Secure Computation with Parallel Calls to 2-ary Functions

Varun Narayanan[1], Shubham Vivek Pawar[2], and Akshayaram Srinivasan[3] *

[1] University of California, Los Angeles, USA
varunnkv@gmail.com
[2] Royal Holloway, University of London, UK
shubham.pawar.2022@live.rhul.ac.uk
[3] University of Toronto, Canada
akshayaram@cs.toronto.edu

Abstract. Reductions are the workhorses of cryptography. They allow constructions of complex cryptographic primitives from simple building blocks. A prominent example is the non-interactive reduction from securely computing a "complex" function f to securely computing a "simple" function g via randomized encodings.

Prior work equated simplicity with functions of small degree. In this work, we consider a different notion of simplicity where we require g to only take inputs from a small number of parties. In other words, we want the arity of g to be as small as possible.

In more detail, we consider the problem of reducing secure computation of arbitrary functions to secure computation of functions with arity two (two is the minimal arity required to compute non-trivial functions). Specifically, we want to compute a function f via a protocol that makes parallel calls to 2-ary functions. We want this protocol to be secure against malicious adversaries that could corrupt an arbitrary number of parties. We obtain the following results:

- **Negative Result:** We show that there exists a degree-2 polynomial p such that no protocol that makes parallel calls to 2-ary functions can compute p with statistical security with abort.
- **Positive Results:** We give two ways to bypass the above impossibility result.
  1. **Weakening the Security Notion.** We show that every degree-2 polynomial can be computed with statistical privacy with knowledge of outputs (PwKO) by making parallel calls to 2-ary functions. Privacy with knowledge of outputs is weaker than security with abort.
  2. **Computational Security.** We prove that for every function f, there exists a protocol for computing f that makes parallel calls to 2-ary functions and achieves security with abort

* V. Narayanan was NSF grants CNS-2246355, CCF-2220450, US-Israel BSF grant 2022370, and by Sunday Group; S. V. Pawar was supported by EPSRC through grant number EP/S021817/1; A. Srinivasan was supported in part by a NSERC Discovery grant RGPIN-2024-03928.

against computationally-bounded adversaries. The security of this protocol relies on the existence of semi-honest secure oblivious transfer.

- **Applications:** We give connections between this problem and the task of reducing the encoding complexity of Multiparty Randomized Encodings (MPRE) (Applebaum, Brakerski, and Tsabary, TCC 2018). Specifically, we show that under standard computational assumptions, there exists an MPRE where the encoder can be implemented by an NC^0 circuit with constant fan-out.
- **Extensions:** We explore this problem in the honest majority setting and give similar results assuming one-way functions. We also show that if the parties have access to 3-ary functions then we can construct a computationally secure protocol in the dishonest majority setting assuming one-way functions.

1 Introduction

A key research direction in theoretical cryptography is to construct complex cryptographic primitives from simple building blocks. This direction has achieved remarkable success and has led to several fundamental results such as constructing pseudorandom generators [HILL99] and zero-knowledge proofs [GMW86] from one-way functions, secure multiparty computation protocols from oblivious transfer [GMW87, Kil88, IPS08], and CCA-secure encryption from injective trapdoor functions [HKW20]. One such foundational result is a non-interactive reduction from securely computing a "complex" function f to securely computing a "simple" function g. This is achieved using randomized encodings [Yao86, IK00, AIK04] and this approach has been instrumental in constructing round-optimal secure computation protocols [Yao86, GS18, BL18].

Prior work aimed to minimize the degree of g as much as possible and they equated simplicity with functions computable by constant degree polynomials. [IK00, AIK04] showed that under standard cryptographic assumptions, securely computing any function can be reduced to securely computing a degree-3 function. [BL18, GS18, GIS18, ACGJ18, ABT18, ABT19, ACGJ19] (using the notion of multiparty randomized encoding) showed that one can further reduce the (effective) degree to 2 if we allow local pre-processing of the inputs by the parties.

Our Work. We consider a different notion of simplicity. Specifically, we want to reduce the task of securely computing some complex n-party function f to securely computing, in parallel, a set of functions where each function g_S takes inputs only from a proper subset S of the parties (in other words, the arity $|S|$ of g_S is less than n) and delivers the output to *all parties*. It is trivial to see that most multiparty party functions—for instance, n-party AND—cannot be computed, using a single function of arity less than n. So, we need to necessarily allow parallel calls to several such functions. Our goal is to minimize their arity as much as possible.

Our Model. To be more precise, let f be an n-party function and, for a collection $\mathcal{S}$ of k-sized subsets of $[n]$, let $\{g_S\}_{S\in\mathcal{S}}$ be a set of functions, where each g_S is an arity-k function taking inputs from parties in the set S and delivers output to all parties. We say that f securely reduces to $\{g_S\}$ if there is a tuple of randomized algorithms $(\mathsf{Enc}, \mathsf{Dec})$ with the following syntax and satisfying the following two properties.

- **Syntax:** Enc is a randomized function that takes in the index i of a party, its private input x_i and a subset $S \in \mathcal{S}$ such that $i \in S$ and outputs $x_{i,S}$. Dec takes in $\left\{g_S(\{x_{i,S}\}_{i\in S})\right\}_{S\in\mathcal{S}}$ and outputs y.[4]
- **Correctness:** We want the output y of Dec to be $f(x_1, \ldots, x_n)$.
- **Security:** Even if an arbitrary subset of the parties get corrupted by an adversary, we want $\left\{g_S(\{x_{i,S}\}_{i\in S})\right\}_{S\in\mathcal{S}}$ to only reveal the output $f(x_1, \ldots, x_n)$ and nothing else about the private inputs of the uncorrupted parties.

Minimal Arity. If we consider reduction to arity-1 functions, then it is easy to see that the only functions that can be securely computed are of the form $(h_1(x_1), h_2(x_2), \ldots, h_n(x_n))$. Hence, to be useful, we need functions with arity 2 or higher.

Why is arity important? Consider the task of securely computing some complex multiparty function f involving a large number of parties. In such a case, it is unreasonable to expect all the parties to be online throughout the entire protocol execution. However, if we can break this complex computation to simple components of constant arity, then we only need a constant number of parties to be online for computing every component. Further, the computation of each component is not dependent on the outputs from other components (since we only make parallel calls to the functions). Once all the components are computed, the parties can apply the local decoding procedure to learn the output of the function f.

