ). To construct a functional commitment scheme $\mathsf{FC} = (\mathsf{FC.Com}, \mathsf{FC.FOpen}, \mathsf{FC.FVer}, \mathsf{FC.Sim})$ with an associated witness selector $\mathsf{WS} = (\mathsf{WS.Enc}, \mathsf{WS.Dec})$, we rely on the following tools:

- A poly-size universal circuit family $\{U_\lambda\}_{\lambda\in\mathbb{N}}$ for $\mathcal{G}$; we recall that $U_\lambda(v, G) = G(v)$, for $G \in \mathcal{G}_\lambda$ and $v \in \{0,1\}^n$.
- A garbled circuit scheme $\mathsf{GC} = (\mathsf{GC.Gen}, \mathsf{GC.Garble}, \mathsf{GC.Eval}, \mathsf{GC.Sim})$ for the circuit class $\{\{U_\lambda(\star, v)\}_{v\in\{0,1\}^n}\}_{\lambda\in\mathbb{N}}$ of partially evaluated universal circuits on any possible input v; we recall that the input of the circuit $U_\lambda(\star, v)$ is a circuit $G \in \mathcal{G}_\lambda$ represented by a S-bit string $(G[1], \ldots, G[S]) \in \{0,1\}^S$.
- A garbled circuit scheme $\mathsf{oGC} = (\mathsf{oGC.Gen}, \mathsf{oGC.Garble}, \mathsf{oGC.Eval}, \mathsf{oGC.Sim})$ for the class of $\mathsf{o}S$-sized circuits with a sufficiently large polynomial bound $\mathsf{o}S$. The prefix "o" stands for "outer" as this garbled circuit scheme will be used in the WS encryption procedure to garble a circuit containing the $\mathsf{GC.Eval}$.
- A 2-round OT $\mathsf{OT} = (\mathsf{OT.Send}^1, \mathsf{OT.Send}^2, \mathsf{OT.Output})$ with sufficiently large message size $k = |x_0| = |x_1|$.[11]

[11] This is without loss of generality, as we can always repeat in parallel a 1-bit-message 2-round OT to get a poly(λ)-bit-message 2-round OT.

The construction proceeds as follows:

Commitment: $c = \mathsf{FC.Com}(1^\lambda, v; \rho)$ commits to $v \in \{0,1\}^n$ as follows:

1. Generate input labels $\mathsf{key} \xleftarrow{R} \mathsf{GC.Gen}(1^\lambda)$ (using a random tape derived from ρ).
2. Garble $C = U_\lambda(\star, v)$, which is the universal circuit partially evaluated on v: $\widehat{C} \xleftarrow{R} \mathsf{GC.Garble}(\mathsf{key}, C)$.
3. For each $i \in [S]$, for each bit $b \in \{0,1\}$, for each $j \in [|\mathsf{key}[i,b]|]$, generate a first flow $\mu^1_{i,b,j} = \mathsf{OT.Send}^1(1^\lambda, \mathsf{key}[i,b]_j; \rho_{i,b,j})$, where $\mathsf{key}[i,b]_j$ is the j-th bit of the input label $\mathsf{key}[i,b]$ and where the random tape $\rho_{i,b,j}$ is derived from ρ.

And returns $c = (\widehat{C}, \{\mu^1_{i,b,j}\})$.

Functional Opening: $d = \mathsf{FC.FOpen}(c, G, v, \rho)$ derives the functional decommitment d of c to $y = G(v) = U_\lambda(G, v)$ as follows: $d = \{\mathsf{key}'[i], \{\rho'_{i,j}\}_j\}_{i \in [S]}$, where $\mathsf{key}'[i] = \mathsf{key}[i, G[i]]$ and $\rho'_{i,j} = \rho_{i,G[i],j}$.

Functional Verification: $\mathsf{FC.FVer}(c, G, y, d)$ returns 1, if and only if for all $i \in [S]$ and $j \in [|\mathsf{key}'[i]|]$:

$$\mu^1_{i,G[i],j} = \mathsf{OT.Send}^1(1^\lambda, \mathsf{key}'[i]_j; \rho'_{i,j}) \qquad \text{and} \qquad y = \mathsf{GC.Eval}(\widehat{C}, \mathsf{key}') \ .$$

Simulation: $(c, d) \xleftarrow{R} \mathsf{FC.Sim}(1^\lambda, G, y)$ generates the commitment c and its functional decommitment d as follows:

1. Simulate the garble circuit and its partial key: $(\mathsf{key}', \widetilde{C}) \xleftarrow{R} \mathsf{GC.Sim}(1^\lambda, y)$.
2. Define key as follows: $\mathsf{key}[i, G[i]] = \mathsf{key}'[i]$ and $\mathsf{key}[i, 1 - G[i]] = 0^{|\mathsf{key}'[i]|}$.
3. For each $i \in [S]$, for each bit $b \in \{0,1\}$, for each $j \in [|\mathsf{key}[i,b]|]$, generate a first flow $\mu^1_{i,b,j} = \mathsf{OT.Send}^1(1^\lambda, \mathsf{key}[i,b]_j; \rho_{i,b,j})$, where $\mathsf{key}[i,b]_j$ is the j-th bit of the input label $\mathsf{key}[i,b]$ and where the random tape $\rho_{i,b,j}$ is derived from ρ.

And sets $c = (\widetilde{C}, \{\mu^1_{i,b,j}\})$ and $d = \{\mathsf{key}'[i], \{\rho'_{i,j}\}_j\}_{i \in [S]}$, where $\mathsf{key}'[i] = \mathsf{key}[i, G[i]]$ and $\rho'_{i,j} = \rho_{i,G[i],j}$.

Encryption: $\mathsf{ct} \xleftarrow{R} \mathsf{WS.Enc}(1^\lambda, (c, G), \mathsf{M})$ encrypts the messages $\mathsf{M} = \{\mathsf{M}[I, B]\}_{I,B}$ for $q = (c, G)$ into ct as follows:

1. For every $I \in [l]$ and $B \in \{0,1\}$, create the circuit:
$$\mathsf{o}C_{I,B}(\mathsf{key}') = \begin{cases} \mathsf{M}[I,B] & \text{if } y_I = B \text{ where } y = \mathsf{GC.Eval}(\widehat{C}, \mathsf{key}'), \\ \bot & \text{otherwise.} \end{cases}$$
2. For every $I \in [l]$ and $B \in \{0,1\}$, garble this circuit: $\mathsf{okey}_{I,B} \xleftarrow{R} \mathsf{oGC.Gen}(1^\lambda)$ and $\widehat{\mathsf{o}C}_{I,B} \xleftarrow{R} \mathsf{oGC.Garble}(\mathsf{okey}, \mathsf{o}C_{I,B})$; we write $\mathsf{okey}_{I,B}[i,j,b]$ the key corresponding to the j'bit of the input $\mathsf{key}'[i]$ of $\mathsf{o}C_{I,B}$ being b (i.e., $\mathsf{key}'[i]_j = b$, where $\mathsf{key}' = \{\mathsf{key}'[i]\}$ is the input of the circuit $\mathsf{o}C_{I,B}$).

3. Define the OT messages: $x_{i,j,b} = \{\mathsf{okey}_{I,B}[i,j,b]\}_{I,B}$.
4. Compute the second flows of the OT corresponding to the first flows $\mu^1_{i,G[i],j}$: $\mu^2_{i,j} \overset{R}{\leftarrow} \mathsf{OT.Send}^2(\mu^1_{i,G[i],j}, x_{i,j,0}, x_{i,j,1})$;

and return $\mathsf{ct} = (\{\widehat{\mathsf{o}C}_{I,B}\}_{I\in[l],B\in\{0,1\}}, \{\mu^2_{i,j}\}_{i\in[S],j\in[|\mathsf{key}[i,0]|]})$.

Decryption: $\mathsf{M} = \mathsf{WS.Dec}(\mathsf{ct},(y,d))$ decrypts ct as follows:

1. For every $i \in [S]$ and $j \in [|\mathsf{key}'[i]|]$, compute:
$$\{\mathsf{okey}'_{I,B}[i,j]\}_{I,B} = x_{i,j,\mathsf{key}'[i]_j} = \mathsf{OT.Output}(\mu^2_{i,j}, \mathsf{key}'[i]_j, \rho'_{i,j}) \ ;$$
2. For every $I \in [l]$ and $B = y_I$, evaluate the garble circuit $\widehat{\mathsf{o}C}_{I,B}$:
$$\mathsf{M}[I,B] = \mathsf{oGC.Eval}(\widehat{\mathsf{o}C}_{I,B}, \{\mathsf{okey}'_{I,B}[i,j]\}_{i,j})$$

and return $\mathsf{M} = \{\mathsf{M}[I,y_I]\}_{I\in[l]}$.

Correctness of the functional commitment scheme is straightforward. Correctness for the decryption of the witness selector comes from the fact that:
$$\{\mathsf{okey}'_{I,B}[i,j]\}_{I,B} = \{\mathsf{okey}_{I,B}[i,j,\mathsf{key}[i,G[i]]_j]\}_{I,B}$$
and therefore $\mathsf{oGC.Eval}(\widehat{\mathsf{o}C}_{I,B}, \{\mathsf{okey}'_{I,B}[i,j]\}_{i,j})$ is a correct evaluation of the garbled circuit $\widehat{\mathsf{o}C}_{I,B}$ on the input $\mathsf{key}' = \{\mathsf{key}[i,G[i]]\}_{i\in[S]}$, satisfying $\mathsf{GC.Eval}(\widehat{C},\mathsf{key}') = C(G) = G(v) = y$.

<u>Security:</u> We have the following theorem.

Theorem 7.4. *If* OT *is correct, receiver-private, and (semi-honest) sender-private, then the functional commitment scheme* FC *defined above is correct, semi-honest functionally binding, and simulatable. Furthermore, the associated witness selector* WS *is correct and semantically secure.*

Proof of Theorem 7.4. Correctness is straightforward. Let us now prove semi-honest functional binding of FC, simulatability of FC, and semantic security of WS.

<u>Semi-Honest Functional Binding:</u> Semi-honest functional binding follows directly from the semantic security of the witness selector, which we prove below. Indeed, an adversary A against semi-honest functional binding generate a functional decommitment d to some value y for some circuit G, on input ρ and $c = \mathsf{FC.Com}(1^\lambda, v; \rho)$, such that $y \neq G(v)$ (and d is indeed a valid functional decommitment: $\mathsf{FC.FVer}(c,G,y,d) = 1$.) This pair (y,d) can be used to decrypt any ciphertext $\mathsf{ct} \overset{R}{\leftarrow} \mathsf{WS.Enc}(1^\lambda,(c,G),\mathsf{M})$ to $\mathsf{M}' = \{\mathsf{M}[I,y_I]\}_{I\in[l]}$, which breaks semantic security of the witness selector, as $y \neq G(v)$.

<u>Simulatability:</u> We need to prove the computational indistinguishability of the following two distributions:
$$\mathcal{D}_0 = \left\{(c,d) \ : \ \rho \overset{R}{\leftarrow} \{0,1\}^\tau; \ c = \mathsf{FC.Com}(1^\lambda, v; \rho); \ d = \mathsf{FC.FOpen}(c,G,v,\rho)\right\}_{\lambda,v,G} \ ,$$
$$\mathcal{D}_1 = \left\{(c,d) \ : \ (c,d) \overset{R}{\leftarrow} \mathsf{FC.Sim}(1^\lambda, G, G(v))\right\}_{\lambda,v,G} \ .$$

For that, let us introduce the hybrid distribution $\mathcal{H}$, where (c, d) is generated as in $\mathcal{D}_0$ except for $\mu^1_{i,b,j}$ for $i \in [S]$, $b = 1 - G[i]$, and $j \in [|\mathsf{key}[i, b]|]$ that is generated as follows:

$$\mu^1_{i,b,j} = \mathsf{OT.Send}^1(1^\lambda, 0; \rho_{i,b,j}) \ .$$

As $\rho_{i,b,j}$ is never revealed, we have the following claim.

Claim 7.5. *If* OT *is receiver-private, then* $\mathcal{D}_0$ *and* $\mathcal{H}$ *are computationally indistinguishable.*

We also remark that in $\mathcal{H}$, the input labels $\{\mathsf{key}[i, 1 - G[i]]\}$ are never used: only the input labels $\{\mathsf{key}'[i]\} = \{\mathsf{key}[i, G[i]]\}$ are used. The only difference between $\mathcal{H}$ and $\mathcal{D}_1$ is that in $\mathcal{H}$, $(\mathsf{key}', \widehat{C})$ is generated honestly using $\mathsf{GC.Garble}$, while in $\mathcal{D}_1$, it is simulated by $\mathsf{GC.Sim}$. Thus we have the following claim.

Claim 7.6. *If* GC *is simulatable, then* $\mathcal{H}$ *and* $\mathcal{D}_1$ *are computationally indistinguishable.*

Semantic Security of the Witness Selector: We need to prove the computational indistinguishability of the following two distributions:

$$\mathcal{D}_0 = \left\{ ((c, G), (y, d), \rho, \mathsf{ct}) \ : \ \begin{array}{l} \rho \overset{R}{\leftarrow} \{0,1\}^\tau; c = \mathsf{FC.Com}(1^\lambda, v; \rho); \\ y = G(v); d = \mathsf{FC.FOpen}(c, G, y, \rho); \\ \mathsf{ct} \overset{R}{\leftarrow} \mathsf{WS.Enc}(1^\lambda, (c, G), \mathsf{M}) \end{array} \right\}_{\lambda, G, v, \mathsf{M}} ,$$

$$\mathcal{D}_1 = \left\{ ((c, G), (y, d), \rho, \mathsf{ct}) \ : \ \begin{array}{l} \rho \overset{R}{\leftarrow} \{0,1\}^\tau; c = \mathsf{FC.Com}(1^\lambda, v; \rho); \\ y = G(v); d = \mathsf{FC.FOpen}(c, G, y, \rho); \\ \{\mathsf{M}'[I, B] = \mathsf{M}[I, y_I]\}_{I,B} \\ \mathsf{ct} \overset{R}{\leftarrow} \mathsf{WS.Enc}(1^\lambda, (c, G), \mathsf{M}') \end{array} \right\}_{\lambda, G, v, \mathsf{M}} .$$

For that, we consider the following hybrid distributions:

Hybrid $\mathcal{H}_1$: This hybrid is similar to $\mathcal{D}_0$, except that the second flows of the OT protocol $\mu^2_{i,j}$ (generated by $\mathsf{WS.Enc}$) are now generated as follows: for $i \in [S]$ and $j \in [|\mathsf{key}[1, 0]|]$:

$$\mu^2_{i,j} \overset{R}{\leftarrow} \mathsf{OT.Send}^2(\mu^1_{i,G[i],j}, x_{i,j,\mathsf{key}[i,G[i]]_j}, x_{i,j,\mathsf{key}[i,G[i]]_j}) \ .$$

As the first flow $\mu^1_{i,G[i],j}$ is generated as $\mu^1_{i,G[i],j} = \mathsf{OT.Send}^1(1^\lambda, \mathsf{key}[i, G[i]]_j; \rho_{i,b,j}$, we have the following claim.

Claim 7.7. *If* OT *is sender-private, then* $\mathcal{D}_0$ *and* $\mathcal{H}_1$ *are computationally indistinguishable.*

We recall that $x_{i,j,b} = \{\mathsf{okey}_{I,B}[i, j, b]\}_{I,B}$. We remark that in this hybrid, the input labels $\mathsf{okey}_{I,B}[i, j, 1 - \mathsf{key}[i, G[i]]_j]$ are not used. Let us write $\mathsf{okey}'_{I,B}[i, j] = \mathsf{okey}_{I,B}[i, j, \mathsf{key}[i, G[i]]_j]$

Hybrid $\mathcal{H}_2$: This hybrid is similar to $\mathcal{H}_1$ except that the garbled circuits $\widehat{\mathsf{o}C}_{I,B}$ and its input labels $\mathsf{okey}'_{I,B}[i, j]$ are simulated: for every $I \in [l]$ and $B \in \{0, 1\}$:

$$(\mathsf{okey}'_{I,B}, \widetilde{\mathsf{o}C}_{I,B}) \overset{R}{\leftarrow} \mathsf{oGC.Sim}(1^\lambda, \tilde{y}_{I,B}) \ ,$$

and $\widehat{\mathsf{o}C}_{I,B}$ is replaced by $\widetilde{\mathsf{o}C}_{I,B}$, where:

$$\begin{aligned}\tilde{y}_{I,B} &= \mathsf{o}C_{I,B}(\{\mathsf{key}[i,G[i]]\}_{i\in[S]}) \\ &= \begin{cases}\mathsf{M}[I,B] & \text{if } y'_I = B \text{ where } y' = \mathsf{GC.Eval}(\widehat{C},\{\mathsf{key}[i,G[i]]\}_{i\in[S]}), \\ \bot & \text{otherwise.}\end{cases}\end{aligned}$$

We have the following straightforward claim.

Claim 7.8. *If* oGC *is simulatable, then* $\mathcal{H}_1$ *and* $\mathcal{H}_2$ *are computationally indistinguishable.*

We recall that by definition of $\widehat{C}$ and by correctness of garbling:

$$\mathsf{GC.Eval}(\widehat{C},\{\mathsf{key}[i,G[i]]\}_{i\in[S]}) = C(G) = U_\lambda(G,v) = G(v) = y \enspace .$$

In other words:

$$\tilde{y}_{I,B} = \begin{cases}\mathsf{M}[I,B] & \text{if } y_I = B \\ \bot & \text{otherwise.}\end{cases}$$

This hybrid distribution thus only depends on $\{\mathsf{M}[I,y_I]\}_I$.

Hybrid $\mathcal{H}_3$: This hybrid is defined with regards to $\mathcal{D}_1$ exactly as $\mathcal{H}_1$ is defined with regards to $\mathcal{D}_0$.

We have the two following immediate claims.

Claim 7.9. *If* oGC *is simulatable, then* $\mathcal{H}_2$ *and* $\mathcal{H}_3$ *are computationally indistinguishable.*

Claim 7.10. *If* OT *is sender-private, then* $\mathcal{H}_3$ *and* $\mathcal{D}_1$ *are computationally indistinguishable.*

□

8 2-Round Semi-Malicious MPC

Our construction of semi-malicious 2-round MPC is very similar to our construction of semi-honest 2-round MPC in Section 5 with the following two main differences: the functional commitment FC is replaced by a stronger (semi-malicious) equivocable functional commitment eFC and the inner MPC is supposed to be secure against semi-malicious adversaries instead of just semi-honest adversaries.

8.1 Semi-Malicious Equivocable FC with WS

Definition 8.1 ((Semi-Malicious) Equivocable Functional Commitment)**.** Let $\mathcal{G} = \{\mathcal{G}_\lambda\}_{\lambda\in\mathbb{N}}$ be a poly-size circuit class. A *(semi-malicious) equivocable functional commitment* scheme eFC for $\mathcal{G}$ is a tuple of five polynomial-time algorithms $\mathsf{eFC} = (\mathsf{eFC.Com}, \mathsf{eFC.FOpen}, \mathsf{eFC.FVer}, \mathsf{eFC.SimC}, \mathsf{eFC.SimD})$:

Commitment: $c = \mathsf{eFC.Com}(1^\lambda, v; \rho)$ is defined as $\mathsf{FC.Com}$ in Definition 5.1;

Functional Opening: $d = \mathsf{eFC.FOpen}(c, G, y, \rho)$ is defined as $\mathsf{FC.FOpen}$ in Definition 5.1;

Functional Verification: $b = \mathsf{eFC.FVer}(c, G, y, d)$ is defined as $\mathsf{FC.FVer}$ in Definition 5.1;

Commitment Simulation: $(c, \mathsf{trap}) \xleftarrow{R} \mathsf{eFC.SimC}(1^\lambda)$ simulates an (equivocable) commitment c together with a trapdoor trap;

Commitment Equivocation: $d \xleftarrow{R} \mathsf{eFC.SimD}(c, \mathsf{trap}, G, y)$ equivocates the commitment c and output a functional decommitment d of c to y for $G \in \mathcal{G}_\lambda$;

satisfying the following properties:

Correctness: Defined as in Definition 5.1;

Semi-Malicious Functional Binding: For any polynomial-time circuit family $A = \{A_\lambda\}_{\lambda \in \mathbb{N}}$, there exists a negligible function negl, such that for any $\lambda \in \mathbb{N}$, for any $v \in \{0,1\}^n$, for any circuit $G \in \mathcal{G}_\lambda$, for any random tape $\rho \in \{0,1\}^\tau$:

$$\Pr\Big[\mathsf{eFC.FVer}(c, G, y, d) = 1 \text{ and } y \neq G(v) \ : \ c = \mathsf{eFC.Com}(1^\lambda, v; \rho);\ (y, d) \xleftarrow{R} A_\lambda(1^\lambda, v, \rho)\Big] \leq \operatorname{negl}(\lambda) \ ;$$

Simulatability: For any polynomial-time circuit family $A = \{A_\lambda\}_{\lambda \in \mathbb{N}}$, there exists a negligible function negl, such that for any $\lambda \in N$ and for any $v \in \{0,1\}^n$:

$$\left| \Pr\left[A_\lambda(\mathsf{st}, c, d) = 1 \ : \ \begin{array}{l} \rho \xleftarrow{R} \{0,1\}^\tau;\ c \xleftarrow{R} \mathsf{eFC.Com}(1^\lambda, v; \rho); \\ (\mathsf{st}, G) \xleftarrow{R} A_\lambda(c);\ d = \mathsf{eFC.FOpen}(c, G, v, \rho) \end{array}\right] - \Pr\left[A_\lambda(\mathsf{st}, c, d) = 1 \ : \ \begin{array}{l} (c, \mathsf{trap}) \xleftarrow{R} \mathsf{eFC.SimC}(1^\lambda); \\ (\mathsf{st}, G) \xleftarrow{R} A_\lambda(c);\ d \xleftarrow{R} \mathsf{eFC.SimD}(c, \mathsf{trap}, G, G(v)) \end{array}\right] \right| \leq \operatorname{negl}(\lambda) \ .$$

We remark that equivocable functional commitments are actually a generalization of functional commitments. More formally, let $\mathsf{eFC} = (\mathsf{eFC.Com}, \mathsf{eFC.FOpen}, \mathsf{eFC.FVer}, \mathsf{eFC.SimC}, \mathsf{eFC.SimD})$ be an equivocable functional commitment. We can define a polynomial-time algorithm $\mathsf{FC.Sim}$ as follows: $(c, d) \xleftarrow{R} \mathsf{FC.Sim}(1^\lambda, G, y)$ runs

$$(c, \mathsf{trap}) \xleftarrow{R} \mathsf{eFC.SimC}(1^\lambda);\ d \xleftarrow{R} \mathsf{eFC.SimD}(c, \mathsf{trap}, G, y) \ .$$

Then $\mathsf{FC} = (\mathsf{eFC.Com}, \mathsf{eFC.FOpen}, \mathsf{eFC.FVer}, \mathsf{FC.Sim})$ is a functional commitment.

As for functional commitment, let us now define the notion of witness selector associated to eFC.