Case of Semi-Honest Adversaries. In the semi-honest model, there is an easy transformation from securely computing a degree-d function (that has an one-to-one correspondence with degree-d polynomials) to making parallel calls to d-ary functions. Indeed, we can compute each monomial of the degree-d polynomial using an d-ary function and add additive secret shares of 0 to each monomial to ensure that only the output of the polynomial is revealed. Hence, the existing results about the completeness of degree-2 functions [GS18, GIS18, ACGJ18, ABT18, ABT19, ACGJ19] (with local pre-processing) can be extended to our

[4] We model Dec in this way so that a party that does not contribute any private input could still learn the output. This is analogous to the notion of output client in the client-server MPC protocol literature [DI05] and is equivalent to considering publicly decodable transcripts [ABG+20].

model of making parallel calls to 2-ary functions. Somewhat surprisingly, these results do not extend to the malicious setting and this is the focus of this work.

1.1 Our Results

We explore the problem of securely computing n-party functions against malicious adversaries by making parallel calls to 2-ary functions. As argued earlier, two is the minimum arity needed to compute non-trivial functions. We provide both positive and negative results and connect this problem to reducing the encoding complexity of multiparty randomized encodings [ABT18]. We also provide a couple of extensions to these results by (i) considering the honest majority setting, and (ii) allowing the arity to be larger than two.

Impossibility of Statistical Security with Abort. Our first result shows that if we want statistical security, then it is impossible to compute even degree-2 polynomials. Specifically, we give a 3-party function f (that is computable by a degree-2 polynomial) such that no protocol that makes parallel calls to 2-ary functions achieves statistical security with abort. Formally,

Theorem 1.1. *There exists a 3-party function f that can be computed using a degree-2 polynomial such that no 3-party protocol making parallel calls to 2-ary functions can compute f with statistical security with abort against a malicious adversary corrupting two parties.*

In the case of three parties with two corruptions, security with abort is equivalent to security with selective abort. Therefore, our impossibility result also rules out constructions with selective abort.

Positive Results. We give two ways to overcome the above impossibility.

1. We show that if we relax the security requirement to privacy with knowledge of outputs (PwKO) [IKP10], then every degree-2 polynomial can be securely computed with statistical PwKO by making parallel calls to 2-ary functions. PwKO relaxation guarantees that the adversary does not learn anything about the private inputs of the honest parties except the output of the function but after learning the output, it can force the honest parties to output an arbitrary value of its choice. Formally, this is modelled by having the ideal functionality first give the output of the function to the simulator and the simulator sends some arbitrary value as the output to the honest parties. All the honest parties will output the value provided by the simulator.

   Theorem 1.2. *For every n-party function f computable using degree-2 polynomials, there exists a protocol for computing f that makes parallel calls to 2-ary functions and achieves statistical PwKO against a malicious adversary that could corrupt an arbitrary number of parties.*

2. A second approach to bypass the impossibility result is to relax the security to be computational. We show how to extend Theorem 1.2 to securely computing arbitrary circuits and achieve security with unanimous abort.

   Theorem 1.3. *Assume the existence of a semi-honest secure oblivious transfer protocol. For every n-party function f, there exists a protocol for computing f that makes parallel calls to 2-ary functions and achieves security with unanimous abort against a computationally-bounded malicious adversary that could corrupt an arbitrary number of parties.*

Application: Reducing the Complexity of MPRE. Theorem 1.3 has an interesting application to reducing the complexity of the encoding function of Multiparty Randomized Encoding (MPRE) [GS17, ABT18, ABT19]. MPRE is the analog of randomized encodings for distributed computation protocols. In a bit more detail, MPRE for computing a multiparty function f comprises of n pre-processing functions $h_1, \ldots, h_n$ along with an encoder Enc' and a decoder Dec'. The party P_i first applies the local pre-processing function h_i on its private input and randomness. After this, we apply the encoding function Enc' on the pre-processed values. The decoding function Dec' takes in the output of the encoder and computes the output of f applied on the private inputs of all the parties. For security, we require that even if an adversary corrupts an arbitrary number of parties, the output of the encoder only reveals the output of f and nothing else about the inputs of the honest parties.

An important research direction in the study of randomized encodings is to minimize the encoding complexity as much as possible [AIK04, AIK07]. Previous results [GS18, ABT18, GIS18] gave constructions of MPRE where the encoder could be implemented in NC^0. However, the fan-out of the encoder circuit grew linearly with the number of parties. But using Theorem 1.3, we can construct an MPRE where the encoding can be done in NC^0 with constant fan-out. In other words, the encoder has constant input and output locality. This encoder is optimal in the sense that its has constant fan-in, constant fan-out, and constant depth.

This result is obtained directly from the above theorem by replacing the 2-ary functions with an existing MPRE secure against malicious adversaries [GS18]. This MPRE construction is based on the existence of a two-round, malicious-secure oblivious transfer protocol. Since this MPRE computes a two-party functionality, its fan-out is constant and hence, the overall fan-out of the encoder is constant. This is illustrated in Figure 1.1. As a result, we get the following corollary.

Corollary 1.4. *Assume the existence of a two-round oblivious transfer protocol secure against malicious adversaries. Then, there is a reduction from securely computing any multiparty function f against malicious adversaries to securely computing a NC^0 function with constant fan-out against malicious adversaries.*

Extensions. We provide the following extensions.