Definition 8.2. Let $\mathcal{G} = \{\mathcal{G}_\lambda\}_{\lambda \in \mathbb{N}}$ be a poly-size circuit class. Let $\mathsf{eFC} = (\mathsf{eFC.Com}, \mathsf{eFC.FOpen}, \mathsf{eFC.FVer}, \mathsf{eFC.SimC}, \mathsf{eFC.SimD})$ be an *equivocable functional commitment* scheme for $\mathcal{G}$. We define the following *associated non-deterministic oracle family* $\mathcal{O}^{\mathsf{eFC}} = \{\mathcal{O}^{\mathsf{eFC}}_\lambda\}_{\lambda \in \mathbb{N}}$:

$$\mathcal{O}^{\mathsf{eFC}}_\lambda((c, G), (y, d)) = \begin{cases} y & \text{if } \mathsf{eFC.FVer}(c, G, y, d) = 1 \\ \bot & \text{otherwise;} \end{cases}$$

and the following *associated unique-answer distribution* $\mathrm{w}\mathcal{D}^{\mathsf{eFC}} = \{\mathrm{w}\mathcal{D}^{\mathsf{eFC}}_{\lambda,G,v,\rho}\}$ where $\lambda \in \mathbb{N}$, $G \in \mathcal{G}_\lambda$, $v \in \{0,1\}^n$, and $\rho \in \{0,1\}^\tau$:

$$\mathrm{w}\mathcal{D}^{\mathsf{eFC}}_{\lambda,G,v,\rho} = \left\{((c, G),\ (y, d),\ \mathsf{aux} = \rho) \ : \ \begin{array}{l} c = \mathsf{FC.Com}(1^\lambda, v; \rho); \\ y = G(v);\ d = \mathsf{FC.FOpen}(c, G, v, \rho) \end{array}\right\} \ .$$

Finally, a *witness selector* associated to eFC is a witness selector for $(\mathcal{O}^{\mathsf{eFC}}, \mathrm{w}\mathcal{D}^{\mathsf{eFC}})$.

The unique-answer property of $\mathrm{w}\mathcal{D}^{\mathsf{eFC}}$ for $\mathcal{O}^{\mathsf{eFC}}$ follows from the semi-malicious functional binding property of eFC.

We remark that for each (λ, G, v, ρ), the distribution $\mathrm{w}\mathcal{D}^{\mathsf{eFC}}_{\lambda,G,v,\rho}$ only outputs one element with probability 1. In other words, we can see $\mathrm{w}\mathcal{D}^{\mathsf{eFC}}$ as a family instead of an ensemble of distributions. Intuitively, this comes from the fact that the semi-malicious functional binding property is defined with regards to non-uniform adversaries, so they can have hardcoded a collision for a specific tuple (G, v, ρ) for each security parameter λ. This remark significantly simplifies the constructions and proofs.

8.2 Construction of 2-Round Semi-Malicious MPC

Tools and Construction: The tools are similar to the ones used in the construction of the semi-honest 2-round MPC in Section 5.2 with the following differences:

- The inner L-round MPC protocol $\Pi = (\mathsf{Next}, \mathsf{Output})$ for f is supposed to be secure against semi-malicious adversaries.
- The functional commitment scheme FC is replaced by an equivocable functional commitment scheme $\mathsf{eFC} = (\mathsf{eFC.Com}, \mathsf{eFC.FOpen}, \mathsf{eFC.FVer}, \mathsf{eFC.SimC}, \mathsf{eFC.SimD})$.

As in the semi-honest case, the garbled interactive circuit scheme GiC we need for the construction and the equivocable functional encryption scheme eFC can be constructed from Sections 6 and 8.3.

The construction of the protocol is exactly the same.

Unique-Transcript Distribution: The associated unique-transcript distribution $\mathrm{i}\mathcal{D} = \{\mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}\}_{\lambda\in\mathbb{N},\mathrm{id}}$ is actually simpler as it is just a family instead of an ensemble of distributions. More precisely, the index is $\mathrm{id} = (i, \bar{x}, \bar{r}, \bar{m}, \bar{\rho})$ with $\bar{\rho} = \{\rho_j^\ell\}_{j,\ell}$, and $\mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}$ is the following distribution with a unique element (with probability 1):

$$\left\{ (\mathrm{i}C_i,\ \bar{w},\ \bar{\rho} = \{\rho_j^\ell\}_{j,\ell}) \ : \ \begin{array}{l} \forall j \in [N],\ \forall \ell \in [L], \\ \quad c_j^\ell = \mathsf{eFC.Com}(1^\lambda, (x_j, r_j); \rho_j^\ell); \\ \quad G_j^\ell(\star,\star) = \mathsf{Next}_j(\star,\star,\bar{m}^{<\ell}); \\ \quad d_j^\ell = \mathsf{eFC.FOpen}(c_j^\ell, G_j^\ell, (x_j, r_j), \rho_j^\ell); \\ \bar{w} = \{w_j^\ell = (m_j^\ell, d_j^\ell)\}_{j,\ell}; \\ \mathrm{i}C_i \text{ defined in Fig. 4} \end{array} \right\} .$$

We have the following straightforward claim.

Claim 8.3. *The distribution* $\mathrm{i}\mathcal{D}$ *defined above is consistent with* $\mathrm{w}\mathcal{D}^{\mathsf{FC}}$.

Security: We have the following theorem.

Theorem 8.4. *If the inner MPC* $\Pi = (\mathsf{Next}, \mathsf{Output})$ *is correct and secure against semi-malicious adversaries, if the functional commitment scheme* eFC *is correct, semi-malicious functional binding, and simulatable, if the garbled interactive circuit scheme* GiC *is correct and simulatable, then the MPC protocol defined above is correct and secure against semi-malicious adversaries.*

Proof. Correctness is straightforward. Let us prove security against semi-honest adversaries.

We need to exhibit a polynomial-time simulator of the view of any subset $I \subseteq [N]$ of corrupted parties.

Contrary to the semi-honest case in the proof of Theorem 5.3, the simulator cannot start by simulating the inner MPC protocol, as it does not know the input nor the random tapes of the corrupted parties. Instead, it first simulates the commitments of the honest parties in the first round. Then it receives the inputs $x_I = \{x_i\}_{i\in I}$ and random tapes of the inner MPC $r_I = \{r_i\}_{i\in I}$. It uses it to simulates the semi-malicious inner MPC, and equivocate accordingly its commitments in the second round. The remaining of the simulation is similar to the one in the proof of Theorem 5.3.

Let us describe more formally the simulator.

First round: For each honest party P_i with $i \notin I$, simulate the commitments:

$$(c_i^\ell, \mathsf{trap}_i^\ell) \overset{R}{\leftarrow} \mathsf{eFC.SimC}(1^\lambda) \enspace ,$$

for $\ell \in [L]$ and for the circuit $\mathbb{G}_i^\ell$ defined by $G_i^{\ell-1}(\star,\star) = \mathsf{Next}_i(\star,\star,\bar{m}^{<\ell-1})$. Then set the first message $\widetilde{m}_i^1 = \{c_i^\ell\}_{\ell\in[L]}$.

The simulator then receives the commitment of the semi-malicious parties $\{c_i^\ell\}_{i\in I,\ell\in[L]}$ together with the associated random tapes $\{\rho_i^\ell\}_{i\in I,\ell\in[L]}$ and the associated messages, i.e., the inputs $\bar{x}_I$ and randomness $\bar{r}_I$ of the inner MPC: for every $i \in I$ and $\ell \in [L]$,

$$c_i^\ell = \mathsf{eFC.Com}(1^\lambda, (x_i, r_i); \rho_i^\ell) \enspace .$$

The simulator then uses the simulator of the inner MPC to get $\bar{m}$.

Second round: For each honest party P_i with $i \notin I$, equivocate c_i^ℓ for $\ell \in [L]$ as follows:

$$d_i^\ell \overset{R}{\leftarrow} \mathsf{eFC.SimD}(c_i^\ell, \mathsf{trap}_i^\ell, G_i, m_i^\ell) \enspace .$$

It then finishes as in the proof of Theorem 5.3: it computes $\bar{q}^\ell = \{c_j^{\ell-1}, G_j^{\ell-1}\}_{j\in[N]}$, $\bar{w}^\ell = \{m_j^\ell, d_j^\ell\}_{j\in[N]}$, and $o^\ell = (m_i^\ell, d_i^\ell)$, for $\ell \in [L]$, and simulate the garbled interactive circuit:

$$\widetilde{\mathrm{i}C}_i \overset{R}{\leftarrow} \mathsf{GiC.Sim}(1^\lambda, \{\bar{q}^\ell, \bar{w}^\ell, o^\ell\}_{\ell\in[L]}) \enspace .$$

The second message is $\widetilde{m}_i^2 = \widetilde{\mathrm{i}C}_i$.

We now need to prove that the simulation is indistinguishable from the real execution. For that, we consider hybrids. As the proof is very similar to the proof of Theorem 5.3, we skip some steps and compress some hybrids together. More precisely, we define the following hybrids:

Hybrid $\mathcal{H}_1$: This hybrid is similar to the real execution, except that the second messages $\widetilde{m}_j^2 = \widetilde{\mathrm{i}C}_j$ for honest parties P_j for $j \notin I$, which are simulated: $\widetilde{m}_j^2 = \widetilde{\mathrm{i}C}_j = \mathsf{GiC.Sim}(1^\lambda, \{\bar{q}^\ell, \bar{w}^\ell, o^\ell\}_{\ell\in[L]})$.

We have the following claim.

Claim 8.5. *If* GiC *is simulatable, then* $\mathcal{H}_1$ *is computationally indistinguishable from a real execution.*

Hybrid $\mathcal{H}_2$: This hybrid is similar to the previous one, except that instead of generating $\{c_j^\ell, d_j^\ell\}_{j \notin I, \ell \in [L]}$ for honest parties P_j as in the real protocol:

$$c_j^\ell = \mathsf{eFC.Com}(1^\lambda, (x_j, r_j); \rho_j^\ell);\ d_j^\ell = \mathsf{eFC.FOpen}(c_j^\ell, G_j^\ell, r_j, \rho_j^\ell)\ ,$$

(where ρ_j^ℓ is a uniform random tape), we simulate them:

$$(c_j^\ell, \mathsf{trap}_j^\ell) \overset{R}{\leftarrow} \mathsf{eFC.SimC}(1^\lambda, (x_j, r_j); \rho_j^\ell);\ d_j^\ell \overset{R}{\leftarrow} \mathsf{eFC.SimD}(c_j^\ell, \mathsf{trap}_j^\ell, G_j^\ell, m_j^\ell)\ .$$

We have the following straightforward claim.

Claim 8.6. *If* eFC *is simulatable, then* $\mathcal{H}_1$ *and* $\mathcal{H}_2$ *are computationally indistinguishable.*

Furthermore, the only difference between $\mathcal{H}_2$ and a simulated execution is that in the latter, the inner MPC messages $\bar{m}$ are simulated after seeing the random tapes and inputs of the semi-malicious adversary (for the inner MPC). Thus, we have the following claim

Claim 8.7. *If the inner MPC is secure against semi-malicious adversaries, then* $\mathcal{H}_2$ *and a simulated execution are computationally indistinguishable.*

□

8.3 Equivocable FC with WS from 2-Round Semi-Malicious OT

To conclude the construction of semi-malicious 2-round MPC, we need to construct an equivocable functional commitment with witness selector from semi-malicious 2-round OT.

Semi-Malicious 2-Round OT: Let us first define the notion of semi-malicious 2-round OT.

Definition 8.8. A *semi-malicious 2-round oblivious transfer (OT)* is a 2-round oblivious OT (see Definition 7.3) satisfying the following additional property:

Semi-Malicious Sender Privacy: The following two distributions are computationally indistinguishable:

$$\{\mathsf{OT.Send}^2(\mu^1, x_0, x_1)\ :\ \mu^1 = \mathsf{OT.Send}^1(1^\lambda, \sigma; \rho)\}_{\lambda, \sigma, x_0, x_1, \rho}\ ,$$
$$\{\mathsf{OT.Send}^2(\mu^1, x_\sigma, x_\sigma)\ :\ \mu^1 = \mathsf{OT.Send}^1(1^\lambda, \sigma; \rho)\}_{\lambda, \sigma, x_0, x_1, \rho}\ .$$

We remark that semi-honest sender privacy is clearly implied by semi-malicious sender privacy. The only difference between the two notions is that the former notion just needs to hold when the first flow is honestly generated using a uniform random tape $\rho \overset{R}{\leftarrow} \{0,1\}^\tau$, while the latter one needs to hold for any random tape $\rho \in \{0,1\}^\tau$.

Construction of Equivocable Functional Commitment: Let $\mathcal{G} = \{\mathcal{G}_\lambda\}_{\lambda \in \mathbb{N}}$ be a S-size circuit class (where S is polynomial in λ). To construct an equivocable functional commitment scheme $\mathsf{eFC} = (\mathsf{eFC.Com}, \mathsf{eFC.FOpen}, \mathsf{eFC.FVer}, \mathsf{eFC.SimC}, \mathsf{eFC.SimD})$ with an associated witness selector $\mathsf{WS} = (\mathsf{WS.Enc}, \mathsf{WS.Dec})$, we rely on the same tools as in Section 7, except that we suppose the 2-round OT to also be semi-malicious sender-private.

The construction is very similar to the one of Section 7. The semi-malicious binding property easily follows from the semi-malicious sender privacy property of the OT. The main difficulty is to add the equivocation property. In the construction of Section 7, first flows of the OT protocol are used to commit to the input labels of a garbled circuit of $U(\star, v)$. The issue is that since the garbled circuit is in the clear and the input labels are committed (in a possibly statistically binding way), there is no way to do any equivocation. The idea is to commit both the input labels and the garbled circuit of $U(\star, v)$ in an equivocable way (and compatible with a witness selector): for each bit β of the input labels and of the garbled circuit, we generate two first OT flows both for the selector bit β. The associated "decommitment" is the random tape used to generate the $(\beta + 1)$-th (first) OT flow (i.e., the first one if $\beta = 0$ and the second one if $\beta = 1$). In a simulated commitment generated by $\mathsf{eFC.SimC}$, for each bit, the first (first) OT flow is generated for the selector bit 0, while the second (first) OT flow is generated is generated for the selector bit 1. But a commitment generated by a semi-malicious adversary remains binding, as even a semi-malicious adversary has to use the same selector bit for both OTs.

More precisely, the construction is as follows:

Commitment: $c = \mathsf{eFC.Com}(1^\lambda, v; \rho)$ commits to $v \in \{0,1\}^n$ as follows:

1. Generate input labels $\mathsf{key} \overset{R}{\leftarrow} \mathsf{GC.Gen}(1^\lambda)$ (using a random tape derived from ρ).
2. Garble $C = U_\lambda(\star, v)$, which is the universal circuit partially evaluated on v: $\widehat{C} \overset{R}{\leftarrow} \mathsf{GC.Garble}(\mathsf{key}, C)$.
3. For each $k \in [|\widehat{C}|]$, for each bit $b' \in \{0,1\}$, generate a first flow $\mu^1_{k,b'} = \mathsf{OT.Send}^1(1^\lambda, \widehat{C}[k]; \rho_{k,b'})$, where $\widehat{C}[k]$ is the k-th bit of the garbled circuit $\widehat{C}$ (seen as a bitstring) and where the random tape $\rho_{k,b'}$ is derived from ρ.
4. For each $i \in [S]$, for each bit $b \in \{0,1\}$, for each $j \in [|\mathsf{key}[i,b]|]$, for each bit $b' \in \{0,1\}$, generate a first flow $\mu^1_{i,b,j,b'} = \mathsf{OT.Send}^1(1^\lambda, \mathsf{key}[i,b]_j; \rho_{i,b,j,b'})$, where $\mathsf{key}[i,b]_j$ is the j-th bit of the input label $\mathsf{key}[i,b]$ and where the random tape $\rho_{i,b,j,b'}$ is derived from ρ.

And returns:

$$c = (\{\mu^1_{k,b'}\}_{k,b'}, \{\mu^1_{i,b,j,b'}\}_{i,b,j,b'}) \enspace .$$

Functional Opening: $d = \mathsf{eFC.FOpen}(c, G, v, \rho)$ derives the functional decommitment d of c to $y = G(v) = U_\lambda(G, v)$ as follows:

$$d = (\widehat{C}, \{\rho'_k\}_k, \{\mathsf{key}'[i], \{\rho'_{i,j}\}_j\}_{i \in [S]}) \enspace ,$$

where $\rho'_k = \rho_{k,\widehat{C}[k]}$, $\mathsf{key}'[i] = \mathsf{key}[i, G[i]]$, and $\rho'_{i,j} = \rho_{i,G[i],j,\mathsf{key}'[i]}$.

Functional Verification: $\mathsf{eFC.FVer}(c, G, y, d)$ returns 1, if and only if for all $i \in [S]$ and $j \in [|\mathsf{key}'[i]|]$ and $k \in [|\widehat{C}|]$:

$$\mu^1_{k,\widehat{C}[k]} = \mathsf{OT.Send}^1(1^\lambda, \widehat{C}[k]; \rho'_k) \enspace ,$$
$$\mu^1_{i,G[i],j,\mathsf{key}'[i]_j} = \mathsf{OT.Send}^1(1^\lambda, \mathsf{key}'[i]_j; \rho'_{i,j}) \enspace ,$$
$$y = \mathsf{GC.Eval}(\widehat{C}, \mathsf{key}') \enspace .$$

Simulation: $(c, \mathsf{trap}) \overset{R}{\leftarrow} \mathsf{eFC.SimC}(1^\lambda)$ generates the simulated commitment c as $\mathsf{eFC.Com}$, except that:

$$\mu^1_{k,b'} = \mathsf{OT.Send}^1(1^\lambda, b'; \rho_{k,b'}) \qquad \text{and} \qquad \mu^1_{i,b,j,b'} = \mathsf{OT.Send}^1(1^\lambda, b'; \rho_{i,b,j,b'}) \enspace ,$$

i.e., the first flows $\mu^1_{k,b'}$ and $\mu^1_{i,b,j,b'}$ are for the selector bit b' instead of $\widehat{C}[k]$ and $\mathsf{key}[i,b]_j$. The trapdoor trap is the random tape.

Equivocation: $d \overset{R}{\leftarrow} \mathsf{eFC.SimD}(c, \mathsf{trap}, G, y)$ equivocate the commitment c by simulating $(\mathsf{key}', \widetilde{C}) \overset{R}{\leftarrow} \mathsf{GC.Sim}(1^\lambda, y)$ and then generating the functional decommitment d similarly to $\mathsf{FC.FOpen}$ as follows:

$$d = \left(\widetilde{C},\ \{\rho'_k\}_k,\ \left\{\mathsf{key}'[i], \{\rho'_{i,j}\}_j\right\}_{i \in [S]}\right) \enspace ,$$

where $\rho'_k = \rho_{k,\widetilde{C}[k]}$, $\mathsf{key}'[i] = \mathsf{key}[i, G[i]]$, $\rho'_{i,j} = \rho_{i,G[i],j,\mathsf{key}'[i]}$.

Encryption: $\mathsf{ct} \overset{R}{\leftarrow} \mathsf{WS.Enc}(1^\lambda, (c, G), \mathsf{M})$ encrypts the messages $\mathsf{M} = \{\mathsf{M}[I,B]\}_{I,B}$ for $q = (c, G)$ into ct as follows:

1. For every $I \in [l]$ and $B \in \{0,1\}$, create the circuit:

$$\mathsf{o}C_{I,B}(\widehat{C}, \mathsf{key}') = \begin{cases} \mathsf{M}[I,B] & \text{if } y_I = B \text{ where } y = \mathsf{GC.Eval}(\widehat{C}, \mathsf{key}'), \\ \bot & \text{otherwise.} \end{cases}$$

2. For every $I \in [l]$ and $B \in \{0,1\}$, garble this circuit: $\mathsf{okey}_{I,B} \overset{R}{\leftarrow} \mathsf{oGC.Gen}(1^\lambda)$ and $\widehat{\mathsf{o}C}_{I,B} \overset{R}{\leftarrow} \mathsf{oGC.Garble}(\mathsf{okey}, \mathsf{o}C_{I,B})$; we write $\mathsf{okey}_{I,B}[k,b]$ (resp., $\mathsf{okey}_{I,B}[i,j,b]$) the key corresponding to the k-th bit of the input $\widehat{C}$ of $\mathsf{o}C_{I,B}$ (resp., the j-th bit of the input $\mathsf{key}'[i]$ of $\mathsf{o}C_{I,B}$) being b.
3. Define the OT messages: $x_{k,b} = \{\mathsf{okey}_{I,B}[k,b]\}_{I,B}$ and $x_{i,j,b} = \{\mathsf{okey}_{I,B}[i,j,b]\}_{I,B}$.
4. Compute the second flows of the OT corresponding to the first flows $\mu^1_{k,b'}$ and $\mu^1_{i,G[i],j,b'}$:

$$\begin{aligned} \mu^2_{k,0} &\overset{R}{\leftarrow} \mathsf{OT.Send}^2(\mu^1_{k,0}, x_{i,j,0}, \bot) \enspace , \\ \mu^2_{k,1} &\overset{R}{\leftarrow} \mathsf{OT.Send}^2(\mu^1_{k,1}, \bot, x_{i,j,1}) \enspace , \\ \mu^2_{i,j,0} &\overset{R}{\leftarrow} \mathsf{OT.Send}^2(\mu^1_{i,G[i],j,0}, x_{i,j,0}, \bot) \enspace , \\ \mu^2_{i,j,1} &\overset{R}{\leftarrow} \mathsf{OT.Send}^2(\mu^1_{i,G[i],j,1}, \bot, x_{i,j,1}) \enspace , \end{aligned}$$

   where $\bot$ is an arbitrary message.

And return

$$\mathsf{ct} = \left(\{\widehat{\mathsf{o}C}_{I,B}\}_{I \in [l], B \in \{0,1\}}, \{\mu^2_{k,b'}\}_{k,b'}, \{\mu^2_{i,j,b'}\}_{i,j,b'}\right) \enspace .$$

Decryption: $\mathsf{M} = \mathsf{WS.Dec}(\mathsf{ct}, (y, d))$ decrypts ct as follows:

1. For every k, compute:

$$\{\mathsf{okey}'_{I,B}[k]\}_{I,B} = x_{k,\widehat{C}[k]} = \mathsf{OT.Output}(\mu^2_{k,\widehat{C}[k]}, \widehat{C}[k], \rho'_k) \enspace .$$

2. For every i and j, compute:

$$\{\mathsf{okey}'_{I,B}[i,j]\}_{I,B} = x_{i,j,\mathsf{key}'[i]_j} = \mathsf{OT.Output}(\mu^2_{i,j,\mathsf{key}'[i]_j}, \mathsf{key}'[i]_j, \rho'_{i,j}) \ .$$

3. For every $I \in [l]$ and $B = y_I$, evaluate the garble circuit $\widehat{\mathsf{o}C}_{I,B}$:

$$\mathsf{M}[I,B] = \mathsf{oGC.Eval}(\widehat{\mathsf{o}C}_{I,B}, (\{\mathsf{okey}'_{I,B}[k]\}_k, \{\mathsf{okey}'_{I,B}[i,j]\}_{i,j}))$$

and return $\mathsf{M} = \{\mathsf{M}[I,y_I]\}_{I\in[l]}$.

We have the following theorem.

Theorem 8.9. *If* OT *is correct, receiver-private, and semi-malicious sender-private, then the equivocable functional commitment scheme* eFC *defined above is correct, semi-malicious functionally binding, and simulatable. Furthermore, the associated witness selector* WS *is correct and semantically secure.*

Proof. As in the proof of Theorem 7.4, correctness is straightforward and semi-malicious functional binding follows from the semantic security of the witness selector. Furthermore simulatability of eFC and semantic security of the witness selector WS can be proven similarly as in the proof of Theorem 7.4. □

9 k-Round Semi-Malicious MPC from k-Round OT

In this section, we generalize our constructions of 2-round MPC from 2-round OT, to constructions of k-round MPC from k-round OT, for any $k \geq 2$.

Let us first present an overview of the techniques for the semi-honest case. The semi-malicious case is very similar. We recall that our 2-round MPC construction makes use of a functional commitment scheme (with an associated witness selector) built from 2-round OT. k-round OT does not seem sufficient to construct a functional commitment scheme (even without an associated witness selector). However, if we generalize the notion of functional commitment to allow for an interactive $(k-1)$-round commitment phase, then we can achieve it from k-round OT: the construction is similar to the one in Section 7 except that the commitment consists in the first $(k-1)$ flows of the OT, where the sender of the OT (i.e., the receiver of the commitment) uses random messages. Then, the associated witness selector proceeds similarly as in Section 7, except that instead of using the second flow of the OT to send the input messages $x_{i,j,b} = \{\mathsf{okey}_{I,B}[i,j,b]\}_{I,B}$ (for some indices i,j, and for both bits $b \in \{0,1\}$), it sends the last flow (i.e., the k-th one) of the OT together with the XOR of these messages $x_{i,j,b} = \{\mathsf{okey}_{I,B}[i,j,b]\}_{I,B}$ and the random messages chosen previously.

We remark that to generate the last flow of the OT protocol, the encryptor needs to know the randomness used by the receiver of the commitment (i.e., the sender of the OT protocol). The notions of witness selector and garbled interactive circuit need therefore to be adapted to allow for this extra information that we call *designated-encryptor information* denc.

9.1 Semi-Honest Construction

In this section, we present more formally the semi-honest construction.

9.1.1 Designated-Encryptor Witness Selector

We define a *unique-answer distribution with designated-encryptor information* $\mathrm{w}\mathcal{D} = \{\mathrm{w}\mathcal{D}_{\lambda,\mathsf{id}}\}_{\lambda\in\mathbb{N},\mathsf{id}}$ similarly as a unique-answer distribution (Definition 6.1), except that $\mathrm{w}\mathcal{D}_{\lambda,\mathsf{id}}$ is a distribution over tuples $(q, w, \mathsf{aux}, \mathsf{denc})$, instead of just (q, q, aux). The required properties are straightforward extensions:

Non-$\perp$ Answer: For any $\lambda \in \mathbb{N}$, any index $\mathsf{id} \in \{0,1\}^{\mathrm{poly}(\lambda)}$, and any $(q, w, \mathsf{aux}, \mathsf{denc})$ in the support of $\mathrm{w}\mathcal{D}_{\lambda,\mathsf{id}}$, $\mathcal{O}_\lambda(q, w) \neq \perp$.

Computationally Unique Answer: For any poly-size circuit family $A = \{A_\lambda\}_{\lambda\in\mathbb{N}}$, for any sequence of indices $\{\mathsf{id}_\lambda\}_\lambda$, there exists a negligible function negl, such that for any $\lambda \in \mathbb{N}$:

$$\Pr\Big[\mathcal{O}_\lambda(q, w') \neq \perp \text{ and } \mathcal{O}_\lambda(q, w') \neq \mathcal{O}_\lambda(q, w) \ : \\ (q, w, \mathsf{aux}, \mathsf{denc}) \xleftarrow{R} \mathrm{w}\mathcal{D}_{\lambda,\mathsf{id}_\lambda};\ w' \xleftarrow{R} A_\lambda(q, w, \mathsf{aux})\Big] \leq \mathrm{negl}(\lambda) \ .$$

We then define a *designated-encryptor witness selector* similarly to a witness selector $\mathsf{WS} = (\mathsf{WS.Enc}, \mathsf{WS.Dec})$ (Definition 6.2), except that $\mathsf{WS.Enc}$ takes an additional input denc. Correctness and semantic security are modified accordingly. The distinguisher for semantic security does not directly see denc: it only sees it through $\mathsf{WS.Enc}(1^\lambda, q, \mathsf{M}, \mathsf{denc})$ or $\mathsf{WS.Enc}(1^\lambda, q, \mathsf{M}', \mathsf{denc})$. This is important as in our construction denc would reveal M or M'. More formally, the security properties are defined as follows:

Correctness: For any security parameter $\lambda \in \mathbb{N}$, for any index id, for any $(q, w, \mathsf{aux}, \mathsf{denc})$ in the support of $\mathrm{w}\mathcal{D}_{\lambda,\mathsf{id}}$, for any messages $\mathsf{M} = \{\mathsf{M}[i, b]\}_{i,b}$, for $a = \mathcal{O}(q, w)$:

$$\Pr\Big[\mathsf{WS.Dec}(\mathsf{WS.Enc}(1^\lambda, q, \mathsf{M}, \mathsf{denc}),\ w) = \{\mathsf{M}[i, a_i]\}_{i\in[l]}\Big] = 1 \ ;$$

Semantic Security: The following two distributions are indistinguishable:

$$\big\{(q, w, \mathsf{aux}, \mathsf{WS.Enc}(1^\lambda, q, \mathsf{M}, \mathsf{denc})) \ : \ (q, w, \mathsf{aux}, \mathsf{denc}) \xleftarrow{R} \mathrm{w}\mathcal{D}_{\lambda,\mathsf{id}} \ \big\}_{\lambda,\mathsf{id},\mathsf{M}} \ ,$$

$$\left\{ (q, w, \mathsf{aux}, \mathsf{WS.Enc}(1^\lambda, q, \mathsf{M}', \mathsf{denc})) \ : \ \begin{array}{l} (q, w, \mathsf{aux}, \mathsf{denc}) \xleftarrow{R} \mathrm{w}\mathcal{D}_{\lambda,\mathsf{id}}; \\ a = \mathcal{O}_\lambda(q, w); \\ \{\mathsf{M}'[i, b]\}_{i,b} = \{\mathsf{M}[i, a_i]\}_{i,b} \end{array} \right\}_{\lambda,\mathsf{id},\mathsf{M}} \ .$$

In the sequel, we omit the adjective "designated-encryptor," when it is clear from context.

9.1.2 Designated-Encryptor Garbled Interactive Circuit

Definition: We extend similarly unique-transcript distributions and garbled interactive circuit schemes. We define a *unique-transcript distribution with designated encryptor information* $\mathrm{i}\mathcal{D} = \{\mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}\}_{\lambda\in\mathbb{N},\mathrm{id}}$ similarly as a unique-transcript distribution except that $\mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}$ is a distribution over tuples $(\mathsf{i}C, \bar{w}, \mathsf{aux}, \overline{\mathsf{denc}})$ (where $\overline{\mathsf{denc}} = \{\mathsf{denc}_k^\ell\}_{\ell,k}$ has the same number of elements as $\bar{w}$) instead of just $(\mathsf{i}C, \bar{w}, \mathsf{aux})$. The required properties (valid execution and computationally unique transcript) are otherwise the same.

Consistency between unique-transcript distributions and unique-answer distributions with designated encryptor information is defined as consistency in Definition 6.3, with the additional requirements that $\mathsf{denc}^{\ell^\star}_{k^\star}$ from the unique-transcript distribution matches (i.e., is equal to) denc from the unique-answer distribution and that $\mathsf{denc}^{\ell^\star}_{k^\star}$ is not "used only once". More formally, the distributions $\mathrm{i}\mathcal{D}$ and $\mathrm{w}\mathcal{D}$ are consistent if for any sequence of indices $\{\ell^\star\}_{\lambda,\mathrm{id}}$ and $\{k^\star\}_{\lambda,\mathrm{id}}$, there exists two probabilistic polynomial-time functions g and h, such that the following two distributions are identical:

$$\left\{((\mathrm{i}C, \bar{w}, \mathsf{aux}, \{\mathsf{denc}^\ell_k\}_{\ell\neq\ell^\star,\, k\neq k^\star}),\ (q^{\ell^\star}_{k^\star}, w^{\ell^\star}_{k^\star}, \mathsf{denc}^{\ell^\star}_{k^\star}))\ :\ \begin{array}{l}(\mathrm{i}C, \bar{w}, \mathsf{aux}, \overline{\mathsf{denc}}) \xleftarrow{R} \mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}};\\ \{\bar{q}^\ell, \bar{a}^\ell, o^\ell\}_{\ell\in[L]} = \mathsf{trans}(\mathrm{i}C, \mathcal{O}, \bar{w})\end{array}\right\}_{\lambda,\mathrm{id}},$$

$$\{(h(q, w, \mathsf{aux}),\ (q, w, \mathsf{denc}))\ :\ (q, w, \mathsf{aux}, \mathsf{denc}) \xleftarrow{R} \mathrm{w}\mathcal{D}_{\lambda, g(1^\lambda,\mathrm{id})}\}_{\lambda,\mathrm{id}}.$$

It is important to remark that the function h does not take as input denc.

We then define a *designated-encryptor garbled interactive circuit* scheme similarly to a garbled interactive circuit scheme $\mathsf{GiC} = (\mathsf{GiC.Garble}, \mathsf{GiC.Eval}, \mathsf{GiC.Sim})$ (Definition 6.2), except that $\mathsf{GiC.Garble}$ and $\mathsf{GiC.Sim}$ take an additional input $\overline{\mathsf{denc}}$. Correctness and simulatability are modified accordingly. The distinguisher for simulatability does not directly see $\overline{\mathsf{denc}}$: it only sees it through $\mathsf{GiC.Garble}(1^\lambda, \mathrm{i}C, \overline{\mathsf{denc}})$ or $\mathsf{GiC.Sim}(1^\lambda, \mathsf{trans}(\mathrm{i}C, \mathcal{O}_\lambda, \bar{w}), \overline{\mathsf{denc}})$. More formally, the security properties are defined as follows:

Correctness: For any $\lambda \in \mathbb{N}$, any index $\mathrm{id} \in \{0,1\}^{\mathrm{poly}(\lambda)}$, any $(\mathrm{i}C, \bar{w}, \mathsf{aux}, \overline{\mathsf{denc}})$ in the support of $\mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}}$, it holds that

$$\Pr\left[\{\mathsf{GiC.Eval}(\widehat{\mathrm{i}C}, \bar{w}^{<\ell})\}_{\ell\in[L]} = \mathsf{out}(\mathrm{i}C, \mathcal{O}_\lambda, \bar{w})\ :\ \widehat{\mathrm{i}C} \xleftarrow{R} \mathsf{GiC.Garble}(1^\lambda, \mathrm{i}C, \overline{\mathsf{denc}})\right] = 1\ ;$$

Simulatability: The following two distributions are computationally indistinguishable:

$$\left\{(\mathrm{i}C, \bar{w}, \mathsf{aux}, \widehat{\mathrm{i}C})\ :\ \begin{array}{l}(\mathrm{i}C, \bar{w}, \mathsf{aux}, \overline{\mathsf{denc}}) \xleftarrow{R} \mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}};\\ \widehat{\mathrm{i}C} \xleftarrow{R} \mathsf{GiC.Garble}(1^\lambda, \mathrm{i}C, \overline{\mathsf{denc}})\end{array}\right\}_{\lambda,\mathrm{id}},$$

$$\left\{(\mathrm{i}C, \bar{w}, \mathsf{aux}, \widetilde{\mathrm{i}C})\ :\ \begin{array}{l}(\mathrm{i}C, \bar{w}, \mathsf{aux}, \overline{\mathsf{denc}}) \xleftarrow{R} \mathrm{i}\mathcal{D}_{\lambda,\mathrm{id}};\\ \widetilde{\mathrm{i}C} \xleftarrow{R} \mathsf{GiC.Sim}(1^\lambda, \mathsf{trans}(\mathrm{i}C, \mathcal{O}_\lambda, \bar{w}), \overline{\mathsf{denc}})\end{array}\right\}_{\lambda,\mathrm{id}}.$$

Construction from Designated-Encryptor WS: We can construct designated-encryptor garbled interactive circuits for a unique-transcript distribution $\mathrm{i}\mathcal{D}$ from designated-encryptor witness selector for a unique-answer distribution $\mathrm{w}\mathcal{D}$ that is consistent with $\mathrm{i}\mathcal{D}$, in a similar way as in Section 6.

The only subtlety compared to the proof of Theorem 6.5 is to ensure that when using semantic security of WS to prove indistinguishability of hybrids, the associated designated-encrypted information denc is only used to generate the corresponding WS ciphertext $\mathsf{WS.Enc}(1^\lambda, q, \mathsf{M}, \mathsf{denc})$ and is not used anywhere else. This is enforced by the definition of consistency between $\mathrm{i}\mathcal{D}$ and $\mathrm{w}\mathcal{D}$, which ensures that each denc is only used once.

9.1.3 k-Round Interactive Functional Commitment

Definition: We define a k-round functional commitment scheme as follows.

Definition 9.1 (k-Round Interactive Functional Commitment)**.** Let $\mathcal{G} = \{\mathcal{G}_\lambda\}_{\lambda \in \mathbb{N}}$ be a poly-size circuit class. A *k-round interactive functional commitment* scheme iFC for $\mathcal{G}$ is a tuple of two polynomial-time interactive Turing machines and three polynomial-time algorithms $\mathsf{iFC} = (\mathsf{iFC.S}, \mathsf{iFC.R}, \mathsf{iFC.FOpen}, \mathsf{iFC.FVer}, \mathsf{iFC.Sim})$:

Commitment is performed via a $(k-1)$-round interaction between a sender $\mathsf{iFC.S}$ on input the message to be committed v and with a random tape $\rho \in \{0,1\}^\tau$ and a receiver $\mathsf{iFC.R}$ on random tape $\rho' \in \{0,1\}^{\tau'}$. The resulting commitment $c = \langle \mathsf{iFC.S}(1^\lambda, v; \rho), \mathsf{iFC.R}(1^\lambda; \rho') \rangle$ is the transcript of the interaction;

Functional Opening: $d = \mathsf{iFC.FOpen}(c, G, y, \rho)$ is defined as $\mathsf{FC.FOpen}$ in Definition 5.1;

Functional Verification: $b = \mathsf{iFC.FVer}(c, G, y, d)$ is defined as $\mathsf{FC.FVer}$ in Definition 5.1;

Simulation: $(c, d, \rho') = \mathsf{iFC.Sim}(1^\lambda, G, y)$ is defined as $\mathsf{FC.Sim}$ in Definition 5.1, except that it also outputs the randomness ρ' used by the receiver;

satisfying the following properties:

Correctness: For any security parameter $\lambda \in \mathbb{N}$, for any $v \in \{0,1\}^n$, for any circuit $G \in \mathcal{G}_\lambda$, for any $\rho \in \{0,1\}^\tau$, for any $\rho' \in \{0,1\}^{\tau'}$, it holds that, if $c = \langle \mathsf{iFC.S}(1^\lambda, v; \rho), \mathsf{iFC.R}(1^\lambda; \rho') \rangle$, then:

$$\mathsf{iFC.FVer}(c, G, G(v), \mathsf{iFC.FOpen}(1^\lambda, G, v, \rho)) = 1 \ ;$$

Semi-Honest Functional Binding: For any polynomial-time circuit family $A = \{A_\lambda\}_{\lambda \in \mathbb{N}}$, there exists a negligible function negl, such that for any $\lambda \in \mathbb{N}$, for any $v \in \{0,1\}^n$, for any circuit $G \in \mathcal{G}_\lambda$:

$$\Pr\Big[\mathsf{FC.FVer}(c, G, y, d) = 1 \text{ and } y \neq G(v) \ : \ \rho \overset{R}{\leftarrow} \{0,1\}^\tau;\ \rho' \overset{R}{\leftarrow} \{0,1\}^{\tau'}; \\ c = \langle \mathsf{iFC.S}(1^\lambda, v; \rho), \mathsf{iFC.R}(1^\lambda; \rho') \rangle;\ (y, d) \overset{R}{\leftarrow} A_\lambda(1^\lambda, c, v, \rho)\Big] \leq \mathrm{negl}(\lambda) \ ;$$

Semi-Honest Simulatability: The following two distributions are computationally indistinguishable:

$$\left\{(c, d, \rho') \ : \ \begin{array}{l} \rho \overset{R}{\leftarrow} \{0,1\}^\tau;\ \rho' \overset{R}{\leftarrow} \{0,1\}^{\tau'};\ c = \langle \mathsf{iFC.S}(1^\lambda, v; \rho), \mathsf{iFC.R}(1^\lambda; \rho') \rangle; \\ d = \mathsf{FC.FOpen}(c, G, v, \rho) \end{array} \right\}_{\lambda, G, v} ,$$
$$\{(c, d, \rho') \ : \ (c, d, \rho') \overset{R}{\leftarrow} \mathsf{FC.Sim}(1^\lambda, G, G(v)) \ \}_{\lambda, G, v} \ .$$

The associated non-deterministic oracle family $\mathcal{O}^{\mathsf{iFC}}$ and unique-answer distribution (with designated-encryptor information) $\mathrm{w}\mathcal{D}^{\mathsf{iFC}}$ for iFC are defined similarly to the ones for functional commitments in Definition 5.2, except that $c = \mathsf{FC.Com}(1^\lambda, v; \rho)$ is replaced by $c = \langle \mathsf{iFC.S}(1^\lambda, v; \rho), \mathsf{iFC.R}(1^\lambda; \rho') \rangle$ and that denc is set to ρ'.

Construction from k-Round OT: We can construct a k-round functional commitment with an associated designated-encryptor witness selector from a k-round OT with semi-honest sender privacy and semi-honest receiver privacy. Let us first define more formally such OT protocols.

Definition 9.2. A *k-round oblivious transfer (OT)* is a tuple of two polynomial-time interactive Turing machines $\mathsf{OT} = (\mathsf{OT.S}, \mathsf{OT.R})$ where $(t, x) = \langle \mathsf{OT.S}(1^\lambda, x_0, x_1; \rho), \mathsf{OT.R}(1^\lambda, \sigma; \rho') \rangle$ is the pair composed of the transcript t and the output x of the receiver after interaction between the sender $\mathsf{OT.S}$ with messages $(x_0, x_1) \in (\{0,1\}^k)^2$ and randomness $\rho \in \{0,1\}^\tau$, and the receiver $\mathsf{OT.R}$ with selection bit $\sigma \in \{0,1\}$ and randomness $\rho' \in \{0,1\}^{\tau'}$, satisfying the following properties:

Correctness: For any security parameter $\lambda \in \mathbb{N}$, for any selection bit $\sigma \in \{0,1\}$, for any messages $(x_0, x_1) \in (\{0,1\}^k)^2$, for any $\rho \in \{0,1\}^\tau$ and $\rho \in \{0,1\}^{\tau'}$, it holds that:

$$\Pr\left[x_\sigma = x \ : \ \rho \xleftarrow{R} \{0,1\}^\tau;\ \rho' \xleftarrow{R} \{0,1\}^{\tau'};\ (t,x) = \langle \mathsf{OT.S}(1^\lambda, x_0, x_1; \rho), \mathsf{OT.R}(1^\lambda, \sigma; \rho') \rangle \right] = 1 \ ;$$

Semi-Honest Receiver Privacy: The following two distributions are computationally indistinguishable:

$$\{(t,\rho) \ : \ \rho \xleftarrow{R} \{0,1\}^\tau;\ \rho' \xleftarrow{R} \{0,1\}^{\tau'};\ (t,x) = \langle \mathsf{OT.S}(1^\lambda, x_0, x_1; \rho), \mathsf{OT.R}(1^\lambda, 0; \rho') \rangle\}_{\lambda, x_0, x_1} \ ,$$
$$\{(t,\rho) \ : \ \rho \xleftarrow{R} \{0,1\}^\tau;\ \rho' \xleftarrow{R} \{0,1\}^{\tau'};\ (t,x) = \langle \mathsf{OT.S}(1^\lambda, x_0, x_1; \rho), \mathsf{OT.R}(1^\lambda, 1; \rho') \rangle\}_{\lambda, x_0, x_1} \ ;$$

Semi-Honest Sender Privacy: The following two distributions are computationally indistinguishable:

$$\{(t,x,\rho') \ : \ \rho \xleftarrow{R} \{0,1\}^\tau;\ \rho' \xleftarrow{R} \{0,1\}^{\tau'};\ (t,x) = \langle \mathsf{OT.S}(1^\lambda, x_0, x_1; \rho), \mathsf{OT.R}(1^\lambda, \sigma; \rho') \rangle\}_{\lambda, x_0, x_1, \sigma} \ ,$$
$$\{(t,x,\rho') \ : \ \rho \xleftarrow{R} \{0,1\}^\tau;\ \rho' \xleftarrow{R} \{0,1\}^{\tau'};\ (t,x) = \langle \mathsf{OT.S}(1^\lambda, x_\sigma, x_\sigma; \rho), \mathsf{OT.R}(1^\lambda, \sigma; \rho') \rangle\}_{\lambda, x_0, x_1, \sigma} \ .$$

Then, the construction is similar to the one in Section 7, except that the commitment uses the first $(k-1)$ flows of the OT protocols instead of the first flow, as explained in the overview of this section. The commitment receiver plays the role of the OT sender, while the commitment sender plays the role of the OT receiver. The designated-encryptor information consists of all the random tapes of the OT senders, in all the OT protocols. The garbled circuits used in the constructions can be sent in any of the flows going from the OT receiver to the OT sender.

Witness selector encryption uses the last flows of the OT protocols. We recall that the messages to be sent in an OT can always be input just in the last flow, because the OT sender can always use random messages at the beginning and provide the XOR of these random messages and of the real messages in the last flow.

9.1.4 Semi-Honest k-Round MPC

Combining the previous tools, we can easily construct a semi-honest k-round MPC from any semi-honest k-round OT, following the construction in Section 5, with the following difference: each party P_i needs to commit its input and random tape L times to each party (including itself to make the protocol more symmetric for the sake of simplicity; that means each party does $L \cdot N$ commitments), instead of just L times in total. The reason is that now a commitment is done to a specific party, the commitment receiver, and only this commitment receiver knows the associated designated-encryptor information.

9.2 Semi-Malicious Construction

We can extend the previous construction to the semi-malicious case similarly as what we did in Section 8. More precisely, we can construct a semi-malicious k-round MPC from a semi-malicious inner MPC (with an arbitrary number of rounds) and a semi-malicious k-round OT.

Contrary to the 2-round case, the unique-answer distribution $\mathrm{w}\mathcal{D} = \{\mathrm{w}\mathcal{D}_{\lambda,\mathsf{id}}\}_{\lambda,\mathsf{id}}$ does not contain a single element for each (λ, id). Actually, we even need the index id to also contain a non-uniform poly-time interactive Turing machine A corresponding to the semi-malicious adversary playing the role of the sender in the commitment (i.e., the receiver in the OT protocol). Apart from this subtlety, the construction is essentially a merge of the constructions in Sections 8 and 9.1.