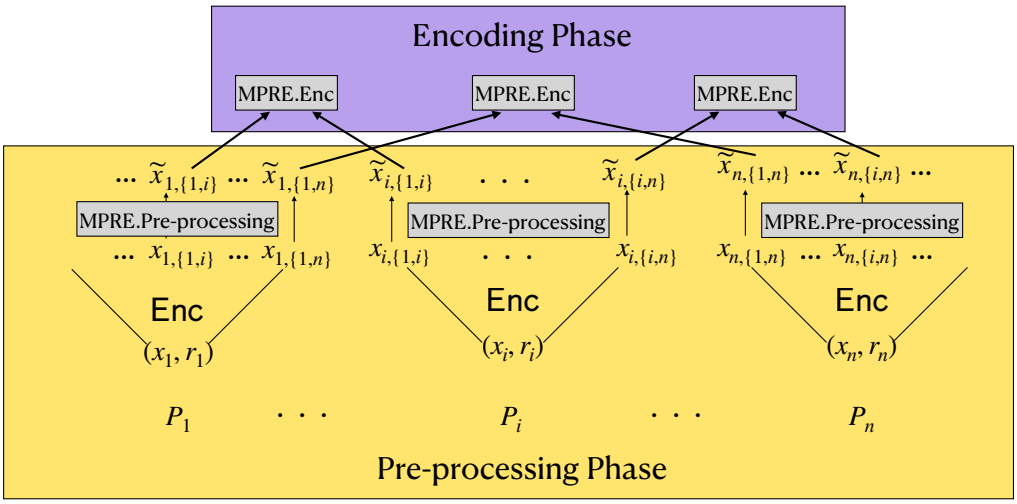

Fig. 1.1. Construction of MPRE where the encoder has constant input and output locality. Here, Enc denotes the encoder function from Theorem 1.3 and MPRE is for computing the 2-ary function from Theorem 1.3.

– **Honest Majority:** When security is required in the presence of an honest majority, we can replace the assumption of semi-honest secure oblivious transfer with the weaker assumption of one-way function.

  Theorem 1.5. *Assume the existence of one-way functions. For every n-party function f, there exists a protocol for computing f that makes parallel calls to 2-ary functions and achieves security with unanimous abort against a computationally-bounded malicious adversary that could corrupt a strict minority of parties.*

– **3-ary functions.** We further pursue the objective of realizing computational security with unanimous abort against an arbitrary number of malicious corruptions based only on the existence of one-way functions. We show that this is possible if the parties have access to 3-ary instead of 2-ary functions.

  Theorem 1.6. *Assume the existence of one-way functions. For every n-party function f, there exists a protocol for computing f that makes parallel calls to 3-ary functions and achieves security with unanimous abort against a computationally-bounded malicious adversary that could corrupt an arbitrary number of parties.*

1.2 Related Work

Fitzi et al. [FGMO01] considered the problem of constructing secure computation protocols in the dishonest majority setting with full security (i.e., guaranteed

output delivery) with help of functions that take inputs from less than n parties. Without the help of additional functions, achieving fairness (which is weaker than full security) is impossible [Cle86]. Fitzi et al. showed a negative result that it is impossible to achieve full security with access to functions of arity less than n. See [IPP+22] for a detailed discussion on the works constructing fully-secure protocols with calls to functions with arity n.

Applebaum and Goel [AG21] based on Baum et al.'s result [BOSS20] gave a black-box protocol for computing any multiparty function by making parallel calls to constant degree functions and satisfying security with identifiable abort. This constant degree function takes inputs from all the parties. Their results do not extend to our setting where we restrict the function to take inputs only from two parties.

Applebaum et al. [ABG+20] showed that if we restrict our problem to only include 2-ary functions that give outputs to the two parties that provide inputs, then it is impossible to achieve statistical security even against semi-honest adversaries. Specifically, they gave a 3-party function that cannot be computed by any protocol that makes parallel calls to such 2-ary functions with statistical security against semi-honest adversaries.

2 Technical Overview

In Section 2.1, we give the main intuition behind our impossibility result (see Theorem 1.1). In Section 2.2, we show that if one relaxes the security requirement to privacy with knowledge of outputs [IKP10], then every degree-2 polynomial can be computed with parallel calls to 2-ary functions (see Theorem 1.2). In Section 2.3, we explain how to securely compute arbitrary circuits with (unanimous) abort by relaxing to computational security. This assumes the existence of semi-honest secure oblivious transfer (see Theorem 1.3).

2.1 Impossibility of Achieving Statistical Security

The key intuition behind the impossibility result is that a corrupt party can send inconsistent inputs in its interaction (via the 2-ary functions) with two different honest parties. This would allow the adversary to learn additional information about the private inputs of the honest parties that is not learnable in the ideal world. However, formalizing this intuition requires great care as we need to choose an appropriate function for which the impossibility holds and also provide a formal attack that works against any protocol. We elaborate on this below.

Consider the 3-party function f that takes x_1, x_2, and x_3 belonging to $\{0, 1, 2\}$ from P_1, P_2 and P_3, respectively and outputs 1 iff $x_1 + x_2 = x_3 \bmod 3$. We will first argue that f cannot be computed by making parallel calls to 2-ary functions. We will then show how to extend this impossibility to a function that is computable by a degree-2 polynomial.

Towards a contradiction, assume that there exists a non-interactive protocol $\Pi = (\mathsf{Enc}, \{\mathcal{O}_{\{1,2\}}, \mathcal{O}_{\{2,3\}}, \mathcal{O}_{\{1,3\}}\}, \mathsf{Dec})$ that securely computes f against ma-

licious adversaries with abort. Here, $\mathcal{O}_{\{i,j\}}$ denotes the 2-ary function that is invoked between parties P_i and P_j for each distinct $i, j \in [3]$.

Consider an adversary $\mathcal{A}_{1,2}$ that corrupts P_1 and P_2, and makes the calls to $\mathcal{O}_{\{1,3\}}$ and $\mathcal{O}_{\{2,3\}}$ by honestly emulating P_1 and P_2 after setting their inputs to x_1 and x_2 respectively. Let $y_{\{1,3\}}$ and $y_{\{2,3\}}$ be the outputs of $\mathcal{O}_{\{1,3\}}$ and $\mathcal{O}_{\{2,3\}}$ respectively. At this point, the adversary can learn $f(x_1, x_2, x_3)$ without even invoking $\mathcal{O}_{\{1,2\}}$. This can be obtained by honestly computing the output of $\mathcal{O}_{\{1,2\}}$ in its head and applying the decoder on the outputs of the functions. The output is guaranteed to be $f(x_1, x_2, x_3)$ by correctness of Π.