Let us give more details. Instead of using a k-round interactive functional commitment scheme, we use a k-round semi-malicious equivocable interactive functional commitment scheme

Definition 9.3 (k-Round Semi-Malicious Equivocable Interactive Functional Commitment)**.** Let $\mathcal{G} = \{\mathcal{G}_\lambda\}_{\lambda\in\mathbb{N}}$ be a poly-size circuit class. A *k-round semi-malicious equivocable interactive functional commitment (eiFC)* scheme eiFC for $\mathcal{G}$ is a tuple of three polynomial-time interactive Turing machines and three polynomial-time algorithms $\mathsf{eiFC} = (\mathsf{eiFC.S}, \mathsf{eiFC.R}, \mathsf{eiFC.SimC}, \mathsf{eiFC.FOpen}, \mathsf{eiFC.FVer}, \mathsf{eiFC.SimD})$:

Commitment is performed via a $(k-1)$-round interaction between a sender $\mathsf{eiFC.S}$ on input the message to be committed v and with a random tape $\rho \in \{0,1\}^\tau$ and a receiver $\mathsf{eiFC.R}$ on random tape $\rho' \in \{0,1\}^{\tau'}$. The resulting commitment $c = \langle \mathsf{eiFC.S}(1^\lambda, v; \rho), \mathsf{eiFC.R}(1^\lambda; \rho')\rangle$ is the transcript of the interaction;

Functional Opening: $d = \mathsf{eiFC.FOpen}(c, G, y, \rho)$ is defined as $\mathsf{FC.FOpen}$ in Definition 5.1;

Functional Verification: $b = \mathsf{eiFC.FVer}(c, G, y, d)$ is defined as $\mathsf{FC.FVer}$ in Definition 5.1;

Commitment Simulation is performed via a $(k-1)$-round interaction between a simulated sender $\mathsf{eiFC.SimC}$ and a non-uniform poly-time interactive Turing machine A: $(c, \mathsf{trap}) \xleftarrow{R} \langle \mathsf{eiFC.SimC}(1^\lambda), A(1^\lambda)\rangle$ where c is the transcript of the interaction and trap is an additional output of $\mathsf{eiFC.SimC}$;

Commitment Equivocation: $d \xleftarrow{R} \mathsf{eiFC.SimD}(c, \mathsf{trap}, G, y)$ equivocates the commitment c and output a functional decommitment d of c to y for $G \in \mathcal{G}_\lambda$;

satisfying the following properties:

Correctness: For any security parameter $\lambda \in \mathbb{N}$, for any $v \in \{0,1\}^n$, for any circuit $G \in \mathcal{G}_\lambda$, for any $\rho \in \{0,1\}^\tau$, for any $\rho' \in \{0,1\}^{\tau'}$, it holds that, if $c \xleftarrow{R} \langle \mathsf{eiFC.S}(1^\lambda, v; \rho), \mathsf{eiFC.R}(1^\lambda; \rho')\rangle$, then:

$$\mathsf{eiFC.FVer}(c, G, G(v), \mathsf{eiFC.FOpen}(1^\lambda, G, v, \rho)) = 1 \ \ ;$$

Semi-Malicious Functional Binding: For any non-uniform poly-time interactive Turing machine A with the semi-malicious property defined below, there exists a negligible function negl, such that for any $\lambda \in \mathbb{N}$:

$$\Pr\Big[\mathsf{eiFC.FVer}(c, G, y, d) = 1 \text{ and } y \neq G(v) \ : \\ \rho' \xleftarrow{R} \{0,1\}^{\tau'};\ (c, (v, \rho)) = \langle A(1^\lambda), \mathsf{eiFC.R}(1^\lambda; \rho')\rangle;\ (G, y, d) \xleftarrow{R} A(1^\lambda)\Big] \leq \mathrm{negl}(\lambda) \ .$$

The adversary A is allowed to keep a state between the interaction with $\mathsf{eiFC.R}$ and the guess stage. Furthermore it is supposed to be semi-malicious, namely, after each message sent to $\mathsf{eiFC.R}$, it also outputs a valid witness (v, ρ) explaining all the previous messages. As with semi-malicious adversaries for MPC, witnesses at each round do not need to be consistent. The last round witness is the second output of $\langle A(1^\lambda), \mathsf{eiFC.R}(1^\lambda; \rho') \rangle$.

Semi-Malicious Simulatability: For any non-interactive poly-time interactive Turing machine A, there exists a negligible function negl, such that for any $\lambda \in N$:

$$\left| \Pr\left[A(d) = 1 \;:\; \begin{array}{l} \rho \xleftarrow{R} \{0,1\}^\tau;\; c \xleftarrow{R} \langle \mathsf{eiFC.S}(1^\lambda, v; \rho), A(1^\lambda) \rangle; \\ G \xleftarrow{R} A(1^\lambda);\; d = \mathsf{eiFC.FOpen}(c, G, v, \rho) \end{array} \right] - \right.$$

$$\left. \Pr\left[A(d) = 1 \;:\; \begin{array}{l} (c, \mathsf{trap}) \xleftarrow{R} \langle \mathsf{eFC.SimC}(1^\lambda), A(1^\lambda) \rangle; \\ G \xleftarrow{R} A(1^\lambda);\; d \xleftarrow{R} \mathsf{eFC.SimD}(c, \mathsf{trap}, G, G(v)) \end{array} \right] \right| \le \operatorname{negl}(\lambda) \enspace .$$

We then define the non-deterministic oracle family $\mathcal{O}^{\mathsf{eiFC}}$ associated to eiFC as $\mathcal{O}^{\mathsf{FC}}$ to FC. Finally, we need to define the unique-answer distribution $\mathrm{w}\mathcal{D}^{\mathsf{eiFC}} = \{\mathrm{w}\mathcal{D}^{\mathsf{eFC}}_{\lambda,A}\}$, where $\lambda \in \mathbb{N}$ and A is a non-uniform poly-time interactive Turing machine (as in the semi-malicious functional binding property), by defining $\mathrm{w}\mathcal{D}^{\mathsf{eFC}}_{\lambda,A}$ to be:[12]

$$\left\{ ((c, G),\; (y, d),\; \mathsf{aux} = \rho,\; \mathsf{denc} = \rho') \;:\; \begin{array}{l} \rho' \xleftarrow{R} \{0,1\}^{\tau'}; \\ (c, (v, \rho)) = \langle A(1^\lambda), \mathsf{eiFC.R}(1^\lambda; \rho') \rangle; \\ G \xleftarrow{R} A(1^\lambda);\; y = G(v);\; d = \mathsf{FC.FOpen}(c, G, v, \rho) \end{array} \right\} \enspace .$$

Furthermore, we define k-round semi-malicious OT schemes similarly as in Definition 9.2, except that semi-honest receiver privacy and semi-honest sender privacy are replaced by the following properties:

Semi-Malicious Receiver Privacy: The following two distributions are computationally indistinguishable:

$$\left\{ \mathsf{st} \;:\; \rho' \xleftarrow{R} \{0,1\}^{\tau'};\; (t, x, \mathsf{st}) = \langle A(1^\lambda), \mathsf{OT.R}(1^\lambda, 0; \rho') \rangle \right\}_{\lambda, A} \enspace ,$$
$$\left\{ \mathsf{st} \;:\; \rho' \xleftarrow{R} \{0,1\}^{\tau'};\; (t, x, \mathsf{st}) = \langle A(1^\lambda), \mathsf{OT.R}(1^\lambda, 1; \rho') \rangle \right\}_{\lambda, A} \enspace ;$$

where A is a semi-malicious adversary playing the role of the sender (and outputting a state st), i.e., A outputs a witness (x_0, x_1, ρ) after each message it sends to the receiver (as usual, witnesses need to explain all the previous messages but do not need to be consistent with each other);

Semi-Malicious Sender Privacy: The following two distributions are computationally indistinguishable:

$$\left\{ \mathsf{st} \;:\; \rho \xleftarrow{R} \{0,1\}^{\tau};\; (t, x, \mathsf{st}) = \langle \mathsf{OT.S}(1^\lambda, x_0, x_1; \rho), A(1^\lambda) \rangle \right\}_{\lambda, x_0, x_1} \enspace ,$$
$$\left\{ \mathsf{st} \;:\; \rho \xleftarrow{R} \{0,1\}^{\tau};\; (t, x, \mathsf{st}) = \langle \mathsf{OT.S}'(1^\lambda, x_0, x_1; \rho), A(1^\lambda) \rangle \right\}_{\lambda, x_0, x_1} \enspace ;$$

[12] For the security definitions to make sense, we suppose that the polynomial bounding the time of A is fixed. Another way to look at it would be to consider that the index contains a circuit A_λ, which can interact like an interactive Turing machine.

where A is a semi-malicious adversary playing the role of the receiver and $\mathsf{OT.S}'$ acts as $\mathsf{OT.S}$ except that before the last round it reads the witness tape (σ, ρ') of the adversary and uses as OT messages (x_σ, x_σ) instead of (σ_0, σ_1). We recall that we assume w.l.o.g. that the OT messages are only used to generate the last round of the OT protocol and thus $\mathsf{OT.S}'$ does not need to know them before.

A k-round semi-malicious eiFC with a WS can be constructed from a k-round semi-malicious OT. The construction is a straightforward merge of the ones in Section 8.2 and in Section 9.1.3. A k-round semi-malicious OT is defined similarly as a k-round OT (Definition 9.2) except that semi-honest receiver privacy and semi-honest sender privacy are replaced by semi-malicious ones: a semi-malicious adversary plays the role of the sender and of the receiver (respectively). In the case of sender privacy, the adversary playing the receiver can choose x_0 and x_1 after the last flow of the receiver (i.e., the $(k-1)$-th flow). We recall that without loss of generality, we suppose that the inputs x_0 and x_1 of the sender are only used in the last flow.

Then from a k-round semi-malicious eiFC with WS, we can construct GIC (with designated-encrypted information) as in Section 9.1.2. And from this, following Section 9.1.4, we can construct a k-round semi-malicious MPC (if we start from a semi-malicious inner MPC, instead of just a semi-honest one). The security proof is very similar.

9.3 Extension to Malicious Security in the CRS Model

In the CRS model, we can let each party prove using NIZK that each message is generated in a semi-malicious way (i.e., according to the protocol w.r.t. some input and random tape) as done in [AJL+12], which immediately gives Corollary 1.3 in the introduction — For any $k \geq 2$, there is a k-round malicious UC protocol in the common reference string model for any functionality f, from any k-round semi-malicious OT protocol and NIZK.

10 k-Round Malicious MPC from k-Round OT

Following the overview in Section 2.6, we construct k-round malicious MPC from k-round malicious OT, as follows: we first construct a k-round delayed-semi-malicious MPC (see Section 3.3.5) from k-round OT. Then, we show how to transform this specific k-round delayed-semi-malicious MPC into a k-round malicious MPC. We also show a (slightly simpler) generic transformation from *any* k-round delayed-semi-malicious MPC into a $(k+1)$-round malicious MPC.

The first part uses similar ideas as our semi-malicious MPC construction in Section 9.2 and requires us to define the notion of malicious equivocable interactive functional commitment (eiFC) with WS.

10.1 Delayed-Semi-Malicious Equivocable Interactive FC with WS

A delayed-semi-malicious eiFC is similar to a semi-malicious eiFC defined in Definition 9.3 except that

1. *semi-malicious functional binding* is replaced by *delayed-semi-malicious functional binding* and holds against delayed-semi-malicious adversaries (instead of semi-malicious ones), when we view an eiFC as a k-round protocol with $(k-1)$ rounds for the commitment phase and

1 round for the opening phase. Concretely, the adversary A only needs to output a witness (v, ρ) at the end of the interaction with $\mathsf{eiFC.R}$;

2. *semi-malicious simulatability* is replaced by *delayed-semi-malicious simulatability* and holds against delayed-semi-malicious adversaries (instead of semi-malicious ones), which are in this context equivalent the same as malicious adversaries. In other words, the adversary A does not need to output any witness in the experiment of the security property as it is playing the receiver of the commitment and its last flow is the $(k-2)$-th one.

A k-round delayed-semi-malicious eiFC with WS can be constructed from a k-round delayed-semi-malicious OT similarly as in Section 9.2, the only difference being the fact the base OT is delayed-semi-maliciously sender and receiver private (i.e., the adversaries in these security properties only need to provide a valid witness for the second last and the last round) instead of semi-maliciously secure. We point out that our definition of OT is much weaker than simulation-based ones against malicious adversary. In particular, a 2-round delayed-semi-malicious OT is a 2-round semi-malicious OT.

We also point out that the commitment simulator $\mathsf{eiFC.SimC}$ and the commitment equivocator $\mathsf{eiFC.SimD}$ cannot rewind the adversary. But this is not an issue as in our constructions, they act as honest OT receivers, albeit using selection bits which a commitment sender would not be allowed to use (even a malicious one as it still needs to output a pair (v, ρ)).

10.2 Delayed-Semi-Malicious MPC from Malicious eiFC with WS

When following the semi-malicious construction from Section 9.2 albeit using a malicious eiFC with WS, instead of a semi-malicious one, we automatically get a delayed-semi-malicious MPC instead of a semi-malicious one. We point out that the inner MPC still just needs to be secure against semi-malicious adversaries, as it is only simulated in the last round and at the last round the inputs and randomness of the corrupted parties is known (as the adversary is delayed semi-malicious and it needed to commit to them in the $(k-1)$-th rounds).

10.3 k-Round Malicious MPC from $(k-1)$-Round Delayed-Semi-Malicious MPC

In this section, for any $k \geq 5$, we show how to generically transform any $(k-1)$-round delayed-semi-malicious MPC protocol into a k-round malicious MPC protocol. In the next section, we show how to start from our *specific* delayed-semi-malicious MPC protocols, but with k rounds, to get k-round malicious MPC. The latter will establish that k-round malicious MPC is implied by k-round delayed-semi-malicious OT. Nonetheless, we present the former transformation since it is more modular, and may be useful for other applications.

10.3.1 Construction of Malicious MPC

Let N be any integer and f any N-party functionality. For any $k \geq 5$, to construct a k-round malicious MPC protocol Π for computing f in the plain model, we rely on the following tools.

Tools Used in Our Malicious MPC Protocols.

- A $(k-1)$-round delayed-semi-malicious MPC protocol $\Phi = (\mathsf{Next}^{\Phi}, \mathsf{Output}^{\Phi})$ for computing f in the plain model.

- a 2-message statistically binding commitment scheme Com, which is implied by one-way functions [Nao91], and
- the 4-round delayed-input NMZK proof system NMZK from one-way functions by [COSV17] (COSV), which is many-many non-malleable zero-knowledge in the synchronous setting and publicly verifiable. Our construction of malicious MPC protocol Π makes non-black-box use of the COSV NMZK protocol. We detail the tools used inside their construction next.

Tools Used in the COSV NMZK Protocol.

- A 3-round trapdoor-setup protocol Trap between a sender and a receiver, which proceeds as follows:

  i) In the first round, the sender samples a pair of $(\mathrm{sk}, \mathrm{vk})$ signing and verification key of a signature scheme, and sends vk to the receiver,

  ii) the receiver sends a random challenge message $m \overset{R}{\leftarrow} \{0,1\}^\lambda$, and

  iii) the sender returns a signature σ of m signed by sk, and the receiver accepts if (m, σ) is valid w.r.t. vk.

  A valid trapdoor $\mathrm{td} = (m_0, \sigma_0, m_1, \sigma_1)$ w.r.t. the verification key vk consists of two valid signatures for distinct messages $m_0 \neq m_1$ w.r.t. vk. Clearly, for a malicious receiver who participates in one execution of the protocol with an honest sender, it is hard to find a trapdoor by the unforgeability of the signature scheme. On the other hand, a resetting receiver can easily obtain a trapdoor by rewinding the honest sender to obtain signatures for different messages.

- A 4-round public-coin commitment scheme NMCom that is many-many non-malleable (w.r.t. commitment) in the synchronous setting, and extractable (with over-extraction) by rewinding the last two messages. Such a commitment scheme was constructed by [GPR16] from one-way functions.

- The 4-round delayed-input WI proof $\mathsf{WI}^{\mathrm{or}}$ constructed in [COSV17] from one-way functions. The COSV WI protocol $\mathsf{WI}^{\mathrm{or}}$ is for an or-language $\mathcal{L} \in \mathrm{NP}$ defined by two arbitrary NP languages $\mathcal{L}_0, \mathcal{L}_1$, such that, a statement $x = (x_0, x_1)$ is in $\mathcal{L}$ if and only if $x_0 \in \mathcal{L}_0$ or $x_1 \in \mathcal{L}_1$. Moreover, the protocol $\mathsf{WI}^{\mathrm{or}}$ is public-coin, adaptive-input special sound, and satisfies the following robustness property w.r.t. any extractable statistically binding commitment schemes $\langle C, R \rangle$. In particular, $\mathsf{WI}^{\mathrm{or}}$ is robust w.r.t. NMCom described above. Below we detail on the definitions of adaptive-input special-soundness and robustness.

  – Adaptive-input Special-soundness: Given two accepting proofs with the same first two messages and different third messages for potentially two different statements, $(\alpha, \beta, \gamma, \tau)$ for statement x and $(\alpha, \beta, \gamma', \tau')$ for statement x', there is an efficient way of computing a valid witness w for x and a valid witness w' for x'.

  – Robustness: For any many-in-the-middle adversary A consider the following execution using $\mathsf{WI}^{\mathrm{or}}$ and $\langle C, R \rangle$ with security parameter λ: $A(1^\lambda)$ interacts with the honest prover of $\mathsf{WI}^{\mathrm{or}}$ on the left and the honest receiver R of $\langle C, R \rangle$ on the right. In the left session, after the first three messages, A adaptively chooses a challenge statement $x = (x_0, x_1)$

and valid witnesses w_0, w_1 for statements x_0, x_1 respectively. The honest prover of $\mathsf{WI}^{\mathrm{or}}$ then uses witness w_b for a bit $b \in \{0,1\}$ to generate the last prover's message. In the right session, A sends a commitment c using $\langle C, R\rangle$. Denote by $\mathsf{wiMIM}^{A}_{\langle C,R\rangle}(1^\lambda, b)$ the view view_A of A together with the value v it commits to in c; v is set to $\perp$ if c is invalid. (Note that the random variable v is well-defined with overwhelming probability, since by the statistical binding property of $\langle C, R\rangle$, the probability that there exist two valid decommitment of c to different values is negligible.)

Claim 10.1 (Robustness of the COSV WI proof)**.** *Let $\langle C, R\rangle$ be any extractable statistical-binding commitment scheme. The delayed-input WI proof system of [COSV17] satisfies the following robustness property w.r.t. $\langle C, R\rangle$: For any computationally efficient predicate $P: \{0,1\}^* \times (\{0,1\}^* \cup \{\perp\}) \to \{0,1\}$ satisfying that whenever the second input is $\perp$, P outputs 0, there exists a negligible function* negl*, such that,*

$$|\Pr[P(\mathsf{wiMIM}^{A}_{\langle C,R\rangle}(1^\lambda, 0)) = 1] - \Pr[P(\mathsf{wiMIM}^{A}_{\langle C,R\rangle}(1^\lambda, 1)) = 1]| \le \mathrm{negl}(\lambda) \ .$$

The Malicious MPC Protocol Π. To compute f on inputs $x_1, \cdots, x_N$, parties $P_i(1^\lambda, x_i)$ run the following sub-protocols

- **Component 1, Computation by the delayed-semi-malicious MPC protocol Φ (Round $1, \cdots, k-3, k-1, k$):**
  For every $i \in [N]$, P_i samples a sufficiently long random tape r_i at the beginning of its execution. Then, all the parties jointly run Φ, where P_i uses input x_i and random tape r_i. The $k-1$ messages of Φ are sent in rounds $1, \cdots, k-3, k-1, k$, that is, only round $k-2$ is skipped. Let m_i^ℓ be the message broadcast by P_i in round ℓ of Φ, and $\bar{m}^{<\ell} = \{m_j^{\ell'}\}_{j,\ell'<\ell}$ be messages broadcast by all parties in the first $\ell - 1$ rounds of Φ.

- **Component 2, $\langle C, R\rangle$ commitment to input and randomness (Round 1 and 2):**
  For every distinct $i, j \in [N]$, P_i commits to (x_i, r_i) to P_j using $\langle C, R\rangle$. The 2 messages of $\langle C, R\rangle$ are sent in the first two rounds. Let $c_{i\to j}$ be the commitment, and $\rho_{i\to j}$ the decommitment.

- **Component 3, NMZK proof of correctness of the first $k-2$ rounds of Φ (Round $k-4, \cdots, k-1$):**
  For every distinct $i, j \in [N]$, P_i proves to P_j using COSV NMZK NMZK that it has generated its first $k-2$ messages in Φ correctly using the input and random tape committed in $c_{i\to j}$, that is, prove that the statement $X^{k-2}_{i\to j}$ is in language $\mathcal{L}_X$ defined by relation $\mathcal{R}_X$ below. Let $\mathsf{NMZK}_{i\to j}$ be the produced proof.

$$\begin{array}{lll} X^t_{i\to j} = (t,\ \bar{m}^{\le t},\ c_{i\to j}) & & \\ \mathcal{R}_X(X^t_{i\to j},\ \rho_{i\to j}) = 1 \quad & \text{iff} \quad \text{i)} & \rho_{i\to j} \text{ decommits } c_{i\to j} \text{ to } x_i, r_i, \text{ and} \\ & \quad\quad\ \text{ii)} & \forall \ell \le t, \quad m_i^\ell = \mathsf{Next}_i^\Phi(1^\lambda, x_i, r_i, m^{\le \ell-1}) \end{array} \qquad (5)$$

  More precisely, to give the NMZK proof, P_i and P_j do the following:

  - Trapdoor of Trap: P_i and P_j run the trapdoor setup protocol Trap in rounds $k-4, k-3, k-2$, where P_j acts as the sender and P_j the receiver. Let $\mathsf{Trap}_{j\to i}$ denote the produced transcript, and $\mathrm{vk}_{j\to i}$ the verification key that P_j sends to P_i.

  - Commitment of NMCom to one share $s^0_{i\to j}$: P_i commits to a random string $s^0_{i\to j}$ to P_j using the non-malleable commitment scheme NMCom, in rounds $k-4, k-3, k-2, k-1$. Let $\mathsf{NMCom}_{i\to j}$ denote the produced commitment and $\tau_{i\to j}$ the decommitment.
  - Another share $s^1_{i\to j}$ in the clear: P_i sends another random string $s^1_{i\to j}$ in the clear to P_j.
  - The first WI^{or} proof: P_i proves to P_j using the WI proof WI^{or} in rounds $k-4, k-3, k-2, k-1$ that
    * Honest Statement: Either, the statement $X^{k-1}_{i\to j}$ is in $\mathcal{L}_X$,
    * Cheating Statement: Or, the XOR of the share $s^0_{i\to j}$ committed to in $\mathsf{NMCom}_{i\to j}$ and the share $s^1_{i\to j}$ is a trapdoor w.r.t. $\text{vk}_{j\to i}$.

    We refer to the above two statements the honest and cheating statements respectively. That is, P_i proves that the statement $Y^{k-1}_{i\to j}$ is in the or-language $\mathcal{L}_Y$ defined below.

$$\begin{aligned} & Y^t_{i\to j} = (X^t_{i\to j}, \text{vk}_{j\to i}, \mathsf{NMCom}_{i\to j}, s^1_{i\to j}) \\ & \mathcal{R}_Y(Y^t_{i\to j}, (\rho_{i\to j}, \tau_{i\to j})) = 1 \quad \textit{iff} \\ & \quad \mathcal{R}_X(X^t_{i\to j}, \rho_{i\to j}) = 1, \\ & \quad OR \begin{array}{l} \text{i) } \tau_{i\to j} \text{ decommits } \mathsf{NMCom}_{i\to j} \text{ to } s^0_{i\to j}, \text{ and} \\ \text{ii) } s^0_{i\to j} \oplus s^1_{i\to j} \text{ contains two signatures for distinct messages w.r.t. } \text{vk}_{i\to j} \end{array} \end{aligned} \tag{6}$$

  Let $\mathsf{WI}^{\text{or},1}_{i\to j}$ be the produced proof. Each party P_i verifies all WI^{or} proofs $\{\mathsf{WI}^{\text{or},1}_{i\to j}\}_{i,j\in[N]}$. If any proof is not accepting, it aborts and outputs $\bot$. This can be done thanks to the fact that WI^{or} is public-coin and hence publicly verifiable.

- **Component 4, The second WI^{or} proof of correctness of all $k-1$ rounds of Φ (Round $k-3, \cdots, k$):**
  For every distinct $i, j \in [N]$, P_i proves to P_j using WI^{or} that
  - Honest statement: Either, it has generated all its messages in Φ correctly using the input and random tape committed to in $c_{i\to j}$, that is, $X^k_{i\to j}$ is in $\mathcal{L}_X$,
  - Cheating statement: Or, the XOR of the share $s^0_{i\to j}$ committed to in $\mathsf{NMCom}_{i\to j}$ and the share $s^1_{i\to j}$ is a trapdoor w.r.t. $\text{vk}_{j\to i}$.

  That is, P_i proves that the statement $Y^k_{i\to j}$ is in $\mathcal{L}_Y$ as defined above. Let $\mathsf{WI}^{\text{or},2}_{i\to j}$ be the produced proof.

- **Deriving Output:** Each party P_i verifies all WI^{or} proofs $\{\mathsf{WI}^{\text{or},2}_{i\to j}\}_{i,j\in[N]}$. If any proof is not accepting, it aborts and outputs $\bot$. Otherwise, it returns the output derived from the MPC protocol Φ, that is,

$$y_i = \mathsf{Output}^{\Phi}_i(1^\lambda, x_i, r_i, \bar{m}^{\le k-1})$$

It is easy to see that the correctness of the protocol Π follows from that of Φ.

10.3.2 Proof of Malicious Security

We prove that Π is maliciously secure.

Theorem 10.2. *Let f be any functionality. The above MPC protocol Π for f is secure against malicious adversaries.*

To show the theorem, we need to show that for every non-uniform poly-time interactive Turing machine A corrupting a set of parties I, there exists a non-uniform expected-poly-time interactive Turing Sim, such that, for every vector of inputs $\bar{x}$, the distribution of the outputs of honest parties and A running Π in the real world, is indistinguishable to the outputs of the honest parties and Sim interacting with the ideal functionality in the ideal world. That is,

$$\{\mathsf{Ideal}_{I,\mathsf{Sim}}(1^\lambda, \bar{x})\}_{\lambda,I,\bar{x}} \approx \{\mathsf{Real}_{I,A}(1^\lambda, \bar{x})\}_{\lambda,I,\bar{x}} \ .$$

We start with describing the simulator Sim for adversary A.

Overview of the Simulator Sim: In the ideal world, the simulator Sim corrupting parties in set I, internally runs A and simulates an execution of Π for A by simulating the honest parties in $\bar{I}$. To to simulate messages in Φ for A, Sim employs the simulator Sim_Φ of Φ. Recall that Φ is secure against delayed-semi-malicious adversaries A_Φ, who is arbitrarily malicious, except that it outputs a witness at the second last round (i.e., round $k-2$) of Φ. A valid witness contains the inputs and random tapes of corrupted parties that are consistent with (i.e., generates) their messages sent in *all* rounds (If the witness is invalid, honest parties abort and output $\perp$). The security of Φ guarantees that there exists a universal simulator $\mathsf{Sim}_\Phi(1^\lambda, I)$ that simulates the honest parties' messages of Φ for A_Φ *in a straight-line*, by making use of the witness that A_Φ outputs. In order to use Sim_Φ, the simulators Sim needs to "turn" the malicious adversary A (against Π) into a delayed-semi-malicious adversary A_Φ (against Φ). It achieves so by extracting witnesses from the first WI proofs $\mathsf{WI}^{\mathrm{or},1}$ sent by A. In addition, when messages of the honest parties of Φ are simulated by Sim_Φ, Sim can on longer convince A in the WI proofs $\mathsf{WI}^{\mathrm{or},1}, \mathsf{WI}^{\mathrm{or},2}$ that these messages are honestly generated. Therefore, Sim cheats by proving the fake statement that it has committed to a trapdoor in the NMCom commitment. While doing so, we rely on the non-malleability of NMCom, the robustness of $\mathsf{WI}^{\mathrm{or}}$ against NMCom, and other properties to ensure that A never commits to a trapdoor itself and hence must prove the honest statements in WI proofs $\mathsf{WI}^{\mathrm{or},1}, \mathsf{WI}^{\mathrm{or},2}$ it sends. This, in particular, ensures that the witnesses extracted from the first WI proofs $\mathsf{WI}^{\mathrm{or},1}$ from A form a valid witness for messages of corrupted parties in Φ.

The Simulator Sim: We now describe the procedure of Sim formally, where simulation of the *main thread* — the thread in which the output of A is output in the end — and rewindings for different extraction purposes are interleaved.

- **Stage 1, Simulate main thread rounds $1, \ldots, k-2$:** Sim simulates honest parties' messages as follows:

   1. Simulate messages in Φ: Run $\mathsf{Sim}_\Phi(1^\lambda, I)$ to simulate messages from honest parties in Φ. In slight more detail, Sim forwards messages of Φ between Sim_Φ and A; whenever Sim_Φ sends the inputs $\bar{x}_I$ of the corrupted parties, Sim forwards it and obtain the outputs of the corrupted parties $\bar{y}_I$, and whenever Sim_Φ decides to abort the honest parties in the

ideal world, Sim decides to abort them too. Note that, Sim_Φ does not need any witness in order to simulate the first $k-2$ rounds, and only needs it for simulating the last $k-1$ round.

2. Simulate Com commitments: For every distinct $i, j \in [N]$ s.t. i or j is in $\bar{I}$, simulate the Com commitment $c_{i\to j}$ as follows: If P_i is honest, commit to $0^{\text{poly}(\lambda)}$ in $c_{i\to j}$ (instead of the input and random tape x_i, r_i of P_i), and if P_j is honest, emulate the receiver of Com honestly.
3. Simulate NMZK proofs: For every distinct $i, j \in [N]$ s.t. i or j is in $\bar{I}$, emulate the first three messages of the NMZK proof $\mathsf{NMZK}_{i\to j}$ by running the honest prover (if P_i is honest) or the honest verifier (if P_j is honest) of NMZK. Note that this can be done since NMZK is delayed-input and the first three messages depend only on the length of the statement. In particular, if P_i is honest, Sim does the following to emulate the prover's message:
   - Act as the honest receiver of Trap: Upon receiving verification key $\text{vk}_{j\to i}$, send a random challenge message $m_{i\to j}$ and receive a signature $\sigma_{i\to j}$.
   - Commit in the first three messages of $\mathsf{NMCom}_{i\to j}$ to a random share $s^0_{i\to j}$.
   - Act as the honest prover to in the first three messages of $\mathsf{WI}^{\text{or},1}_{i\to j}$.
4. Simulate the second WI^{or} proofs: For every distinct $i, j \in [N]$ s.t. i or j is in $\bar{I}$, emulate the first two messages of the WI proofs $\mathsf{WI}^{\text{or},2}_{i\to j}$ by running the honest prover (if P_i is honest) and verifier (if P_j is honest) of WI^{or}.

If A aborts at any point, Sim aborts and outputs $\perp$. Let $\mathsf{trans}^{\leq k-2}$ denote the obtained transcript.

- **Stage 2, Rewind rounds $k-3, k-2$ to extract trapdoors:** Sim keeps rewinding A from round $k-3$ to round $k-2$ in order to extract a trapdoor w.r.t. every verification key $\text{vk}_{j\to i}$ sent to an honest party P_i. In each rewinding, Sim restores the state of A at the end of round $k-4$ in the main thread, and re-execute round $k-3$ and $k-2$ as follows:

  1. *Replay* the honest parties' messages in round $k-3$ of Φ in the main thread. (In round $k-2$, there are no messages of Φ being sent.) Note that this avoid rewinding messages in Φ.
  2. Simulate the honest parties' messages in other components $c_{i\to j}, \mathsf{NMZK}_{i\to j}, \mathsf{WI}^{\text{or},2}_{i\to j}$ exactly as done in the main thread, described above.

  Sim keeps rewinding until it obtains another successful transcript trans' of round $k-3$ and $k-2$. For every $\text{vk}_{j\to i}$ sent to an honest party P_i, find two valid message-signature pairs w.r.t. $\text{vk}_{j\to i}$, $(m_{i\to j}, \sigma_{i\to j})$ and $(m'_{i\to j}, \sigma'_{i\to j})$ in the transcript $\mathsf{trans}^{\leq k-2}$ of the main thread and trans' respectively; if $m_{i\to j} = m'_{i\to j}$, Sim aborts and outputs err_1, and otherwise, it sets trapdoor $\text{td}_{j\to i}$ to these two pairs.

- **Stage 3, Simulate main thread round $k-1$ using trapdoors:** Using trapdoors obtained above, Sim simulates honest parties' messages in round $k-1$ as follows:

1. Simulate messages in Φ and the second $\mathsf{WI}^{\mathrm{or}}$ proofs: Simulate honest player's messages in round $k-2$ of Φ using Sim_Φ, and simulate third messages in the second WI proofs $\mathsf{WI}^{\mathrm{or},2}_{i\to j}$ honestly, as in Stage 1.
2. Simulate the NMZK proofs: For every honest $i \in \bar{I}$, and any $j \neq i$, emulate the fourth message of $\mathsf{NMZK}_{i\to j}$ (from the prover) continuing from Stage 1 as follows:
   - Commit in the last message of $\mathsf{NMCom}_{i\to j}$ to the random share $s^0_{i\to j}$ tossed in Stage 1. Let $\tau_{i\to j}$ be the decommitment.
   - Send the other share $s^1_{i\to j} = s^0_{i\to j}\ \mathsf{xor}\ \mathrm{td}_{j\to i}$ in the clear.
   - Prove in the last message of $\mathsf{WI}^{\mathrm{or},2}_{i\to j}$ the fake statement that $\mathsf{NMCom}_{i\to j}$ commits to $s^0_{i\to j}$, such that, $\mathrm{td}_{j\to i} = s^0_{i\to j}\,\mathsf{xor}\, s^1_{i\to j}$ is a valid trapdoor w.r.t. $\mathrm{vk}_{j\to i}$. More precisely, prove the statement $Y^{k-1}_{i\to j}$ defined in Equation 6 using the witness $(\perp, \tau_{i\to j})$.

- **Stage 4, Rewind rounds $k-2, k-1$ to extract inputs and random tapes of corrupted parties:** Sim keeps rewinding A from round $k-2$ to round $k-1$ in order to extract a witness $w_{j\to i}$ from every WI proof $\mathsf{WI}^{\mathrm{or},1}_{j\to i}$ where the prover P_j is corrupted and the verifier P_i is honest. In each rewinding, Sim restores the state of A at the end of round $k-3$ in the main thread, and re-execute rounds $k-2$ and $k-1$ as follows:

  1. *Replay* the honest parties' messages in round $k-1$ of Φ in the main thread. (In round $k-2$, there are no messages of Φ being sent.)
  2. Simulate the honest parties' messages in other components $\mathsf{NMZK}_{i\to j}, \mathsf{WI}^{\mathrm{or},2}_{i\to j}$ exactly as done in the main thread.

  Sim keeps rewinding until it obtains another successful transcript trans'' of round $k-2$ and $k-1$. For every WI proof $\mathsf{WI}^{\mathrm{or},1}_{j\to i}$ where the prover P_j is corrupted and the verifier P_i is honest, find two accepting transcripts of $\mathsf{WI}^{\mathrm{or},1}_{j\to i}$ sharing the same first two messages in $\mathsf{trans}^{\leq k-2}$ of the main thread and trans''; if these two transcripts have the same third message, Sim aborts and outputs err_2, and otherwise, it extracts a witness $w_{j\to i}$ from them by relying on the adaptive-input special soundness of $\mathsf{WI}^{\mathrm{or}}$. In addition, if any witness $w_{j\to i}$ is not a witness of the honest statement $X^{k-2}_{j\to i}$ defined in Equation 5, it aborts and outputs err_3. Otherwise, every $w_{j\to i}$ is of form $(\rho_{j\to i}, \star)$, such that, $\rho_{j\to i}$ decommits $\mathsf{Com}_{j\to i}$ to a pair $(x_{j\to i}, r_{j\to i})$ that is consistent with P_j's messages in the first $k-1$ rounds of Φ. For every corrupted party P_j, set (x_j, r_j) to $(x_{j\to i}, r_{j\to i})$ for an arbitrary honest $i \in \bar{I}$.

- **Stage 5, Simulate main thread round k using extracted input and randomness:** Using the corrupted parties' inputs and random tapes $\{x_j, r_j\}_{j\in I}$ extracted from A, Sim simulates honest parties' messages in round k as follows:

  1. Simulate messages in Φ: Feed Sim_Φ the witness $w = \{x_j, r_j\}_{j\in I}$, which simulate honest player's last messages in Φ.
  2. Simulate the second $\mathsf{WI}^{\mathrm{or}}$ proofs: For every honest $i \in \bar{I}$, and any $j \neq i$, prove in the last message of $\mathsf{WI}^{\mathrm{or},2}_{i\to j}$, the fake statement that $\mathsf{NMCom}_{i\to j}$ commits to $s^0_{i\to j}$, such that, $\mathrm{td}_{j\to i} = s^0_{i\to j}\ \mathsf{xor}\ s^1_{i\to j}$ is a valid trapdoor w.r.t. $\mathrm{vk}_{j\to i}$. More precisely, prove the statement $Y^k_{i\to j}$ defined in Equation 6 using the witness $(\perp, \tau_{i\to j})$.

Finally, if the honest parties do not abort in the main thread, and for any $j \in I$, (x_j, r_j) obtained in Stage 4 is not consistent with P_j's last message in Φ, Sim outputs err_4.

To establish the correctness of Sim, we need to show that Sim runs in expected polynomial time, and the distributions of outputs in the real and ideal worlds are indistinguishable.

Lemma 10.3. *For any N-party functionality f, and any poly-sized adversary A controlling any subset $I \subset [N]$ of parties. The simulator Sim for A described above runs in expected polynomial time.*

Lemma 10.4. *For any N-party functionality f, and any poly-sized adversary A controlling any subset $I \subset [N]$ of parties. The simulator Sim for A described above satisfies that:*

$$\left\{\mathsf{Ideal}_{I,\mathsf{Sim}}(1^\lambda, \bar{x})\right\}_{\lambda, I, \bar{x}} \approx \left\{\mathsf{Real}_{I,A}(1^\lambda, \bar{x})\right\}_{\lambda, I, \bar{x}} \ .$$

Correctness of Simulation — Proof of Lemma 10.3 and 10.4 For any adversary A, we show the correctness of the simulator Sim through a sequence of hybrids $\mathcal{H}_0, \cdots, \mathcal{H}_6$ that gradually "morph" a real world execution $\mathcal{H}_0(1^\lambda, \bar{x}) = \mathsf{Real}_{I,A}(1^\lambda, \bar{x})$ into an ideal world execution $\mathcal{H}_6(1^\lambda, \bar{x}) = \mathsf{Ideal}_{I,\mathsf{Sim}}(1^\lambda, \bar{x})$ in an indistinguishable way. Throughout all hybrids, we show that the runtime of all hybrids are polynomial in expectation, and that the outputs of all parties are indistinguishable in every pair of neighboring hybrids. Towards this, we will maintain the following soundness condition that the following event bad happens with negligible probability:

- Event bad: In the main thread, there exists some $j \in I$ and $i \in \bar{I}$, such that, the XOR of the share $s^0_{j\to i}$ that A commits to in $\mathsf{NMCom}_{j\to i}$ and the share $s^1_{j\to i}$ is a trapdoor w.r.t. $\mathrm{vk}_{i\to j}$, that is, $s^0_{j\to i} \;\mathsf{xor}\; s^1_{j\to i}$ contains two valid signatures for two distinct messages under $\mathrm{vk}_{i\to j}$.

We abuse notation and denote by $\mathcal{H}_k(1^\lambda, \bar{x})$ the distribution of the outputs of honest parties and the output of A in the main thread.

Hybrid $\mathcal{H}_0$: This hybrid is identical to the real world execution $\mathsf{Real}_{I,A}(1^\lambda, \bar{x})$. Clearly, $\mathcal{H}_0$ runs in strict polynomial time. We claim that event bad almost never occurs in this hybrid.

Claim 10.5. *For every λ and inputs $\bar{x}$, the probability that event bad occurs in $\mathcal{H}_0$ is negligible.*

Proof. Recall that in the real world, for any corrupted $j \in I$ and honest $i \in \bar{I}$, A obtains only a single signature under $\mathrm{vk}_{i\to j}$ (of a message of its choice) in $\mathsf{Trap}_{i\to j}$, where $\mathrm{vk}_{i\to j}$ is sampled honestly by the honest player P_i. Suppose for contradiction that event bad occurs with noticeable probability; it must occur w.r.t. some i, j with noticeable probability. Then, one can extract two signatures of different messages under $\mathrm{vk}_{i\to j}$ without rewinding $\mathsf{Trap}_{i\to j}$, which violates the unforgeability of the signature scheme. Extraction works as follows: By the extractability of $\mathsf{NMCom}_{i\to j}$, one can rewind the last two messages of $\mathsf{NMCom}_{i\to j}$ in rounds $k-2, k-1$ to extract the committed share $s^0_{i\to j}$. When event bad occurs, the commitment $\mathsf{NMCom}_{i\to j}$ is valid and hence extraction must succeed. Moreover, this extraction does not rewind $\mathsf{Trap}_{i\to j}$ since one can replay the same signature in round $k-2$. XORing $s^0_{i\to j} \;\mathsf{xor}\; s^1_{i\to j}$ gives two valid signatures of different messages under $\mathrm{vk}_{i\to j}$. □

Hybrid $\mathcal{H}_1$: This hybrid is identical to $\mathcal{H}_0$ except that after generating messages in the first $k-2$ rounds in the main thread, $\mathcal{H}_1$ keeps rewinding rounds $k-3, k-2$ to extract a trapdoor w.r.t. every verification key $\mathrm{vk}_{j\to i}$ sent to an honest party P_i. Extraction is done as in Stage 2 of Sim, except that here the main thread is generated honestly, that is,

- *Replay* the honest parties' messages in round $k-3$ of Φ in the main thread.
- Generate honest parties' messages in other components $c_{i\to j}, \mathsf{NMZK}_{i\to j}, \mathsf{WI}^{\mathrm{or},2}_{i\to j}$ in rounds $k-3, k-2$ as done in the main thread. (Different from the main thread of Sim, here $c_{i\to j}$ commits to the input and random tape (x_i, r_i) of P_i used in Φ.)

Keep rewinding until obtaining another successful transcript trans' of rounds $k-3, k-2$. For every $\mathrm{vk}_{j\to i}$ sent to an honest party P_i, find two valid signatures for messages $m_{i\to j}, m'_{i\to j}$ in the main thread and in trans'; if $m_{i\to j} = m'_{i\to j}$, $\mathcal{H}_1$ aborts and outputs err_1, and otherwise, it sets trapdoor $\mathrm{td}_{j\to i}$ to these two pairs.

We first argue that this hybrid runs in expected polynomial time.

Claim 10.6. *There exists a universal polynomial $\mathcal{T}_1$, such that, for every λ and inputs $\bar{x}$, the run time of $\mathcal{H}_1$ is $\mathcal{T}_1(\mathcal{T}_A(\lambda))$ in expectation, where $\mathcal{T}_A(\lambda)$ is the runtime of A.*

Proof. Observe that the time for generating messages in the main thread and in *each* rewinding thread is a fixed polynomial $\mathcal{T}'(\mathcal{T}_A)$ in the runtime of A. Therefore, to bound the runtime of $\mathcal{H}_1$, it suffices to bound the expected number of rewindings performed for extracting the trapdoor. Let ρ be any prefix of execution of $\mathcal{H}_1$ up to the end of round $k-4$ in the main thread. It is without of loss generality to assume that honest parties' random tapes for computing messages in the instance of Φ are sampled inside ρ. Let $p[\rho]$ be the probability that no party aborts before the end of round $k-2$ in the main thread conditioned on ρ appearing. In each rewinding of rounds $k-3, k-2$, $\mathcal{H}_1$ simulates honest parties' messages identically as in the main thread, continuing from ρ. In particular, since the honest parties' random tapes for running Φ are contained in ρ, their messages in round $k-3$ of Φ must be identical to that in the main thread. (Honest parties' other messages are generated randomly and independently as in the main thread.) Thus, the probability that A does not abort in a rewinding is exactly $p[\rho]$. The expected number of rewindings is thus $p[\rho] \times 1/p[\rho] = 1$.

Therefore, there exists a universal polynomial $\mathcal{T}_1$, such that, $\mathcal{H}_1$ runs in time $\mathcal{T}_1(\mathcal{T}_A)$ in expectation. □

Next, we show the following useful claim.

Claim 10.7. *For every λ and inputs $\bar{x}$, the probability that $\mathcal{H}_1$ outputs err_1 is negligible.*

Proof. Suppose for contradiction that $\mathcal{H}_1$ outputs err_1 with noticeable probability. This means for some $i \in \bar{I}$ and $j \neq i$, $\mathcal{H}_1$ fails to extract signatures for two different messages under $\mathrm{vk}_{i\to j}$ with noticeable probability. By definition of $\mathcal{H}_1$, it keeps rewinding until it obtains a successful rewinding of rounds $k-3, k-2$. Thus, $\mathcal{H}_1$ must find two valid signatures for messages $m_{i\to j}, m'_{i\to j}$, and it fails to obtain a trapdoor only if $m_{i\to j} = m'_{i\to j}$.

By Claim 10.6, $\mathcal{H}_1$ runs in expected polynomial time. Thus, the probability that $\mathcal{H}_1$ makes more than 2^λ rewindings is negligible. In each rewinding k, the honest party P_i chooses the challenge message $m^k_{i \to j}$ to be signed at random. Therefore, when the challenge messages are sufficiently long, the probability that any two challenge messages in these 2^λ rewindings collide is negligible. This implies that the probability that $m_{i \to j} = m'_{i \to j}$ is negligible, which is a contradiction. □

Note that the views of all parties in the main threads of $\mathcal{H}_0$ and $\mathcal{H}_1$ are identical, except that $\mathcal{H}_1$ may output err_1. By the above claim, we have that the views of all parties in the main threads of $\mathcal{H}_0$ and $\mathcal{H}_1$ are statistically close. Therefore,

Claim 10.8. *The outputs of all parties in $\mathcal{H}_0$ and $\mathcal{H}_1$ are indistinguishable, that is,*

$$\{\mathcal{H}_0(1^\lambda, \bar{x})\}_{\lambda, \bar{x}} \approx \{\mathcal{H}_1(1^\lambda, \bar{x})\}_{\lambda, \bar{x}}$$

Claim 10.9. *For every λ and inputs $\bar{x}$, the probability that event* bad *occurs in $\mathcal{H}_1$ is negligible.*

Hybrid $\mathcal{H}_2$: This hybrid proceeds identically to $\mathcal{H}_1$ except for the following difference in the main thread:

- in $\mathcal{H}_1$ for every honest $i \in \bar{I}$ and $j \neq i$, the party P_i commits to a random share $s^0_{i \to j}$ in $\mathsf{NMCom}_{i \to j}$ and sends another random share $s^1_{i \to j}$ in the clear, but
- in $\mathcal{H}_2$, share $s^1_{i \to j}$ is set to $s^0_{i \to j} \,\mathsf{xor}\, \mathrm{td}_{j \to i}$, where $\mathrm{td}_{j \to i}$ is the extracted trapdoor for $\mathrm{vk}_{j \to i}$.