The crucial lemma in the impossibility is that for any $y'_{\{1,2\}}$ in the range of $\mathcal{O}_{\{1,2\}}$, the output of $\mathsf{Dec}(y'_{\{1,2\}}, y_{\{2,3\}}, y_{\{1,3\}})$ is either $\perp$ or $f(x_1, x_2, x_3)$. Otherwise, the adversary can simultaneously learn $f(x_1, x_2, x_3)$ (by computing this in its head) and force honest P_3 to output a value other than $\perp$ or $f(x_1, x_2, x_3)$ (by sending $y'_{\{1,2\}}$ as the output of $\mathcal{O}_{\{1,2\}}$). This is impossible in the ideal world, thereby, contradicting security of Π.

Appealing to another adversary $\mathcal{A}_{1,3}$ that corrupts P_1 and P_3, and behaves analogously to $\mathcal{A}_{1,2}$, we can argue that, for any $y'_{\{1,3\}}$ in the range of $\mathcal{O}_{\{1,3\}}$, the output of $\mathsf{Dec}(y_{\{1,2\}}, y_{\{2,3\}}, y'_{\{1,3\}})$ is either $\perp$ or $f(x_1, x_2, x_3)$, when $y_{\{1,2\}}$ and $y_{\{2,3\}}$ are, respectively, the outputs of $\mathcal{O}_{\{1,2\}}$ and $\mathcal{O}_{\{2,3\}}$, when corrupt P_1 and P_3 behave honestly.

Finally, consider an adversary $\mathcal{A}_1$ that only corrupts P_1 and behaves as follows. $\mathcal{A}_1$ samples x_1 uniformly and invokes $\mathcal{O}_{\{1,2\}}$ by emulating P_1 with input x_1, whereas, it invokes $\mathcal{O}_{\{1,3\}}$ by emulating P_1 honestly but with its input set to $x_1 + 1 \bmod 3$. In essence, $\mathcal{A}_1$ provides inconsistent inputs while emulating $\mathcal{O}_{\{1,2\}}$ and $\mathcal{O}_{\{1,3\}}$.

Let $y_{\{1,2\}}$, $y_{\{2,3\}}$, and $y_{\{1,3\}}$ be the outputs of $\mathcal{O}_{\{1,2\}}$, $\mathcal{O}_{\{2,3\}}$ and $\mathcal{O}_{\{1,3\}}$, respectively in the real execution of the protocol with $\mathcal{A}_1$. $\mathcal{A}_1$ then chooses $y'_{\{1,3\}}$ and $y'_{\{1,2\}}$ in the range of $\mathcal{O}_{\{1,3\}}$ and $\mathcal{O}_{\{1,2\}}$ respectively such that $z' = \mathsf{Dec}(y'_{\{1,2\}}, y_{\{2,3\}}, y_{\{1,3\}}) \neq \perp$ and $z = \mathsf{Dec}(y_{\{1,2\}}, y_{\{2,3\}}, y'_{\{1,3\}}) \neq \perp$, and outputs (z, z'). Note that such $y'_{\{1,2\}}$ and $y'_{\{1,3\}}$ will always exist as we can set them to be the outputs of honest emulations of $\mathcal{O}_{\{1,2\}}$ and $\mathcal{O}_{\{1,3\}}$ with inputs $x_1 + 1$ and x_1 respectively. The adversary simply chooses the first string in the range of these two functions that produces a non-bot output. This is where we use the fact that the adversary is computationally unbounded as such strings might not be efficiently computable.

We claim that Π is not secure against $\mathcal{A}_1$. By our previous two lemmas, since z and z' are not $\perp$, it has to be the case that $z = f(x_1, x_2, x_3)$ and $z' = f(x_1 + 1, x_2, x_3)$. Thus, $\mathcal{A}_1$ simultaneously learns the output of the functions for two possible inputs of P_1. This allows $\mathcal{A}$ to learn the value of $x_3 - x_2$ with absolute certainty. However, when x_2 and x_3 are chosen uniformly at random, an ideal adversary will fail to guess $x_2 - x_3$ with constant probability and thus, we obtain a contradiction to the security of Π.

Making f to be degree-2. We observe that the same impossibility proof extends if we consider the function f that checks if $x_1 + x_2 + x_3 = 0$ where the $+$ operation is over $\mathsf{GF}(2^2)$. This can be expressed as a degree-2 polynomial over $\{0, 1\}$.

2.2 Securely Computing Degree-2 Polynomials with PwKO

The previous impossibility crucially relied on the honest party either outputting the correct function output or $\perp$. However, this impossibility result does not extend to the case where the adversary could force the honest party to output an arbitrary value. Therefore, there is hope of obtaining a protocol that satisfies statistical privacy with knowledge of outputs (PwKO). However, such a protocol is highly non-trivial to construct and it needs new techniques. We now explain our approach to construct such a protocol for computing arbitrary degree-2 polynomials.

For simplicity, let's assume that each party P_i gets a finite field element x_i as its private input. Let $p(\cdot)$ be a degree-2 polynomial and the parties want to compute $p(x_1, \ldots, x_n) = \sum_{i,j \in [n]} c_{i,j} \cdot x_i \cdot x_j$. Our goal is to design a protocol for computing p that makes parallel calls to 2-ary functions and achieves PwKO against a malicious adversary that corrupts an arbitrary subset of the parties. In the rest of the overview, we will use $\mathcal{O}_{\{i,j\}}$ to denote the function that is computed using P_i and P_j's inputs.[5]

The starting point of our construction is the semi-honest secure protocol for computing p with parallel calls to 2-ary functions. In this semi-honest secure protocol, we define $\mathcal{O}_{\{i,j\}}$ to take $(x_i, s_i[j])$ from P_i and $(x_j, s_j[i])$ from party P_j and to output $c_{i,j} \cdot x_i \cdot x_j + s_i[j] + s_j[i]$ to every party. For every $i \in [n]$, if P_i chooses $\{s_i[j]\}_{j \in [n]}$ as a random secret sharing of 0, then the parties can add the outputs of all the 2-ary functions to obtain $p(x_1, \ldots, x_n)$. The security follows since $\{s_i[j]\}_{j \in [n]}$ are all random subject to their sum being 0 and thus, only $p(x_1, \ldots, x_n)$ is revealed. The key challenge is to extend this protocol to be secure against malicious adversaries.

If we analyze the protocol a bit more carefully, we realize that a malicious adversary can only mount two kinds of attacks:

- **Sending inconsistent inputs.** An adversary corrupting P_i could send $(x_i, \cdot)$ to $\mathcal{O}_{\{i,j\}}$ and $(x'_i, \cdot)$ (where $x_i \neq x'_i$) to $\mathcal{O}_{\{i,j'\}}$ for two different parties j, j'.
- **Generating bad secret shares.** An adversary that is corrupting party P_i could generate $\{s_i[j]\}_{j \in [n]}$ as secret shares of a value other than 0.

Let's assume for now that the adversary is restricted to sending consistent private inputs to each 2-ary function. In other words, it is disallowed from mounting the first attack strategy explained above. However, note that this adversary is still allowed to generate $\{s_i[j]\}_{j \in [n]}$ as shares of a value other than 0. If that

[5] For simplicity, we allow each pair of parties to potentially invoke a different function $\mathcal{O}_{\{i,j\}}$. However, this can be easily modified to compute a single function g that takes (i, j) as additional input.

is the case, we argue that the above protocol already satisfies PwKO. At a high-level, if the real-world adversary generates $\{s_i[j]\}_{j\in[n]}$ to be shares of a value other than 0, then it can be shown that this attack corresponds to adding an offset to the actual output. Thus, we can have the simulator (in the ideal world) to add the same offset to the output obtained from the ideal functionality and make all the honest parties obtain this modified output.

The next step to consider is what happens if the adversary is allowed to send inconsistent inputs. Can the same protocol be proved to satisfy PwKO? We explain that this is not the case. First, observe that if the adversary sends x_i to $\mathcal{O}_{\{i,j\}}$ and x'_i to $\mathcal{O}_{\{i,j'\}}$, then the offset that is added to the real output is given by $c_{i,j}\cdot(x'_i - x_i)\cdot x_{j'}$. However, this offset depends on the private input of $P_{j'}$ and this is disallowed as per the PwKO definition. Specifically, the adversary is only allowed to set the output received by the honest parties based on the output of the function and it cannot depend on the private inputs of the honest parties in any other way. Therefore, the above protocol as such does not satisfy PwKO. Hence, we need a mechanism to force the adversary to give consistent inputs to every 2-ary function.

Key Idea. Suppose we construct a protocol that satisfies the following two properties.

1. If every corrupted party P_i uses consistent inputs with every honest party, i.e., P_i sends the same input x_i to every $\mathcal{O}_{\{i,j\}}$ where P_j is honest, then we want all the honest parties to compute the output of the polynomial subject to the adversary adding some offset.
2. If a corrupted party sends different x_i and $x_{i'}$ to $\mathcal{O}_{\{i,j\}}$ and $\mathcal{O}_{\{i,j'\}}$ where P_j and $P_{j'}$ are honest, then we want the adversary not to learn any information about the private inputs of the honest parties. Specifically, we want the outputs of all the 2-ary functions involving an honest party to be random.

We argue that the above two properties are sufficient to show PwKO.

- If adversary sends consistent inputs in each execution of a 2-ary function with an honest party, then the adversary can only send shares that do not add up to 0 or cheat in the executions of 2-ary functions that involve only corrupt parties. Both these attacks can be translated to the adversary adding some offset to the output of the polynomial evaluation and hence, our protocol satisfies PwKO.
- If the adversary sends inconsistent inputs to two different honest parties, we need to show that the adversary learns no information about the private inputs of the honest parties and the output obtained by the honest parties is independent of their private inputs. The first property follows directly as the outputs of each 2-ary function involving an honest party is random. To argue the second property, note that the outputs obtained by the honest parties depend only on the random values output by the 2-ary functions involving at least one honest party and the values provided by the 2-ary functions involving two corrupt parties. These are independent of the private inputs of the honest parties as required by PwKO.

However, on close observation, we realize that it is highly non-trivial to satisfy the second condition. In particular, even if the adversary cheats by sending inconsistent inputs to any pair of honest parties, then we want this cheating to be detected in all other 2-ary functions involving at least one honest party and the adversary should get no information about these outputs. This is especially challenging since the 2-ary functions do not have any shared randomness and all the functions are invoked in parallel.

Construction. To construct such a protocol, we tightly couple the outputs of each 2-ary function. This coupling ensures that even if the output of a single 2-ary function call gets erased (or equivalently, switched to random), this creates a "domino effect" and the outputs of all the 2-ary function calls involving at least one honest party get erased (i.e., switched to random). This allows us to argue the second condition. Let us explain how this domino effect is created.

We first design a special conditional disclosure of secrets (CDS) protocol that switches the output of $\mathcal{O}_{\{j,j'\}}$ to random if a corrupt party P_i sends inconsistent inputs to $\mathcal{O}_{\{i,j\}}$ and $\mathcal{O}_{\{i,j'\}}$ where P_j and $P_{j'}$ are honest. If this happens, observe that $s_j[j']$ and $s_{j'}[j]$ are erased from the adversary's view. This means that each $\{s_j[k]\}_{k\neq j'}$ and $\{s_{j'}[k]\}_{k\neq j}$ is randomly distributed and hence, the outputs of each 2-ary function $\mathcal{O}_{\{j,k\}}$ for every $k \neq j'$ and $\mathcal{O}_{\{j',k\}}$ for $k \neq j$ is randomly distributed. Thus, for every honest party P_h, $s_h[j]$ and $s_h[j']$ get erased from the adversary's view and we can continue the above argument to switch the output of each 2-ary function's output that involves at least one honest party to random.

CDS Protocol. A key technical contribution of this work is a construction of a CDS protocol that switches the output of $\mathcal{O}_{\{j,j'\}}$ if P_i sends inconsistent inputs to $\mathcal{O}_{\{i,j\}}$ and $\mathcal{O}_{\{i,j'\}}$.

We build this CDS protocol in two steps. We first build a protocol that has constant soundness error. That is, even if the corrupt P_i is sending inconsistent inputs to $\mathcal{O}_{\{i,j\}}$ and $\mathcal{O}_{\{i,j'\}}$, there is a constant probability that the output of $\mathcal{O}_{\{j,j'\}}$ is not switched to random. In the second step, we show how to bring down this soundness error to negligible.