Note that $\mathcal{H}_1$ and $\mathcal{H}_2$ proceed identically till the end of round $k-2$ in the main thread. Thus, their expected run-time are identical up to this point. After this point, both hybrids run in a fixed polynomial time. Hence,

Claim 10.10. *There exists a universal polynomial $\mathcal{T}_2$, such that, for every λ and inputs $\bar{x}$, the run time of $\mathcal{H}_2$ is $\mathcal{T}_2(\mathcal{T}_A(\lambda))$ in expectation, where $\mathcal{T}_A(\lambda)$ is the runtime of A.*

Alternatively, we can view the shares in $\mathcal{H}_1$ and $\mathcal{H}_2$ as being generated as follows: In $\mathcal{H}_1$, they are set as $s^0_{i \to j} = u_{i \to j}$ and $s^1_{i \to j} = u'_{i \to j} \,\mathsf{xor}\, \mathrm{td}_{j \to i}$ for randomly and independently sampled $u_{i \to j}$ and $u'_{i \to j}$, whereas in $\mathcal{H}_2$, they are $s^0_{i \to j} = u'_{i \to j}$ and $s^1_{i \to j} = u'_{i \to j} \,\mathsf{xor}\, \mathrm{td}_{j \to i}$. In this equivalent view, the only difference in main threads of $\mathcal{H}_1$ and $\mathcal{H}_2$ lies in the values committed in $\mathsf{NMCom}_{i \to j}$ by honest parties. By the hiding of NMCom, we have that

Claim 10.11. *The outputs of all parties in $\mathcal{H}_0$ and $\mathcal{H}_1$ are indistinguishable, that is,*

$$\{\mathcal{H}_1(1^\lambda, \bar{x})\}_{\lambda, \bar{x}} \approx \{\mathcal{H}_2(1^\lambda, \bar{x})\}_{\lambda, \bar{x}}$$

Furthermore, since NMCom is many-many non-malleable in synchronous setting, we have that the values that A commits to in $\mathsf{NMCom}_{j \to i}$ sent by corrupted parties P_j do not change in $\mathcal{H}_1$ and $\mathcal{H}_2$, in particular, this implies:

Claim 10.12. *For every λ and inputs $\bar{x}$, the probability that event* bad *occurs in $\mathcal{H}_2$ is negligible.*

We remark that we can reduce the above claims to the hiding and non-malleability of NMCom, despite that $\mathcal{H}_1$ and $\mathcal{H}_2$ internally rewind rounds $k-3, k-2$. This is because, during rewindings, the committer's messages of NMCom from honest parties can be emulated internally by committing to freshly sampled s^0 shares (the s^1 shares are never sent in rewindings), while the receiver's messages of NMCom from honest parties can be emulated by sending random coins, thanks to the fact that NMCom is public-coin. Therefore, the NMCom commitments in the main thread are never rewound.

Hybrid $\mathcal{H}_3$: This hybrid proceeds identically to $\mathcal{H}_2$ except for the following difference in the main thread:

- in $\mathcal{H}_2$, for every honest $i \in \bar{I}$ and $j \neq i$, the honest party P_i proves the honest statement in the WI proofs $\mathsf{WI}^{\text{or},1}_{i \to j}, \mathsf{WI}^{\text{or},2}_{i \to j}$, whereas
- in $\mathcal{H}_3$, P_i proves the fake statement that $\mathsf{NMCom}_{i \to j}$ commits to $s^0_{i \to j}$, such that, $\text{td}_{j \to i} = s^0_{i \to j} \mathsf{\ xor\ } s^1_{i \to j}$ is a valid trapdoor w.r.t. $\text{vk}_{j \to i}$.

Since the WI proofs use delayed inputs. $\mathcal{H}_2$ and $\mathcal{H}_3$ proceed identically till the end of round $k-2$ in the main thread. After this point, both hybrids run in a fixed polynomial time in $\mathcal{T}_A$. Hence,

Claim 10.13. *There exists a universal polynomial $\mathcal{T}_3$, such that, for every λ and inputs $\bar{x}$, the run time of $\mathcal{H}_3$ is $\mathcal{T}_3(\mathcal{T}_A(\lambda))$ in expectation, where $\mathcal{T}_A(\lambda)$ is the runtime of A.*

Therefore, by the witness indistinguishability of WI^{or}, we have that

Claim 10.14. *The outputs of all parties in $\mathcal{H}_2$ and $\mathcal{H}_3$ are indistinguishable, that is,*

$$\{\mathcal{H}_2(1^\lambda, \bar{x})\}_{\lambda,\bar{x}} \approx \{\mathcal{H}_3(1^\lambda, \bar{x})\}_{\lambda,\bar{x}}$$

Furthermore, WI^{or} is robust w.r.t. NMCom (see Claim 10.1). Notice that whether event bad occurs w.r.t. a commitment $\mathsf{NMCom}_{j \to i}$ for a corrupted $j \in I$ and an honest $i \in \bar{I}$ can be decided efficiently given the view of A and the value committed to in $\mathsf{NMCom}_{j \to i}$. Thus, it follows from robustness that for any $j \in I$ and $i \in \bar{I}$, the probabilities that bad occurs w.r.t. $\mathsf{NMCom}_{j \to i}$ in $\mathcal{H}_2$ and $\mathcal{H}_3$ differ only by a negligible amount. Combine this fact with Claim 10.12, we have that the probability that bad occur w.r.t. any such i, j is negligible.

Claim 10.15. *For every λ and inputs $\bar{x}$, the probability that event bad occurs in $\mathcal{H}_3$ is negligible.*

Again, we remark that we can reduce the above claims to the hiding and robustness of WI^{or}, despite of rewindings of rounds $k-3, k-2$. This is because, during rewindings, the first prover's messages of WI^{or} from honest parties can be emulated by internally running the prover's algorithm, which does not depend on statements nor witnesses (the second prover's messages are never sent in rewindings) thanks to the delayed-input property of WI^{or}. The receiver's messages of WI^{or} can be emulated by sending random coins, thanks to the public-coin property of WI^{or} is. Therefore, the WI^{or} proofs in the main thread are never rewound.

Hybrid $\mathcal{H}_4$: This hybrid proceeds identically to $\mathcal{H}_3$ except that after generating messages in round $k-1$ in the main thread, $\mathcal{H}_1$ keeps rewinding rounds $k-2, k-1$ to extract the inputs and random tapes of corrupted parties from the $\mathsf{WI}^{\mathrm{or},1}$ proofs given by A. Extraction is done identically as in Stage 4 of Sim, except that, the main thread here is simulated differently. In more detail, Sim simulates each rewinding as follows.

- *Replay* the honest parties' messages in round $k-1$ of Φ in the main thread.
- Generate honest parties' messages in other components $c_{i\to j}, \mathsf{NMZK}_{i\to j}, \mathsf{WI}^{\mathrm{or},2}_{i\to j}$ in rounds $k-2, k-1$ as done in the main thread of $\mathcal{H}_3/\mathcal{H}_4$. (Different from the main thread in Sim, here $c_{i\to j}$ commits to the input and random tape (x_i, r_i) of P_i used in Φ.)

Keep rewinding until obtaining another successful transcript trans'' of rounds $k-2, k-1$. For every WI proof $\mathsf{WI}^{\mathrm{or},1}_{j\to i}$ where prover is P_j is corrupted and the verifier P_i is honest, find two accepting transcripts of $\mathsf{WI}^{\mathrm{or},1}_{j\to i}$ with the same first two messages in the main thread and trans''. If they have the same third message, $\mathcal{H}_4$ aborts and outputs err_2; otherwise, $\mathcal{H}_4$ extracts a witness $w_{j\to i}$ by relying on the adaptive-input special soundness of $\mathsf{WI}^{\mathrm{or}}$. If any witness $w_{j\to i}$ is not a witness to the honest statement $X^{k-1}_{j\to i}$ defined in Equation 5, $\mathcal{H}_4$ aborts and outputs err_3; otherwise, $w_{j\to i}$ contains the decommitment $\rho_{j\to i}$ of $c_{j\to i}$ to input and random tape $(x_{j\to i}, r_{j\to i})$ that are consistent with P_j's messages in the first $k-2$ rounds of Φ. For corrupted $j \in I$, set (x_j, r_j) to $(x_{j\to i}, r_{j\to i})$ for an arbitrary honest $i \in \bar{I}$. Finally, after the last round k in main thread, $\mathcal{H}_4$ outputs err_4, if the honest parties do not abort, and for any $j \in I$, (x_j, r_j) is not consistent with P_j's last message in Φ.

We first bound the expected runtime of $\mathcal{H}_4$.

Claim 10.16. *There exists a universal polynomial $\mathcal{T}_4$, such that, for every λ and inputs $\bar{x}$, the run time of $\mathcal{H}_4$ is $\mathcal{T}_4(\mathcal{T}_A(\lambda))$ in expectation, where $\mathcal{T}_A(\lambda)$ is the runtime of A.*

Proof. Observe that hybrid $\mathcal{H}_3$ and $\mathcal{H}_4$ proceed identically, except that $\mathcal{H}_4$ performs rewindings in order to extract the inputs and random tapes of corrupted parties. Also observe that messages in each rewinding of rounds $k-2, k-1$ By are generated in a fixed polynomial time $\mathcal{T}'(\mathcal{T}_A)$ in the runtime of A. Therefore, by Claim 10.13, to bound the run-time of $\mathcal{H}_4$, it suffices to bound the expected number of rewindings of rounds $k-2, k-1$. This follows from the same argument as in 10.6.

Let ρ be any prefix of execution of $\mathcal{H}_4$ up to the end of round $k-3$ in the main thread. It is without loss of generality to assume that honest parties' random tapes for computing messages in the instance of Φ are sampled inside ρ. Let $p[\rho]$ be the probability that no party aborts in the first $k-1$ rounds in the main thread conditioned on ρ appearing. In each rewinding of rounds $k-2, k-1$, execution starts from ρ, and $\mathcal{H}_4$ simulates honest parties' messages exactly as in the main thread. In particular, since the honest parties' random tapes for running Φ are fixed in ρ, and round $k-2, k-1$ contains only round $k-2$ of Φ, honest parties' messages this round of Φ must be identical to that in the main thread. Thus, the probability that no party aborts in a rewinding is exactly $p[\rho]$. The expected number of rewindings is thus $p[\rho] \times 1/p[\rho] = 1$. □

We establish the following useful claim for $\mathcal{H}_4$.

Claim 10.17. *For every λ and inputs $\bar{x}$, the probability that $\mathcal{H}_4$ outputs err_2 or err_3 or err_4 is negligible.*

Proof. $\mathcal{H}_4$ outputs err_2 if for some $\mathsf{WI}^{\mathrm{or},1}_{j\to i}$ for $i \in \bar{I}, j \in I$, the two accepting transcripts obtained in the main and the successful rewinding threads have the same third messages. It follows from the same argument as the proof of Claim 10.7 that the probability that this occurs is negligible.

Next, suppose for contradiction that $\mathcal{H}_4$ outputs err_3 with noticeable probability. This means, for some $i \in \bar{I}, j \in I$, the witness $w_{j\to i}$ extracted from $\mathsf{WI}^{\mathrm{or}}_{j\to i}$ is not a witness for the honest statement with noticeable probability. By the soundness of $\mathsf{WI}^{\mathrm{or}}_{j\to i}$, $w_{j\to i}$ must be a valid witness of the fake statement, that is, the XOR of the share $s^0_{j\to i}$ committed in $\mathsf{NMCom}_{j\to i}$ and the share $s^1_{j\to i}$ is a trapdoor $\mathrm{td}_{i\to j}$ w.r.t. $\mathrm{vk}_{i\to j}$ with noticeable probability. However, note messages in main threads of $\mathcal{H}_3$ and $\mathcal{H}_4$ are generated identically. By Claim 10.15, event bad occurs with negligible probability in the main thread of $\mathcal{H}_4$, which gives a contradiction.

Finally, suppose for contradiction that $\mathcal{H}_4$ outputs err_4 with noticeable probability. This means honest parties do not abort after k rounds in the main thread, and for some corrupted j and honest i, $(x_j, r_j) = (x_{j\to i}, r_{j\to i})$ is not consistent with P_j's last message in Φ. By the above analysis, (x_j, r_j) is obtained from a witness $w_{j\to i}$ for the honest statement of $\mathsf{WI}^{\mathrm{or}}_{j\to i}$, and hence it is the value committed in $c_{j\to i}$. If (x_j, r_j) is not consistent with P_j's last message in Φ, the honest statement $X^k_{j\to i}$ of the second WI proof $\mathsf{WI}^{\mathrm{or},2}_{j\to i}$ from P_j is false. By the fact that honest parties accept the proof $\mathsf{WI}^{\mathrm{or},2}_{j\to i}$ at the end of round k and the soundness of the proof, the fake statement must be true. However, this would again contradict with Claim 10.15 that event bad occurs with only negligible probability. □

Note that the only difference between $\mathcal{H}_3$ and $\mathcal{H}_4$ is that the latter may output $\mathsf{err}_2, \mathsf{err}_3, \mathsf{err}_4$. Thus,

Claim 10.18. *The outputs of all parties in $\mathcal{H}_3$ and $\mathcal{H}_4$ are indistinguishable, that is,*

$$\{\mathcal{H}_3(1^\lambda, \bar{x})\}_{\lambda,\bar{x}} \approx \{\mathcal{H}_4(1^\lambda, \bar{x})\}_{\lambda,\bar{x}}$$

Hybrid $\mathcal{H}_5$: This hybrid is identical to $\mathcal{H}_4$, except for the following difference in both the main and rewinding threads:

- in $\mathcal{H}_4$, for every honest $i \in \bar{I}$ and $j \neq i$, the honest party P_i commits to its input and random tape (x_i, r_i) for Φ in $c_{i\to j}$, whereas,
- in $\mathcal{H}_5$, P_i commits to $0^{\mathrm{poly}(\lambda)}$ in $c_{i\to j}$.

By the same analysis of the expected runtime of $\mathcal{H}_1$ and $\mathcal{H}_4$, we have

Claim 10.19. *There exists a universal polynomial $\mathcal{T}_5$, such that, for every λ and inputs $\bar{x}$, the run time of $\mathcal{H}_5$ is $\mathcal{T}_5(\mathcal{T}_A(\lambda))$ in expectation, where $\mathcal{T}_A(\lambda)$ is the runtime of A.*

It follows from the hiding property of Com and the fact that the committer of Com sends only a single message that the following two claims hold.

Claim 10.20. *The outputs of all parties in $\mathcal{H}_4$ and $\mathcal{H}_5$ are indistinguishable, that is,*

$$\{\mathcal{H}_4(1^\lambda, \bar{x})\}_{\lambda,\bar{x}} \approx \{\mathcal{H}_5(1^\lambda, \bar{x})\}_{\lambda,\bar{x}} \ .$$

Claim 10.21. *For every λ and inputs $\bar{x}$, the probability that $\mathcal{H}_5$ outputs err_2 or err_3 or err_4 is negligible.*

Hybrid $\mathcal{H}_6$ This hybrid is identical to $\mathcal{H}_5$, except for the following difference in the main thread:

- in $\mathcal{H}_5$, messages in the delayed-semi-malicious MPC protocol Φ from the honest parties are generated honestly, whereas,
- in $\mathcal{H}_6$, these messages are simulated using the simulator Sim_Φ as Sim does.

By the same analysis of the expected runtime of $\mathcal{H}_1$ and $\mathcal{H}_4$, we have

Claim 10.22. *There exists a universal polynomial $\mathcal{T}_6$, such that, for every λ and inputs $\bar{x}$, the run time of $\mathcal{H}_6$ is $\mathcal{T}_6(\mathcal{T}_A(\lambda))$ in expectation, where $\mathcal{T}_A(\lambda)$ is the runtime of A.*

In both $\mathcal{H}_5$ and $\mathcal{H}_6$, the rewinding threads simply replay the messages of honest parties in Φ in the main thread. Therefore, the execution of Φ in the main thread is never rewound.

Claim 10.23. *The outputs of all parties in $\mathcal{H}_5$ and $\mathcal{H}_6$ are indistinguishable, that is,*

$$\{\mathcal{H}_5(1^\lambda, \bar{x})\}_{\lambda,\bar{x}} \approx \{\mathcal{H}_6(1^\lambda, \bar{x})\}_{\lambda,\bar{x}} \ .$$

Proof. To show this indistinguishability, we need to show that in $\mathcal{H}_5$, the adversary A with the extracted witness acts as a valid delayed-semi-malicious adversary. Then, by the delayed-semi-malicious security of Φ, the outputs of all parties remain indistinguishable when messages in Φ are simulated for A using Sim_Φ. Towards this, it suffices to show that with overwhelming probability, i) if the honest parties do not abort after round $k-2$ of Φ (that is, round $k-1$ of Π), then $\mathcal{H}_5$ must extract a witness w that is consistent with corrupted parties' messages in the first $k-2$ rounds, and ii) if honest parties do not abort after round $k-1$ of Φ (that is, round k of Π), then w must also be consistent with corrupted parties' messages in the last $k-1$ round. In this case, the execution of Φ in $\mathcal{H}_5$ corresponds to a real world execution of Φ with a delayed-semi-malicious adversary, where if the witness that the adversary sends is not valid, the honest parties abort immediately.

We show i) first. In $\mathcal{H}_5$, if the honest parties do not abort after round $k-2$ of Φ (that is, round $k-1$ of Π), $\mathcal{H}_5$ starts extraction. In this case, by Claim 10.21 that $\mathcal{H}_5$ outputs err_2 or err_3 with negligible probability, it must extract inputs and random tapes $\{x_j, r_j\}_{j \in I}$ consistent with corrupted parties' messages in the first $k-2$ rounds of Φ, with overwhelming probability.

We now show ii). In $\mathcal{H}_5$, if the honest parties do not abort after round $k-1$ of Φ (that is, round k of Π), by Claim 10.21 that $\mathcal{H}_5$ outputs err_4 with negligible probability, the extracted inputs and random tapes $\{x_j, r_j\}_{j \in I}$ must be consistent with corrupted parties' last messages of Φ, with overwhelming probability. □

Note that hybrid $\mathcal{H}_0$ and $\mathcal{H}_6$ proceed identically to the real world execution with A and the ideal world execution with A and Sim respectively. Therefore, Claim 10.22 directly shows that Sim runs in expected polynomial time (with black-box oracle access to A), which concludes Lemma 10.3. By a hybrid argument, we have that the outputs of all parties in the real and ideal worlds are indistinguishable, which concludes Lemma 10.4.

10.4 k-Round Malicious MPC from Our k-Round Delayed-Semi-Malicious MPC

In this section, we show that by leveraging specific properties of our delayed-semi-malicious MPC protocols, we can modify the transformation in the previous section to start with our protocols with k rounds, instead of $k-1$ rounds.

10.4.1 Construction of Malicious MPC

Tools Used in Our Malicious MPC Protocols. To construct a malicious MPC protocol for computing a functionality f, we rely on the same tools for as in Section 10.3, namely, a delayed-semi-malicious MPC protocol Φ' for computing f, a two-message statistically binding commitment scheme Com, and all the tools used in COSV NMZK (including the trapdoor setup protocol Trap, the special non-malleable commitment NMCom, and the special WI protocol $\mathsf{WI}^{\mathrm{or}}$). The only difference is that Φ' is our k-round delayed-semi-malicious MPC protocol for f, which satisfies the following properties.

- **Property 1:** The first $k-2$ rounds of Φ' contain only instances of OT. Let $\mathsf{OT}_{i\to j,\ell}$ denote the ℓ-th OT instance between P_i as the sender and P_j as the receiver, and $\{\mathsf{OT}_{i\to j,\ell}\}_{i,j\in[N],\ell\in[\mathrm{poly}(\lambda)]}$ the set of all OT instances executed in Φ'. The first $k-2$ rounds of Φ' contain only the first $k-2$ rounds of these OT instances.

- **Property 2:** Messages in the first $k-2$ rounds of Φ' does not depend on the input. Because of property 1 and the fact that OT can work with delayed inputs, that is, the OT receiver and the OT sender use their inputs only for generating the last message they send, namely, in round $k-1$ and round k respectively, only messages in round $k-1$ and k of Φ' depend on parties' inputs.

- **Property 3:** The simulator $\mathsf{Sim}_{\Phi'}$ of Φ' simulates honest parties' messages in the first $k-2$ rounds of Φ' by simply running the honest OT sender and receiver algorithms. Only honest parties' messages in round $k-1$ and k are simulated differently.

The Malicious MPC Protocol Π'. With these special properties, we now describe how to modify the construction of Π in Section 10.3 to use our k-round delayed-semi-malicious protocol Φ'. To compute f on inputs $x_1, \cdots, x_N$, parties $P_i(1^\lambda, x_i)$ run the same components as in Π, except that the first component is modified as follows:

- **Component 1, Computation by the delayed-semi-malicious MPC protocol Φ' (Round $1, \cdots, k$):**
  - For every OT instance $\mathsf{OT}_{i\to j,\ell}$ in Φ', if the $(k-2)$-th message in OT is from the sender (or receiver), P_i (or P_j respectively) samples a bit $b_{i\to j,\ell} \xleftarrow{R} \{0,1\}$ randomly and independently.
  - In the first $k-3$ rounds, for every OT instance $\mathsf{OT}_{i\to j,\ell}$ in Φ', run 2 independent OT instances $\mathsf{OT}^0_{i\to j,\ell}$ and $\mathsf{OT}^1_{i\to j,\ell}$ from P_i to P_j. (As OT works with delayed-inputs, these messages do not depend on parties' inputs.)
  - In round $k-2, k-1, k$, all parties jointly complete the execution of Φ' using inputs $\{x_i\}$, continuing from the OT instances $\{\mathsf{OT}^{b_{i\to j,\ell}}_{i\to j,\ell}\}$. The other OT instances $\{\mathsf{OT}^{1-b_{i\to j,\ell}}_{i\to j,\ell}\}$ are aborted in round $k-2$.

In summary, the instance of Φ' contains only $\{\mathsf{OT}^{b_{i\to j,\ell}}_{i\to j,\ell}\}$. Refer to them as the *real OT instances*, and the other aborted ones the *shadow OT instances*. As we will see shortly, the shadow OT instances are only used for simulation. Similar to before, let x_i, r_i be the input and random tape that P_i used in the instance of Φ'; m_i^ℓ and $\bar{m}^{<\ell} = \{m_j^{\ell'}\}_{j,\ell'<\ell}$ denote the messages broadcast by P_i and all parties in the first $\ell - 1$ rounds of Φ'.

- **Components 2-4** are essentially identical to that in Π, with the following slight modification:
  - In $c_{i\to j}$, P_i commits to (x_i, r_i') using Com in the first two rounds, where r_i' is the random tape P_i used for generating messages in all OT instances and Φ' (from round $k-2$). Note that P_i cannot commit to only its random tape used in Φ', because which OT instances are included in Φ' are only chosen in round $k-2 > 2$.
  - In $\mathsf{NMZK}_{i\to j}$ (consisting of $\mathsf{Trap}_{j\to i}, \mathsf{NMCom}_{i\to j}, s^1_{i\to j}, \mathsf{WI}^{\text{or},1}_{i\to j}$), P_i proves that its messages in the first $k-1$ rounds of Φ' are consistent with (x_i, r_i') committed in $c_{i\to j}$. Formally, statement $X^{k-1}_{i\to j}$ is in language $\mathcal{L}_X$, which are defined identically to Equation 5, with the underlined difference.

    $$\begin{aligned} &\forall\ X^t_{i\to j} = (t,\ \bar{m}^{\le t},\ c_{i\to j}), \qquad \mathcal{R}_X(X^t_{i\to j},\ \rho_{i\to j}) = 1 \\ &\quad \text{iff} \quad \text{i)} \quad \rho_{i\to j} \text{ decommits } c_{i\to j} \text{ to } x_i, r_i', \text{ and} \\ &\qquad\qquad \text{ii)} \quad \underline{r_i' \text{ contains } r_i} \text{ s.t. } \forall \ell \le t,\quad m_i^\ell = \mathsf{Next}_i^\Phi(1^\lambda, x_i, r_i, m^{\le \ell-1}) \end{aligned} \tag{7}$$

    Recall that inside $\mathsf{NMZK}_{i\to j}$, P_i proves in $\mathsf{WI}^{\text{or},1}_{i\to j}$ that either the above honest statement is true, or the following fake statement is true: The share $s^0_{i\to j}$ committed in $\mathsf{NMCom}_{i\to j}$ XOR'ed with $s^1_{i\to j}$ is a trapdoor w.r.t. $\text{vk}_{j\to i}$ in $\mathsf{Trap}_{j\to i}$. Formally, the statement $Y^{k-1}_{i\to j}$ is in $\mathcal{L}_Y$ defined in Equation (6) (w.r.t. the modified X statement above).
  - In $\mathsf{WI}^{\text{or},2}_{i\to j}$, P_i proves that either its messages in *all* k rounds of Φ' are consistent with (x_i, r_i') committed in $c_{i\to j}$ or the above fake statement is true. Formally, the statement $Y^k_{i\to j}$ is in language $\mathcal{L}_Y$.
- **Derive Outputs:** Each party derives its output from the instance of Φ'.