CDS Functionality. For simplicity of notation, let us fix $j = 1$, $j' = 2$ and $i = 3$. Assume for simplicity that P_3's private input is a single bit and P_1 and P_2's private inputs used in the CDS protocol are two random bits y_1 and y_2 respectively. The CDS protocol ensures that if P_3's input to $\mathcal{O}_{\{1,3\}}$ and $\mathcal{O}_{\{2,3\}}$ are the same then, all the parties can recover $y_1 \oplus y_2$. Else, y_1, y_2 are hidden from the view of the adversary. We observe that this is sufficient to erase the output of $\mathcal{O}_{\{1,2\}}$ as we can add $y_1 \oplus y_2$ to the original output and send this modified output to each party. If the adversary sends consistent outputs, then every party can recover $y_1 \oplus y_2$ and use this to unmask the output of $\mathcal{O}_{\{1,2\}}$. To protect the privacy of honest parties, we additionally need that CDS protocol not to leak any information about x_3 if P_3 was honest.

Overview of Conditional Disclosure Protocol. We slightly abuse the notation and we describe our CDS protocol as making parallel calls to 2-ary functions $(\mathcal{O}_{\{1,2\}}, \mathcal{O}_{\{2,3\}}, \mathcal{O}_{\{1,3\}})$. In the final protocol, these CDS computations are baked into the 2-ary functions.

The following protocol will serve as the starting point for our construction. P_3 sends to both $\mathcal{O}_{\{1,3\}}$ and $\mathcal{O}_{\{2,3\}}$ the same degree-2 polynomial $q(x)$ over $\mathbb{F}_4$ sampled uniformly at random subject to $q(0) = x_3$. Inputs of P_1 and P_2 to $\mathcal{O}_{\{1,3\}}$ and $\mathcal{O}_{\{2,3\}}$ are, respectively, α_1 and α_2, which are uniform *non-zero* elements of $\mathbb{F}_4$. The output of $\mathcal{O}_{\{1,3\}}$ (resp. $\mathcal{O}_{\{2,3\}}$) is $(\alpha_1, q(\alpha_1))$ (resp. $(\alpha_2, q(\alpha_2))$). The function $\mathcal{O}_{\{1,2\}}$ is not used in this construction.

This is not a conditional disclosure protocol since it does not reveal $y_1 \oplus y_2$ in an honest execution. But, it hides (honest) P_3's input from colluding P_1, P_2. P_1 and P_2 learn the evaluation of $q(x)$ on at most two non-zero values. This perfectly hides x_3 from the collusion for the same reason a 2-private Shamir secret sharing is perfectly secure. Furthermore, if P_1 and P_2 are honest, with at least $1/9$ probability, all honest parties detect if a corrupt P_3 uses distinct values of x_3 for computing inputs to $\mathcal{O}_{\{1,3\}}$ and $\mathcal{O}_{\{2,3\}}$. Because, in this case, P_3's inputs to $\mathcal{O}_{\{1,3\}}$ and $\mathcal{O}_{\{2,3\}}$, say $q_1(x)$ and $q_2(x)$, respectively, are distinct degree-2 polynomials (if degree is more than 2, the functions output $\perp$). Hence, there exists a non-zero $\alpha \in \mathbb{F}_4$ on which both these polynomials evaluate to distinct values.[6] With probability $1/9$, (honest) P_1 and P_2 would choose $\alpha_1 = \alpha$ and $\alpha_2 = \alpha$, respectively, in which case, the function outputs reveal that $q_1(x) \neq q_2(x)$, resulting in all honest parties aborting the protocol.

Next, we tweak this protocol so that $y_1 \oplus y_2$ is revealed if the evaluation of the polynomial(s) on α_1 and α_2 are not in conflict, and is perfectly hidden if $\alpha_1 = \alpha_2$ and $q_1(\alpha_1) \neq q_2(\alpha_2)$. We achieve this as follows. We use $\mathcal{O}_{\{1,2\}}$ to output a list $\{y_1 \oplus y_2 \oplus \theta_p \oplus \phi_p\}_p$, where p ranges over all polynomials of degree at most 2 over $\mathbb{F}_4$, and θ_p and ϕ_p are one-time pads chosen by P_1 and P_2 to mask the secret. The behavior of $\mathcal{O}_{\{1,3\}}$ is modified such that, for each p, it gives up the value of θ_p only if $p(\alpha_1) = q_1(\alpha_1)$, where q_1 is the polynomial input by P_3 to $\mathcal{O}_{\{1,3\}}$. Similarly, for each p, $\mathcal{O}_{\{2,3\}}$ reveals ϕ_p for each p such that $p(\alpha_1) = q_2(\alpha_1)$, where q_2 is the polynomial input by P_3 to $\mathcal{O}_{\{2,3\}}$. If there exists a p such that $p(\alpha_1) = q_1(\alpha_1)$ and $p(\alpha_2) = q_2(\alpha_2)$, then by recovering θ_p and ϕ_p, the decoder can unmask $y_1 \oplus y_2$. Clearly such a p exists when P_3 is honest and, hence, $q_1 = q_2$. However, if a corrupt P_3 provides distinct polynomials q_1 and q_2, and P_1 and P_2 chose α_1 and α_2 that coincide with α such that $q_1(\alpha) \neq q_2(\alpha)$, then there is no p for which θ_p and ϕ_p are simultaneously revealed by $\mathcal{O}_{\{1,3\}}$ and $\mathcal{O}_{\{2,3\}}$, respectively. This ensures that, with a constant probability (specifically, $1/9$), $y_1 \oplus y_2$ remains completely hidden from corrupt P_3, when P_1 and P_2 are honest. Note that, privacy of an honest P_3 is still preserved: revealing the set of polynomials that agree with a randomly chosen polynomial q that evaluates to x_3 on 0 on any non-zero α_1 and α_2 of P_1 and P_2's choosing does not reveal x_3.

[6] Note that any two distinct degree-2 polynomials can have the same evaluation on at most two points on the finite field.

This protocol has a clear flaw that allows a rushing P_3 to choose $q_1 \neq q_2$ without getting caught by honest parties: P_3 can learn α_1 chosen by P_1 from the output of $\mathcal{O}_{\{1,3\}}$. And, by looking at the polynomials that agree with q_1 on α_1, it adaptively chooses $q_2(x)$ such that $q_1(0) \neq q_2(0)$. Then, the parties will never detect that $q_1 \neq q_2$ irrespective of the value of α_2. This attack (in fact, all attacks in general from P_3) can be circumvented by ensuring that α_1 and α_2 are hidden from the adversary until the outputs of all 2-ary functions are revealed. We achieve this by appropriately masking the variables in the output of the functions that are sensitive to the value of α_1 and α_2 and ensuring that these masks can be removed only when the outputs of all function calls are known.

Boosting Soundness. The above protocol ensures that if P_3 sends inconsistent input, then with 1/9 probability, $y_1 \oplus y_2$ remains hidden. To boost this probability to be very close to 1, we use the parallel repetition. In particular, we run the above protocol k times in parallel and wherein each repetition, P_3 chooses random polynomials q_1 and q_2 with the same constant term.[7]. We additionally set the secrets of P_1 and P_2 to be the XOR of the secrets used in each repetition. This construction guarantees that for an adversary to learn information about the secrets of P_1 and P_2, it should not get caught in each of the k repetitions. The probability of this event is upper bounded by $(\frac{8}{9})^k$ (which is negligible in k).

2.3 Relaxing to Computational Security

We now explain how to bootstrap the protocol for computing degree-2 polynomials with PwKO to computing arbitrary functions with stronger security guarantees. We do this by relaxing the security to be computational. Recall that the impossibility result only holds against computationally unbounded adversaries.

Let f be an arbitrary multiparty function that is computable by a poly-sized circuit. Consider an augmented functionality g that takes in (x_i, sk_i) from party P_i where sk_i is a signing key for a digital signature scheme. For each $i \in [n]$, g first computes $y = f(x_1, \ldots, x_n)$. It then signs y using sk_i for each $i \in [n]$ to obtain the signature σ_i. Finally, g outputs $(y, \sigma_1, \sigma_2, \ldots, \sigma_n)$.[8]

The parties use an effective degree-2 MPRE with *semi-malicious* security against arbitrary corruptions (see Definition 4.2) for computing the function g.[9] The parties use the previous protocol with PwKO to compute the pre-processing phase and the encoding function for the MPRE.[10] The parties also use the 2-ary functions to broadcast the verification key vk_i corresponding to sk_i. The parties

[7] We can modify the 2-ary function to output $\perp$ if the constant terms are not the same.

[8] A similar transformation was used in [IKSS22] to obtain security with unanimous abort from PwKO.

[9] Such an MPRE was constructed in [ABT18, GS18] assuming the existence of a semi-honest secure oblivious transfer.

[10] Note that we can extend the previous protocol to first apply a local function on the parties private inputs and then compute a degree-2 polynomial on the outputs of

then use the MPRE decoder to learn the output of g. The parties finally check if each σ_i is a valid signature on y under the verification key vk_i. If any of the checks do not pass, the party aborts.

Note that from the security of the PwKO protocol, the adversary only learns the output of degree-2 polynomial computation on the pre-processed values and this corresponds to the output of the encoding function of the MPRE. From the MPRE security, this encoding can be generated only given the output of g, and the inputs and randomness of the corrupted parties. If the adversary sends some other value as the output to the honest parties, then it corresponds to sending a tampered MPRE encoding. Note that all the honest parties will get the same output $(y', \sigma'_1, \ldots, \sigma'_n)$ as the decoding for MPRE does not require any secret state. If $y \neq y'$, it follows from the security of the digital signature scheme that each honest party will abort. If $y = y'$, but some signature check does not pass, then once again all the honest parties will abort as the verification check is publicly computable. The only case where all the honest parties will output y is when $y = y'$ and all the verification checks pass.

Extensions. We observe that the same construction can be instantiated with a degree-2 MPRE with *semi-malicious* security in the honest majority setting to obtain a protocol secure against malicious adversaries that corrupt a minority of the parties. [ABT18][Theorem 1.2] constructed such an MPRE assuming one-way functions. The complexity of this construction is polynomial in the number of parties and the size of the circuit representing the function. Using this MPRE construction, we obtain a protocol in our model that computes any function with unanimous abort against a computationally-bounded malicious adversary corrupting a strict minority of the parties. The security of this construction relies on one-way functions. We also show that if the parties have access to 3-ary functions (instead of 2-ary), then we can rely on the MPRE construction from Beaver et al. [BMR90] to give an analogous result in the dishonest majority. The security of this protocol again relies on the existence of one-way functions.

3 Open Problems

We highlight some interesting problems left open by our work.

– **Minimizing Assumption in Dishonest Majority.** Assuming the existence of one-way functions, 3-ary functions can be used to achieve computational security with unanimous abort against an arbitrary number of corruptions. Similarly, 2-ary functions can be used to achieve the same against a strict minority of corruptions. We leave open the possibility of achieving security with abort with parallel calls to 2-ary functions in the dishonest majority setting based on one-way functions. This work also leaves open the (im)possibility

these local functions. This allows us to compute the pre-processing phase inside the 2-ary functions and thus, allowing us to rely on an MPRE that is secure against semi-malicious adversaries.

of unconditional security with unanimous abort even with parallel calls to $(n-1)$-ary functions.