10.4.2 Proof of Malicious Security

Theorem 10.24. *Let f be any functionality. The above MPC protocol Π' for f is secure against malicious adversaries.*

We modify the simulator Sim for Π in Section 10.3.2 to obtain the simulator Sim' for Π'.

Overview of the Simulator Sim': At a high-level, Sim' proceeds identically to Sim, except for the following differences. First, Sim' simulate the shadow OT instances by simply running the honest OT sender and receiver's algorithms; as mentioned above, this does not require to use parties' real inputs. Second, recall that Π rewinds A at two places: It repeatedly rewinds rounds $k-3, k-2$ in order to extract a trapdoor $\text{td}_{j\to i}$ w.r.t. every verification key $\text{vk}_{j\to i}$ sent to an honest party P_i, and it rewinds rounds $k-2, k-1$ in order to extract the input and random tape (x_j, r_j) of every corrupted party P_j. For the security of the Φ instance to hold, it is important that its messages are never rewound. To achieve so, in these rewindings, Sim simply replays the messages in round

$k-3$ (when rewinding rounds $k-3, k-2$) and round $k-2$ (when rewinding rounds $k-2, k-1$) of Φ in the main thread for A. This works since Φ has only $k-1$ rounds, and the round $k-2$ of Π does not contain any Φ messages. Therefore, in these rewindings, the simulator never needs to simulate honest parties' message in response to new messages sent by A. In contrast, every round of Π' contains one round of Φ'. Sim' must avoid rewinding the Φ' instance in different ways.

- In each rewinding of rounds $k-3, k-2$, Sim' replays messages in round $k-3$ of Φ' in the main thread, and then uses the shadow OT instances to simulate messages in round $k-2$, that is, it generates the $(k-2)$-th message of $\{\mathsf{OT}_{i\to j,\ell}^{1-b_{i\to j,\ell}}\}$ for A. Observe that since $b_{i\to j,\ell}$ is chosen randomly and independently, the view of A in each rewinding is identically distributed as that in the main thread.

- In each rewinding of rounds $k-2, k-1$, Sim' replays messages in round $k-2$ of Φ' in the main thread. Then, it checks whether A sends the *same* messages in round $k-2$ as in the main thread, and aborts if not. Otherwise, it replays again messages in round $k-1$ of Φ' in the main thread. For this strategy to work, A is expected to send the same messages in round $k-2$ again with good probability. This follows because in round $k-1$, A must prove using NMZK that the corrupted parties' messages are consistent with the inputs and random tapes committed to in round $2 < k-2$. If A has good probability of passing the proof, then it has good probability of sending the same messages.

The new rewinding strategy does not rewind any messages in Φ'. Thus, its security holds.

The Simulator Sim': We now describe the modification to Sim formally.

- **Stage 1, Simulate main thread rounds** $1, \ldots, k-2$**:** Sim' simulates honest parties' messages as follows:

  1. Simulate messages in OT instances:
     - In the first $k-3$ rounds, emulate honest parties' messages in all OT instances $\{\mathsf{OT}^0_{i\to j,\ell}, \mathsf{OT}1_{i\to j,\ell}\}$ by running the honest OT sender and receiver algorithms.
     - In round $k-2$, for every pair $(\mathsf{OT}^0_{i\to j,\ell}, \mathsf{OT}1_{i\to j,\ell})$ whose $(k-2)$-th message is from an honest party P_i (or P_j), sample a bit $b_{i\to j,\ell} \xleftarrow{R} \{0,1\}$ at random, and emulate the honest party's next message in $\mathsf{OT}^{b_{i\to j,\ell}}_{i\to j,\ell}$ honestly and abort in $\mathsf{OT}^{1-b_{i\to j,\ell}}_{i\to j,\ell}$.
       For every pair $(\mathsf{OT}^0_{i\to j,\ell}, \mathsf{OT}1_{i\to j,\ell})$ whose next message is from a corrupted party, receive from A the next message of $\mathsf{OT}^{b_{i\to j,\ell}}_{i\to j,\ell}$ for bit $b_{i\to j,\ell}$ chosen by A.

     After round $k-2$, which OT instances are included in the instance of Φ' is determined. By the property of Φ', simulation of the first $k-2$ rounds of Φ' involves only generating OT messages honestly, exactly as done above.
  2. Simulate messages in $c_{i\to j}$, $\mathsf{NMZK}_{i\to j}$, $\mathsf{WI}^{\mathrm{or},2}_{i\to j}$, exactly as Sim does: Commit to a zero string in every $c_{i\to j}$ from an honest party P_i; emulate the first $k-2$ rounds of $\mathsf{NMZK}_{i\to j}$ and $\mathsf{WI}^{\mathrm{or},2}_{i\to j}$ honestly, which does not depend on any statements nor witnesses.

- **Stage 2, Rewind rounds $k-3, k-2$ to extract trapdoors:** Sim' keeps rewinding A from round $k-3$ to round $k-2$ in order to extract a trapdoor w.r.t. every verification key $\mathrm{vk}_{j\to i}$ sent to an honest party P_i. In each rewinding, it does:

1. Simulate the honest parties' messages in all OT instances as follows: In round $k-3$, *Replay* the honest parties' messages in all OT instances in the main thread. In round $k-2$, for every pair $(\mathsf{OT}^0_{i\to j,\ell}, \mathsf{OT}1_{i\to j,\ell})$ whose $(k-2)$-th message is from an honest party P_i (or P_j), *emulate the honest party's next message in* $\mathsf{OT}^{1-b_{i\to j,\ell}}_{i\to j,\ell}$ *honestly and abort in* $\mathsf{OT}^{b_{i\to j,\ell}}_{i\to j,\ell}$.
2. Simulate the honest parties' messages in other components $c_{i\to j}, \mathsf{NMZK}_{i\to j}, \mathsf{WI}^{\mathrm{or},2}_{i\to j}$ exactly as in the main thread.

Sim' keeps rewinding until it obtains another successful transcript trans' of round $k-3$ and $k-2$. For every $\mathrm{vk}_{j\to i}$ sent to an honest party P_i, it finds a trapdoor $\mathrm{td}_{j\to i}$, and outputs err_1 if it fails.

- **Stage 3, Simulate main thread round $k-1$ using trapdoors:** Sim' simulates honest parties' messages in round $k-1$ as Sim does: 1) It simulates messages in the instance of Φ' using the simulator $\mathsf{Sim}'_{\Phi'}(1^\lambda, I)$ of Φ', 2) simulates the third messages in $\{\mathsf{WI}^{\mathrm{or},2}_{i\to j}\}$ honestly, and 3) cheats in $\{\mathsf{NMZK}_{i\to j}\}$ by proving the fake statements.

- **Stage 4, Rewind rounds $k-2, k-1$ to extract inputs and random tapes of corrupted parties:** Sim' keeps rewinding A from round $k-2$ to $k-1$ in order to extract a witness $w_{j\to i}$ from every WI proof $\mathsf{WI}^{\mathrm{or},1}_{j\to i}$ where the prover P_j is corrupted and the verifier P_i is honest. In each rewinding, it does:

  1. Simulate the honest parties' messages in Φ' as follows: In round $k-2$, *replay* the honest parties' messages in round $k-2$ of Φ' in the main thread. Let $\bar{m}^{k-2}_I = \{m^{k-2}_j\}_{j\in I}$ be the messages that corrupted parties sent in round $k-2$ of Φ' in the main-thread. If A sends messages different from $\bar{m}^{k-2}_I$, abort this rewinding. Otherwise, in round $k-1$, *replay* the honest parties' messages in round $k-1$ of Φ' in the main thread.
  2. Simulate the honest parties' messages in other components $\mathsf{NMZK}_{i\to j}, \mathsf{WI}^{\mathrm{or},2}_{i\to j}$ exactly as in the main thread.

  Sim keeps rewinding until it obtains another successful transcript trans'' of round $k-2$ and $k-1$. For every WI proof $\mathsf{WI}^{\mathrm{or},1}_{j\to i}$ where the prover P_j is corrupted and the verifier P_i is honest, it extracts a witness, and outputs err_2 if extraction fails. If any witness $w_{j\to i}$ is not a witness to the honest statement $X^{k-1}_{j\to i}$ defined in Equation (7), it aborts and outputs err_3. Otherwise, it finds in $w_{j\to i}$ for an arbitrary honest $i \in \bar{I}$, an input and random tape (x_j, r_j) consistent with P_j's messages in the first $k-1$ rounds of Φ'.

- **Stage 5, Simulate main thread round k using extracted input and randomness:** Using the corrupted parties' inputs and random tapes $\{x_j, r_j\}_{j\in I}$ extracted from A, Sim' simulates honest parties' messages in round k as Sim does: 1) it simulates the last messages of Φ' using $\mathsf{Sim}'_{\Phi'}$, fed with the witness $w = \{x_j, r_j\}_{j\in I}$, and 2) cheats in $\{\mathsf{WI}^{\mathrm{or},2}_{i\to j}\}$ by proving the fake statement.

  Finally, Sim' outputs err_4 (as Sim does), if the honest parties do not abort in the main thread, and for some $j \in I$, (x_j, r_j) obtained in Stage 4 is not consistent with P_j's last message in Φ'.

To establish the correctness of Sim', we show that

Lemma 10.25. *For any N-party functionality f, and any poly-sized adversary A controlling any subset $I \subset [N]$ of parties. The simulator Sim' for A described above runs in expected polynomial time,* except for a negligible probability.

Lemma 10.26. *For any N-party functionality f, and any poly-sized adversary A controlling any subset $I \subset [N]$ of parties. The simulator Sim' for A described above satisfies that:*

$$\left\{\mathsf{Ideal}_{I,\mathsf{Sim}'}(1^\lambda, \bar{x})\right\}_{\lambda,I,\bar{x}} \approx \left\{\mathsf{Real}_{I,A}(1^\lambda, \bar{x})\right\}_{\lambda,I,\bar{x}} \ .$$

We remark that Lemma 10.25 only establishes that Sim' runs in some polynomial $\mathcal{T}$ time in expectation, with overwhelming probability $1-\mu(\lambda)$. But, we can easily modify Sim' to remove the negligible probability $\mu(\lambda)$: Simply cut off it computation after $1/\mu(\lambda)$ steps. Now, the expected polynomial time is $\mathcal{T} \times (1-\mu(\lambda)) + (1/\mu(\lambda))\mu(\lambda) \leq \mathcal{T} + 1$. Therefore, it suffices to prove the above two lemmas.

Correctness of Simulation — Proof of Lemma 10.25 and 10.26 To show the correctness of Sim', we use essentially the same hybrids $\mathcal{H}'_0, \cdots, \mathcal{H}'_6$ used for showing the correctness of Sim. The main difference lies in the simulation strategy in the rewindings of rounds $k-3, k-2$ and round $k-2, k-1$, and the running time analysis of the hybrids. Below, we describe the hybrids; we focus more on the part that is different from the hybrids in the proof for Sim, and describe briefly the parts that are similar or identical.

Hybrid $\mathcal{H}'_0$: This hybrid is identical to the real world execution $\mathsf{Real}_{I,A}(1^\lambda, \bar{x})$. It runs in strict polynomial time, and it follows from the same proof of Claim 10.5 that event bad almost never occurs in this hybrid.

Hybrid $\mathcal{H}'_1$: This hybrid is identical to $\mathcal{H}'_0$ except that after generating messages in the first $k-2$ rounds in the main thread, $\mathcal{H}'_1$ keeps rewinding rounds $k-3, k-2$ to extract a trapdoor w.r.t. every verification key $\mathrm{vk}_{j\to i}$ sent to an honest party P_i. Procedure of extraction is identical to that in Stage 2 of Sim', except that now the main thread is honestly generated. In more detail, $\mathcal{H}'_1$ simulates in a rewinding as follows.

- *Replay* the honest parties' messages in round $k-3$ of all OT instances in the main thread. In round $k-2$, for every pair $(\mathsf{OT}^0_{i\to j,\ell}, \mathsf{OT1}_{i\to j,\ell})$ whose $(k-2)$-th message is from an honest party, *emulate the honest party's next message in* $\mathsf{OT}^{1-b_{i\to j,\ell}}_{i\to j,\ell}$ *honestly and abort in* $\mathsf{OT}^{b_{i\to j,\ell}}_{i\to j,\ell}$.
- Generate honest parties' messages in other components $c_{i\to j}, \mathsf{NMZK}_{i\to j}, \mathsf{WI}^{\mathrm{or},2}_{i\to j}$ in rounds $k-3, k-2$ as done in the main thread. (Different from the main thread of Sim', here $c_{i\to j}$ commits to the input and random tape (x_i, r'_i) of P_i used in all OT instances and Φ'.)

Keep rewinding until obtaining another successful transcript trans' of rounds $k-3, k-2$. For every $\mathrm{vk}_{j\to i}$ sent to an honest party P_i, it finds a trapdoor $\mathrm{td}_{j\to i}$, and outputs err_1 if it fails.

We first argue that this hybrid runs in expected polynomial time.

Claim 10.27. *There exists a universal polynomial $\mathcal{T}_1$, such that, for every λ and inputs $\bar{x}$, the run time of $\mathcal{H}'_1$ is $\mathcal{T}_1(\mathcal{T}_A(\lambda))$ in expectation, where $\mathcal{T}_A(\lambda)$ is the runtime of A.*

Proof. The analysis of run-time is identical to that for Claim 10.6. To bound the expected run-time of $\mathcal{H}_1'$, it suffices to bound the expected number of rewindings performed for extracting the trapdoor. In Claim 10.6, we showed that if the views of A in the main and each rewinding thread are identically distributed, then the expected number of rewindings is 1, following the "$p \times 1/p$" argument.

Let ρ be any prefix of execution of $\mathcal{H}_1'$ up to the end of round $k-4$ in the main thread. It is without loss of generality to assume that honest parties' messages in round $k-3$ of all OT instances are already generated in ρ, and would be sent immediately after ρ. Let $p[\rho]$ be the probability that no party aborts in the first $k-2$ rounds in the main thread conditioned on ρ appearing. Conditioned on ρ, honest parties' messages are simulated identically in the main and each rewinding thread, except that, in the main thread, round $k-2$ contains honest messages in the OT instances $\{\mathsf{OT}_{i\to j,\ell}^{b_{i\to j,\ell}}\}$, whereas in rewinding, it contains honest messages in OT instances $\{\mathsf{OT}_{i\to j,\ell}^{1-b_{i\to j,\ell}}\}$. However, for each instance $\mathsf{OT}_{i\to j,\ell}$ whose $(k-2)$-th message is sent by an honest party, the bit $b_{i\to j,\ell}$ is sampled by that honest party at random. Therefore, the views of A in the main and rewinding thread are still identically distributed. Thus, by the "$p[\rho] \times 1/p[\rho]$" argument, the expected number of rewindings is at most 1. □

Given that $\mathcal{H}_1'$ runs in expected polynomial time, it follows from the same arguments for Claim 10.7 that $\mathcal{H}_1$ outputs err_1 with negligible probability. Since the only difference in the main threads of $\mathcal{H}_0'$ and $\mathcal{H}_1'$ is that the latter may output err_1, we have that: The views of all parties in the main threads of $\mathcal{H}_0'$ and $\mathcal{H}_1'$ are statistically close, and event bad occurs in $\mathcal{H}_1'$ with negligible probability.

Hybrid $\mathcal{H}_2'$: This hybrid proceeds identically to $\mathcal{H}_1'$ except for the following difference in the main thread:

- in $\mathcal{H}_1'$ for every honest $i \in \bar{I}$ and $j \neq i$, the party P_i commits to a random share $s_{i\to j}^0$ in $\mathsf{NMCom}_{i\to j}$ and sends another random share $s_{i\to j}^1$ in the clear, but
- in $\mathcal{H}_2'$, share $s_{i\to j}^1$ is set to $s_{i\to j}^0 \,\mathsf{xor}\, \mathrm{td}_{j\to i}$, where $\mathrm{td}_{j\to i}$ is the extracted trapdoor for $\mathrm{vk}_{j\to i}$.

Since $\mathcal{H}_1'$ and $\mathcal{H}_2'$ proceed identically till the end of round $k-2$ in the main thread, their expected run-time are identical up to this point. After this point, both hybrids run in a fixed polynomial time in $\mathcal{T}_A$. Hence,

Claim 10.28. *There exists a universal polynomial $\mathcal{T}_2$, such that, for every λ and inputs $\bar{x}$, the run time of $\mathcal{H}_2'$ is $\mathcal{T}_2(\mathcal{T}_A(\lambda))$ in expectation, where $\mathcal{T}_A(\lambda)$ is the runtime of A.*

Alternatively, we can view the difference in $\mathcal{H}_1'$ and $\mathcal{H}_2'$ as follows: In $\mathcal{H}_1'$, the shares are $s_{i\to j}^0 = u_{i\to j}$ and $s_{i\to j}^1 = u_{i\to j}' \;\mathsf{xor}\; \mathrm{td}_{j\to i}$ for randomly sampled $u_{i\to j}, u_{i\to j}'$, whereas in $\mathcal{H}_2'$, they are $s_{i\to j}^0 = u_{i\to j}'$ and $s_{i\to j}^1 = u_{i\to j}' \;\mathsf{xor}\; \mathrm{td}_{j\to i}$. Therefore, by hiding of NMCom, the outputs of all parties in $\mathcal{H}_1', \mathcal{H}_2'$ are indistinguishable. Furthermore, by non-malleability of NMCom, the probabilities that bad occurs in $\mathcal{H}_1', \mathcal{H}_2'$ differ by at most a negligible amount. Since bad almost never occurs in $\mathcal{H}_1'$, the same holds in $\mathcal{H}_2'$.

Hybrid $\mathcal{H}_3'$: This hybrid proceeds identically to $\mathcal{H}_2'$ except for the following difference in the main thread:

- in $\mathcal{H}'_2$, for every honest $i \in \bar{I}$ and $j \neq i$, the honest party P_i proves the honest statement in the WI proofs $\mathsf{WI}^{\mathrm{or},1}_{i\to j}, \mathsf{WI}^{\mathrm{or},2}_{i\to j}$, whereas
- in $\mathcal{H}'_3$, P_i proves the fake statement that the share $s^0_{i\to j}$ committed in $\mathsf{NMCom}_{i\to j}$ XORed with $s^1_{i\to j}$ is a valid trapdoor w.r.t. $\mathrm{vk}_{j\to i}$.

Since the WI proofs use delayed inputs. $\mathcal{H}'_2$ and $\mathcal{H}'_3$ proceed identically till the end of round $k-2$ in the main thread. After this point, both hybrids run in a fixed polynomial time. Hence,

Claim 10.29. *There exists a universal polynomial $\mathcal{T}_3$, such that, for every λ and inputs $\bar{x}$, the run time of $\mathcal{H}'_3$ is $\mathcal{T}_3(\mathcal{T}_A(\lambda))$ in expectation, where $\mathcal{T}_A(\lambda)$ is the runtime of A.*

By the witness indistinguishability of $\mathsf{WI}^{\mathrm{or}}$, we have that the outputs of all parties in $\mathcal{H}'_2, \mathcal{H}'_3$ are indistinguishable. Furthermore, by the robustness of $\mathsf{WI}^{\mathrm{or}}$ w.r.t. NMCom (see Claim 10.1) and the fact that bad occurs with negligible probability in $\mathcal{H}'_1$, the same holds in $\mathcal{H}'_2$.

Hybrid $\mathcal{H}'_4$: This hybrid proceeds identically to $\mathcal{H}'_3$ except that after round $k-1$ in the main thread, $\mathcal{H}'_1$ keeps rewinding round $k-2, k-1$ to extract the inputs and random tapes of corrupted parties from the $\mathsf{WI}^{\mathrm{or},1}$ proofs given by A. Extraction is done identically as in Stage 4 of Sim'. In more detail, Sim' simulates each rewinding as follows.

- In round $k-2$, *replay* the honest parties' messages in the Φ' instances in the main thread. If A sends messages $\bar{m}'^{k-2}_I$ different from the messages it sent $\bar{m}^{k-2}_I$ in the main thread, abort this rewinding. Otherwise, in round $k-1$, *replay* again the honest parties' messages in round $k-1$ of the Φ' instance in the main thread.
- Simulate the honest parties' messages in other components $c_{i\to j}, \mathsf{NMZK}_{i\to j}, \mathsf{WI}^{\mathrm{or},2}_{i\to j}$ in rounds $k-2, k-1$ as done in the main thread of $\mathcal{H}'_3/\mathcal{H}'_4$. (Different from the main thread in Sim', here $c_{i\to j}$ commits to the input and random tape (x_i, r'_i) of P_i used in all OT instances and Φ'.)

Keep rewinding until obtaining another successful transcript trans'' of rounds $k-2, k-1$. For every WI proof $\mathsf{WI}^{\mathrm{or},1}_{j\to i}$ from a corrupted prover P_j to an honest verifier P_i, $\mathcal{H}'_4$ extracts a witness $w_{j\to i}$ by relying on the adaptive-input special soundness of $\mathsf{WI}^{\mathrm{or}}$. If extraction fails, $\mathcal{H}'_4$ aborts and outputs err_2, and if $w_{j\to i}$ is not a witness to the honest statement $X^{k-1}_{j\to i}$ defined in Equation 7, it aborts and outputs err_3. Otherwise, it finds (x_j, r_j) consistent with P_j's messages in the first $k-1$ rounds of Φ'. Finally, after the last round k in main thread, $\mathcal{H}'_4$ outputs err_4, if the honest parties do not abort, and for some $j \in I$, (x_j, r_j) is not consistent with P_j's last message in Φ'.

We first bound the expected run-time of $\mathcal{H}'_4$.

Claim 10.30. *There exists a universal polynomial $\mathcal{T}_4$, such that, for every λ and inputs $\bar{x}$, the run time of $\mathcal{H}'_4$ is $\mathcal{T}_4(\mathcal{T}_A(\lambda))$ in expectation, except for a negligible probability μ_4, where $\mathcal{T}_A(\lambda)$ is the runtime of A.*

Proof. Observe that hybrid $\mathcal{H}'_3$ and $\mathcal{H}'_4$ proceed identically, except that $\mathcal{H}'_4$ performs rewindings in order to extract the inputs and random tapes of corrupted parties. By the fact that $\mathcal{H}'_3$ runs in expected polynomial time and that each rewinding takes a strict polynomial time, to bound the expected run-time of $\mathcal{H}'_4$, it suffices to bound the expected number rewindings of round $k-2, k-1$.

First, observe that the main threads of $\mathcal{H}'_3, \mathcal{H}'_4$ are simulated identically. Since bad almost never occurs in $\mathcal{H}'_3$, neither does it occur in $\mathcal{H}'_4$. We show that the following event bad' also almost never occurs in the main thread of $\mathcal{H}'_4$.

- Fix an arbitrary honest party $i^\star \in \bar{I}$, say the smallest i in $\bar{I}$.
  Event bad' occurs in a thread, if the following conditions hold.
  - *i)* No party aborts in the first $k-1$ rounds.
  - *ii)* There exists a corrupted party P_j, such that, its messages m_j^{k-2} in round $k-2$ of Φ' are *inconsistent* with the *unique* input and random tape (x_j, r'_j) committed to in $c_{j\to i^\star}$. (By the statistical binding property of Com, the value committed in $c_{j\to i^\star}$ is unique with overwhelming probability.) Inconsistency means that m_j^{k-2} is different from the messages $m_j^{\star,k-2}$ computed according to the appropriate random coins in r'_j and messages in Φ'.

If bad occurs with negligible probability, then bad' occurs with negligible probability. Condition i) implies that every corrupted party P_j convinces $P_{i^\star}$ in $\mathsf{NMZK}_{j\to i^\star}$. When bad does not occur, by the soundness of $\mathsf{WI}^{\mathrm{or},1}_{j\to i^\star}$ in $\mathsf{NMZK}_{j\to i^\star}$, the honest statement must be true, which implies that P_j's message m_j^{k-2} in round $k-2$ of Φ' is *consistent* with (x_j, r'_j) committed in $c_{j\to i^\star}$, with overwhelming probability. Let $\mu = \mu(\lambda)$ be the negligible probability that bad' occurs in the main thread of $\mathcal{H}'_4$.

Let ρ be any prefix of execution of $\mathcal{H}'_4$ up to the end of round $k-3$ in the main thread. It is without loss of generality to assume that honest parties' random tapes for computing messages in the instance of Φ' are sampled inside ρ. Let $p[\rho]$ denote the probability that no party aborts in the first $k-1$ rounds in the main thread, conditioned on ρ appearing. By an averaging argument, for a $(1-\sqrt{\mu})$ fraction of ρ, conditioned on ρ appearing, the probability that bad' occurs in the main thread is at most $\sqrt{\mu}$. Let Γ be this set of ρ.

Recall that in $\mathcal{H}'_4$. A rewinding is *successful* if (1) no party aborts in the first $k-1$ rounds, and (2) messages sent by corrupted parties in round $k-2$ of Φ' are identical to that $\bar{m}_I^{k-2}$ they sent in the main thread. We observe that in a rewinding in $\mathcal{H}'_4$, honest parties' messages in round $k-1$ are simulated identically as in the main thread. In particular, since the honest parties' random tapes for running Φ' are fixed inside ρ, their messages in round $k-2$ of Φ' must be identical to that in the main thread. (Other components are also simulated identically.) Moreover, if (2) holds in a rewinding, $\mathcal{H}'_4$ continues to simulate the honest parties' messages in round $k-1$ identically as in the main thread. In particular, if corrupted parties indeed send $\bar{m}_I^{k-2}$ again in round $k-2$, honest parties' responses in round $k-1$ must also be identical to that in the main thread. On the other hand, when (2) does not occur, $\mathcal{H}'_4$ aborts this rewinding.

Consider a hypothetical hybrid $\mathcal{H}''_4$ where when (2) occurs, it does not abort the rewinding and continues to simulate messages in round $k-1$ as in the main thread. (The rest is identical to $\mathcal{H}'_4$.) Still, a rewinding is successful only if both conditions (1) and (2) hold. Observe that the probability that a rewinding is successful is identical in $\mathcal{H}'_4$ and $\mathcal{H}''_4$, since $\mathcal{H}'_4$ only aborts when (2) does not occur. Therefore, they have the same number of rewindings. The advantage of considering $\mathcal{H}_4''$ is that its main thread and rewinding threads are simulated identically. In particular, if conditioned on ρ, the probability that bad' occurs is bounded by $\sqrt{\mu}$ in the main thread, it is also bounded by $\sqrt{\mu}$ in each rewinding.

Next, we bound the expected number of rewindings in $\mathcal{H}_4''$, when the following bad events $\mathsf{overtime}_1$ and $\mathsf{overtime}_2$ do not happen.

- Event $\mathsf{overtime}_1$ occurs if a prefix $\rho \notin \Gamma$ appears or bad' occurs in the main thread. We have that

$$\Pr[\mathsf{overtime}_1] \;\le\; \Pr\left[\,\rho \notin \Gamma\,\right] \;+\; \Pr\left[\,\mathsf{bad}' \text{ in main } \wedge\; \rho \in \Gamma\,\right] \;\le\; 2\sqrt{\mu}\ .$$

- Event $\mathsf{overtime}_2$ occurs if a prefix ρ satisfying that $p[\rho] \le 2\sqrt{\mu}$ appears, and no party aborts in the first $k-1$ rounds in the main thread.

$$\Pr\left[\,\mathsf{overtime}_2\,\right] = \Pr\left[\,p[\rho] \le 2\sqrt{\mu}\,\right] \times \Pr\left[\,\text{no abort in main } : \; p[\rho] \le 2\sqrt{\mu}\,\right] \le 2\sqrt{\mu}\ .$$

We now show that conditioned on $\mathsf{overtime}_1, \mathsf{overtime}_2$ not occurring, the expected number of rewindings is at most 4.

Towards the above, first observe that

$$\begin{aligned} R \;&= \mathrm{E}\left[\,\#\text{ rewindings } : \; \neg\mathsf{overtime}_1 \;\wedge\; \neg\mathsf{overtime}_2\,\right] \\ &= \mathrm{E}\left[\,\#\text{ rewindings } : \; \rho \in \Gamma \;\wedge\; (\neg\mathsf{bad}' \text{ in main}) \;\wedge\; ((p[\rho] > 2\sqrt{\mu}) \vee (\text{abort in main}))\,\right] \end{aligned}$$

Consider two cases, either the main thread is aborted, or not. By Bayesian law,

$$\begin{aligned} R \;\le\; &\mathrm{E}\left[\,\#\text{ rewindings } : \; \rho \in \Gamma \;\wedge\; (\neg\mathsf{bad}' \text{ in main}) \;\wedge\; \text{ abort in main }\,\right] \\ &+\; \mathrm{E}\left[\,\#\text{ rewindings } : \; \rho \in \Gamma \;\wedge\; (\neg\mathsf{bad}' \text{ in main}) \;\wedge\; (p[\rho] > 2\sqrt{\mu}) \;\wedge\; (\neg\text{abort in main})\,\right] \\ &\times\; \Pr\left[\,\neg\text{abort in main } : \; \rho \in \Gamma \;\wedge\; (\neg\mathsf{bad}' \text{ in main}) \;\wedge\; ((p[\rho] > 2\sqrt{\mu}) \vee (\text{abort in main}))\,\right] \end{aligned}$$

The first expectation is simply 0 as when the main thread is aborted, no rewinding is performed. Moreover, the last probability is at most $2p[\rho]$. To see this, consider two cases, either $\rho \in \Gamma$ satisfies that $p[\rho] \le 2\sqrt{\mu}$; then it must happen that the main thread is aborted, and hence the probability of not aborting is zero. Or $\rho \in \Gamma$ satisfies that $p[\rho] > 2\sqrt{\mu}$; conditioned on such a ρ, and conditioned further on bad' not occurring in main, the probability that the main thread is not aborted is at most $p[\rho]/(1-\sqrt{\mu}) \le 2p[\rho]$, as for $\rho \in \Gamma$, the probability that bad' occurs is at most $\sqrt{\mu} \le 1/2$. Therefore,

$$R \le \mathrm{E}\left[\,\#\text{ rewindings } : \; \rho \in \Gamma \;\wedge\; (\neg\mathsf{bad}' \text{ in main}) \;\wedge\; (p[\rho] > 2\sqrt{\mu}) \;\wedge\; (\neg\text{abort in main})\,\right] \times 2p[\rho]$$

The expected number of rewindings is the inverse of the probability that a rewinding is successful. Recall that a rewinding is successful when 1) no party aborts in the first $k-1$ rounds and 2) the corrupted parties send the same messages in $k-2$ round of Φ' as in the main thread. Conditioned on any $\rho \in \Gamma$ satisfying $p[\rho] > 2\sqrt{\mu}$, the main and rewinding threads are simulated identically and independently in $\mathcal{H}_4''$. Therefore, the probability that 1) holds in a rewinding is $p[\rho]$. We claim that 2) occurs whenever bad' does not occur in the rewinding, which happens with probability at least $1-\sqrt{\mu}$. Since bad' does not occur in the main thread, if it also does not occur in a rewinding, the corrupted parties' messages in round $k-2$ of Φ'

are determined by the inputs and random tapes committed to in $\{c_{j\to i^\star}\}_{j\in I}$ in the first two rounds contained in ρ, and hence 2) must occur. Therefore, the probability that a rewinding is successful is at least $p[\rho] - \sqrt{\mu} > p[\rho]/2$. Combined this with the above equation, we have:

$$R \leq \frac{p[\rho]}{2} \times 2p[\rho] = 4$$

We conclude that except with negligible probability $\mu_4 = 4\sqrt{\mu}$, the expected number of rewindings is bounded by 4 in $\mathcal{H}_4''$, as well as in $\mathcal{H}_4'$. Therefore, the claim holds. □

Given that the $\mathcal{H}_4'$ runs in expected polynomial time with overwhelming probability, by the same argument of Claim 10.17, the probability that it fails to extract a witness from some $\mathsf{WI}^{\mathrm{or},1}_{j\to i}$ from a corrupted party is negligible, i.e., $\mathcal{H}_4'$ outputs err_2 with negligible probability. Moreover, by the fact that bad almost never occurs in the main thread of $\mathcal{H}_4'$ and the soundness of $\mathsf{WI}^{\mathrm{or},1}_{j\to i}$, the extracted witness must be a witness of the honest statement with overwhelming probability, i.e., $\mathcal{H}_4'$ outputs err_3 with negligible probability. Therefore, $\mathcal{H}_4'$ can recover inputs and random tapes that explain corrupted parties' messages in the first $k-1$ rounds. By the same argument and the soundness of $\mathsf{WI}^{\mathrm{or},2}_{j\to i}$ proofs, the recovered inputs and random tapes must also be consistent with messages in round k, if the honest parties do not abort, i.e., $\mathcal{H}_4'$ outputs err_4 with negligible probability.

As the only difference between $\mathcal{H}_3'$ and $\mathcal{H}_4'$ is that the latter may output $\mathsf{err}_2, \mathsf{err}_3, \mathsf{err}_4$, which occurs with only negligible probability, we have that the outputs of all parties in $\mathcal{H}_3'$ and $\mathcal{H}_4'$ are statistically close.

Hybrid $\mathcal{H}_5'$: This hybrid is identical to $\mathcal{H}_4'$, except for the following difference in both the main and rewinding threads:

- in $\mathcal{H}_4'$, for every honest $i \in \bar{I}$ and $j \neq i$, the honest party P_i commits to its input and random tape (x_i, r_i') for all OT instances and Φ in $c_{i\to j}$, whereas,
- in $\mathcal{H}_5'$, P_i commits to $0^{\mathrm{poly}(\lambda)}$ in $c_{i\to j}$.

We show that the hiding of Com implies that like $\mathcal{H}_4'$, $\mathcal{H}_5'$ runs in expected polynomial time except with negligible probabilities.

Claim 10.31. *There exists a universal polynomial $\mathcal{T}_5$, such that, for every λ and inputs $\bar{x}$, the run time of $\mathcal{H}_5'$ is $\mathcal{T}_5(\mathcal{T}_A(\lambda))$ in expectation, except for a negligible probability $\mu_5(\lambda)$, where $\mathcal{T}_A(\lambda)$ is the runtime of A.*

Proof. Fix a λ and inputs $\bar{x}$. By Claim 10.30, $\mathcal{H}_4'$ runs in expected polynomial time $\mathcal{T} = \mathcal{T}_4(\mathcal{T}_A)$, except with negligible probability μ_4.

Suppose for contradiction that the claim is false.

- Let $\mathcal{T}^\top$ be the largest time step such that conditioned on $\mathcal{H}_5'$ not running for more than $\mathcal{T}^\top$ steps, the expected run-time of $\mathcal{H}_5'$ is bounded by $2\mathcal{T}$. In other words, for any $\mathcal{T}' > \mathcal{T}^\top$, the expected run-time of $\mathcal{H}_5'$ conditioned on not running for more than $\mathcal{T}'$ steps is larger than $2\mathcal{T}$. By the contradiction hypothesis, we have that the probability that $\mathcal{H}_5'$ runs more than $\mathcal{T}^\top$ steps is some inverse polynomial, denoted as $p^\top$.

Since $p^\top$ is inverse polynomial, we argue that $\mathcal{T}^\top$ must be a polynomial. Suppose not and $\mathcal{T}^\top$ is superpolynomial. Let $\mathcal{H}_5[\mathcal{T}^\top]$ be a hybrid identical to $\mathcal{H}_5'$ except that the execution is cut off after $\mathcal{T}^\top$ steps. Since the expected run-time of $\mathcal{H}_5[\mathcal{T}^\top]$ is bounded by $2\mathcal{T}$, the probability that it runs for more than $\mathcal{T}^\perp = 2\mathcal{T}/p^\top$ steps is at most $p^\top$. As $\mathcal{T}^\top$ is superpolynomial and $\mathcal{T}^\perp$ is polynomial, it holds: i) The probability that $\mathcal{H}_5'$ (without cut off) runs for more than $\mathcal{T}_\perp$ steps is also at most $p^\top$, and ii) this gives a contradiction with the contradiction hypothesis that $\mathcal{H}_5$ runs for more than $\mathcal{T}^\top$ steps with probability at least $p^\top$.

Consider now $\mathcal{H}_5[\mathcal{T}^\top + 1]$ and $\mathcal{H}_4[\mathcal{T}^\top + 1]$, where executions of $\mathcal{H}_5', \mathcal{H}_4'$ are cut off after $\mathcal{T}^\top + 1$ steps. They both run in strict polynomial time. The expected run-time of the former is greater than $2\mathcal{T}$, but the expected run-time of the latter is no greater than $\mathcal{T} + 1$. To see the latter, let $\mathsf{overtime}$ denote the probability μ_4 event where the expected run-time of $\mathcal{H}_4'$ is not bounded.

$$\begin{aligned} &\mathrm{E}\left[\text{Runtime of } \mathcal{H}_4[\mathcal{T}^\top + 1]\right] \\ \leq\ &\mathrm{E}\left[\text{Runtime of } \mathcal{H}_4' \ :\ \neg\mathsf{overtime}\right] \times (1-\mu_4) + (\mathcal{T}^\top + 1) \times \mu_4 \ \leq\ \mathcal{T} + 1 \end{aligned}$$

Since the only difference between $\mathcal{H}_5[\mathcal{T}^\top + 1]$ and $\mathcal{H}_4[\mathcal{T}^\top + 1]$ lies in what values are committed to in the Com commitments. We can use them to build a malicious strict poly-time receiver A' of Com. A' internally runs $\mathcal{H}_4''$ (or $\mathcal{H}_5''$) for a sufficiently large polynomial number of times. Whenever an honest party P_i needs to send a Com commitment in the internal emulation, A' chooses two appropriate challenge messages (x_i, r_i'), $0^{\mathrm{poly}(\lambda)}$, and upon receiving a Com commitment to one of the messages, it forwards this commitment to A' as the commitment from P_i. All other messages are generated identically as in $\mathcal{H}_4''$. Finally, A' outputs the average run-time of all executions. Note that depending on the values committed in the Com commitments, A' perfectly emulates many executions of $\mathcal{H}_4''$ or $\mathcal{H}_5''$. When the number of executions is sufficiently large, in the former case, the average run-time output by A' must be smaller than $3T/2$ with overwhelming probability, and in the latter case, the average run-time must be larger than $3T/2$ with overwhelming probability. This violates hiding of Com and gives a contradiction. □

Given that the expected runtime of $\mathcal{H}_4', \mathcal{H}_5'$ are both polynomial with overwhelming probability. It follows from the hiding property of Com and the fact that the committer of Com sends only a single message that the outputs of all parties in $\mathcal{H}_4'$ and $\mathcal{H}_5'$ are indistinguishable, and the probability that $\mathcal{H}_5'$ outputs err_2 or err_3 or err_4 is almost identical to that $\mathcal{H}_4'$ outputs any of them, which is negligible.

Hybrid $\mathcal{H}_6'$ This hybrid is identical to $\mathcal{H}_5'$, except for the following difference in the main thread:

- in $\mathcal{H}_5'$, messages in our delayed-semi-malicious MPC protocol Φ' from the honest parties are generated honestly, whereas,
- in $\mathcal{H}_6'$, these messages are simulated using the simulator $\mathsf{Sim}'_{\Phi'}$ as Sim' does.

In both $\mathcal{H}_5'$ and $\mathcal{H}_6'$, the rewinding threads simply replay the honest parties' messages in the instance of Φ' in the main thread. Therefore, the instance of Φ' in the main thread is never rewound. By the same analysis of the expected runtime of $\mathcal{H}_5'$ (Claim 10.31), but now relying on the security of Φ', we have

Claim 10.32. *There exists a universal polynomial $\mathcal{T}_6$, such that, for every λ and inputs $\bar{x}$, the run time of $\mathcal{H}'_6$ is $\mathcal{T}_6(\mathcal{T}_A(\lambda))$ in expectation, except for a negligible probability $\mu_6(\lambda)$, where $\mathcal{T}_A(\lambda)$ is the runtime of A.*

Finally, again by the security of Φ', the outputs of all parties in $\mathcal{H}'_5$ and $\mathcal{H}'_6$ are indistinguishable.

Note that hybrid $\mathcal{H}'_0$ and $\mathcal{H}'_6$ are identical to the real world execution with A and the ideal world execution with A and Sim' respectively. Therefore, Claim 10.32 directly shows that Sim' runs in expected polynomial time (with black-box oracle access to A), except with negligible probability, which concludes Lemma 10.25. By a hybrid argument, we have that the outputs of all parties in the real and ideal worlds are indistinguishable, which concludes Lemma 10.26.

Acknowledgments

The authors thank Yuval Ishai, Antigoni Polychroniadou, and Stefano Tessaro for helpful discussions.

This work was supported by NSF grants CNS-1528178, CNS-1514526, CNS-1652849 (CAREER), a Hellman Fellowship, the Defense Advanced Research Projects Agency (DARPA) and Army Research Office (ARO) under Contract No. W911NF-15-C-0236, and a subcontract No. 2017-002 through Galois. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

References

[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. Springer, Heidelberg, August 2017. (Pages 4, 6, 17, and 18.)

[AJL$^+$12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, April 2012. (Pages 4, 5, 6, 17, 20, 25, and 58.)

[BBR97] Eli Biham, Dan Boneh, and Omer Reingold. Generalized Diffie-Hellman modulo a composite is not weaker than factoring. Cryptology ePrint Archive, Report 1997/014, 1997. `http://eprint.iacr.org/1997/014`. (Page 5.)

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001. (Page 7.)

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16*, pages 1292–1303. ACM Press, October 2016. (Page 4.)

[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. Springer, Heidelberg, May 2017. (Page 4.)

[BGI$^+$18] Elette Boyle, Niv Gilboa, Yuval Ishai, Huijia Lin, and Stefano Tessaro. Foundations of homomorphic secret sharing. To appear, ITCS, 2018. (Page 4.)

[BGJS16] Saikrishna Badrinarayanan, Vipul Goyal, Aayush Jain, and Amit Sahai. Verifiable functional encryption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 557–587. Springer, Heidelberg, December 2016. (Page 13.)

[BHP17] Zvika Brakerski, Shai Halevi, and Antigoni Polychroniadou. Four round secure computation without setup. Cryptology ePrint Archive, Report 2017/386, 2017. http://eprint.iacr.org/2017/386. (Pages 4, 5, and 6.)

[BLSV17] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous ibe, leakage resilience and circular security from new assumptions. Cryptology ePrint Archive, Report 2017/967, 2017. https://eprint.iacr.org/2017/967. (Page 7.)

[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. Springer, Heidelberg, August 1990. (Page 5.)

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513. ACM Press, May 1990. (Page 4.)

[BP16] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key FHE with short ciphertexts. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 190–213. Springer, Heidelberg, August 2016. (Page 4.)

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. (Page 6.)

[CDG$^+$17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 33–65. Springer, Heidelberg, August 2017. (Pages 7, 10, 19, and 21.)

[CGP15] Ran Canetti, Shafi Goldwasser, and Oxana Poburinnaya. Adaptively secure two-party computation from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 557–585. Springer, Heidelberg, March 2015. (Page 4.)

[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. ACM Press, May 2002. (Page 6.)

[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. IEEE Computer Society Press, October 2010. (Page 4.)

[CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 630–656. Springer, Heidelberg, August 2015. (Page 4.)

[COSV17] Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Delayed-input non-malleable zero knowledge and multi-party coin tossing in four rounds. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 711–742. Springer, Heidelberg, November 2017. (Pages 18, 60, and 61.)

[DG17] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 537–569. Springer, Heidelberg, August 2017. (Pages 7, 10, 19, and 21.)

[DKR15] Dana Dachman-Soled, Jonathan Katz, and Vanishree Rao. Adaptively secure, universally composable, multiparty computation in constant rounds. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 586–613. Springer, Heidelberg, March 2015. (Page 4.)

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013. (Page 7.)

[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. Springer, Heidelberg, February 2014. (Pages 4 and 7.)

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013. (Pages 6 and 10.)

[GKM$^+$00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st FOCS*, pages 325–335. IEEE Computer Society Press, November 2000. (Page 5.)

[GKP$^+$13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 536–553. Springer, Heidelberg, August 2013. (Page 11.)

[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. Springer, Heidelberg, August 2015. (Pages 4, 7, 10, and 19.)

[GMPP16] Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 448–476. Springer, Heidelberg, May 2016. (Page 4.)

[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. ACM Press, May 1987. (Pages 4 and 7.)

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004. (Page 23.)

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, Heidelberg, May / June 2006. (Page 19.)

[GP15] Sanjam Garg and Antigoni Polychroniadou. Two-round adaptively secure MPC from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 614–637. Springer, Heidelberg, March 2015. (Page 4.)

[GPR16] Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 1128–1141. ACM Press, June 2016. (Page 60.)

[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. (Pages 4, 5, 7, and 19.)

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 469–477. ACM Press, June 2015. (Page 13.)

[Hai08] Iftach Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 412–426. Springer, Heidelberg, March 2008. (Page 26.)

[HK12] Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *Journal of Cryptology*, 25(1):158–193, January 2012. (Page 6.)

[LRY16] Benoît Libert, Somindu C. Ramanna, and Moti Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *ICALP 2016*, volume 55 of *LIPIcs*, pages 30:1–30:14. Schloss Dagstuhl, July 2016. (Page 13.)

[McC88] Kevin S. McCurley. A key distribution system equivalent to factoring. *Journal of Cryptology*, 1(2):95–105, 1988. (Page 5.)

[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. Springer, Heidelberg, May 2016. (Page 4.)

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991. (Page 60.)

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *12th SODA*, pages 448–457. ACM-SIAM, January 2001. (Page 6.)

[PS16] Chris Peikert and Sina Shiehian. Multi-key FHE from LWE, revisited. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 217–238. Springer, Heidelberg, October / November 2016. (Page 4.)

[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. Springer, Heidelberg, August 2008. (Page 5.)

[Shm85] Z. Shmuely. Composite Diffie-Hellman Public-Key Generating Systems are Hard to Break. Technical report, Technion, 1985. (Page 5.)

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982. (Page 4.)

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. (Page 4.)

[YZ16] Yu Yu and Jiang Zhang. Cryptography with auxiliary input and trapdoor from constant-noise LPN. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 214–243. Springer, Heidelberg, August 2016. (Page 5.)