- **Using Private Decoders.** Our impossibility result relied on the decoder to be public, i.e., not have any secret state. One way to get around our impossibility in the statistical setting is to resort to using private decoders (which take the secret state of a party as an additional input) and settling for security with selective abort, where the adversary may select the set of honest parties who are denied the output of the computation. [ABT18][Theorem 1.1] showed the existence of a perfectly secure degree-2 MPRE (hence, perfectly secure semi-malicious degree-2 MPRE) in the honest majority setting for any function that is computed by a branching program. Hence, using our CDS construction (which is secure against dishonest majority), we can securely compute any branching program with statistical PwKO when there is an honest majority. We can then use a standard transformation from PwKO to selective abort described in [IKP10]. This transformation uses a Message Authentication Code (MACs) and the private decoding is needed so that the parties can check the validity of the tags.
  We leave open the possibility of achieving statistical security with selective abort for arbitrary corruptions using private decoding. An astute reader may have noticed that the construction sketched above gives rise to such a protocol if there exists a statistical degree-2 MPRE for arbitrary corruptions or a statistical degree-2 RE. Hence, the existence of such MPRE or RE–a long standing open problem–acts as a barrier to proving the impossibility of constructing such protocols with private decoding.
- **Achieving Identifiable Abort.** Our protocols do not satisfy identifiable abort security which guarantees that at least one of the corrupt parties can be detected by the honest parties whenever the protocol aborts. We leave open the problem of achieving identifiable abort in our model.
- **Guaranteed Output Delivery.** We also leave open the possibility of achieving guaranteed output delivery in our model. The existing results [FGMO01] imply this is impossible when there is a dishonest majority. However, the problem remains open when there is an honest majority.

4 Definitions

Notations. The set $\{1, 2, \ldots, m\}$ is denoted by $[m]$. A sequence $(x_{i_1}, \ldots, x_{i_m})$, where $(i_1, \ldots, i_m) = S$ will be denoted by $\{x_i\}_{i \in S}$; when S is clear from the context, simply write $\{x_i\}$. When $S \subseteq [m]$, we will sometimes abuse the notation to denote $f(x_1, \ldots, x_m)$ as $f(\{x_i\}_{i \in S}, \{x_i\}_{i \in [m] \setminus S})$.

We consider a secure computation model in which n parties securely compute a function by making parallel calls to 2-ary functions with no further interaction. Universally composable security (UC-security) against malicious adversaries, as conceived in [Can01], will be the focus of this work.

Let $\lambda \in \mathbb{N}$ be the security parameter. Let $n \in \mathbb{N}$ and let $P_1, \ldots, P_n$ be a set of distinct parties. Let f be an n-party functionality $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \to \mathcal{Y}$.

Definition 4.1. *An n-party protocol $\Pi = (\{\mathcal{O}_{\{i,j\}}\}, \mathsf{Enc}, \mathsf{Dec})$ for securely realizing a functionality f using parallel calls to 2-ary functions $\{\mathcal{O}_{\{i,j\}}\}$, where $\mathcal{O}_{\{i,j\}} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ for each $\{i,j\} \in \binom{[n]}{2}$, is described by an encoding function Enc and a decoding function Dec. Π proceeds as follows.*

- *Each party P_i has private input $x_i \in \mathcal{X}_i$. It samples private randomness r_i uniformly from $\{0,1\}^*$, and computes $y_{(i,j)} = \mathsf{Enc}(1^\lambda, (i,j), x_i, r_i)$ for each $j \neq i$.*
- *For each $\{i,j\} \in \binom{[n]}{2}$, P_i and P_j invoke the 2-ary function $\mathcal{O}_{\{i,j\}}$ with inputs $y_{(i,j)}$ and $y_{(j,i)}$, respectively, which delivers $z_{\{i,j\}} = \mathcal{O}_{\{i,j\}}(y_{(i,j)}, y_{(j,i)})$ to all parties.*
- *Finally, each P_k outputs $\mathsf{Dec}(1^\lambda, \{z_{\{i,j\}}\})$ and terminates. If output of P_k is $\perp$, we say that P_k has aborted the protocol.*

We say that Π securely computes f if, for any non-uniform polynomial time adversary $\mathcal{A}$ that corrupts an arbitrary subset $M \subset [n]$ of the parties, there exists an ideal world PPT simulator Sim such that, for every choice of inputs $\{x_i\}_{i\in[n]}$:

$$(\mathsf{View}_{\mathcal{A}}, \mathsf{out}) \approx_c \mathsf{Ideal}(1^\lambda, \mathsf{Sim}^{\mathcal{F}(\{x_i\}_{i\in[n]\setminus M}, \cdot)}, \{x_i\}_{i\in M}) \tag{1}$$

In the above, $\mathsf{View}_{\mathcal{A}}$ refers to the view of the adversary $\mathcal{A}$ (which includes the input, randomness, and the outputs of the function calls) and out refers to the output of all (honest) parties in the real execution of the protocol, which coincide as they use the same decoder. Ideal refers to the ideal execution of the protocol. This starts with the Sim run on $1^\lambda, \{x_i\}_{i\in M}$. $\mathcal{F}(\{x_i\}_{i\in[n]\setminus M}, \cdot)$ takes inputs $\{\widetilde{x}_i\}_{i\in M}$ from Sim and computes the output of f with the inputs of the honest parties being fixed to $\{x_i\}_{i\in[n]\setminus M}$ and the inputs of the corrupt parties being fixed to $\{\widetilde{x}_i\}_{i\in M}$. It delivers this output to Sim. If Sim sends an instruction to deliver the outputs to the honest parties, the functionality delivers the above computed output. Else, it asks the honest parties to abort. The output of Ideal corresponds to the output of Sim and the output of all the honest parties.

We can extend the above definition to consider computationally unbounded adversaries $\mathcal{A}$. In this case, we allow the simulator Sim and the distinguisher in Equation (1) to be unbounded. In this case, we say that the protocol satisfies statistical security.

Definition 4.1 can be generalized to n-party non-interactive protocols using k-ary functions ($k < n$) which take inputs from sets of k parties.

Definition 4.1 considers a strong security guarantee where all the honest parties either compute the output or all of them abort. This is known as *security with unanimous abort*. Some of the protocols we construct enjoy a weaker notion of security, which we define below:

Privacy with Knowledge of Output. A protocol is said to guarantee privacy with knowledge of outputs (PwKO) if the malicious adversary learns only the output of the function computation, and each honest party outputs a value that is chosen by the adversary for that party based (only) on the output of the function

computation. This corresponds to realizing the functionality $\mathcal{F}^{\text{PwKO}}$ which works exactly like $\mathcal{F}$ until it sends the output to Sim. However, after delivering the output to Sim, the functionality accepts an input $o \in \text{codomain}(f) \cup \{\perp\}$ from Sim, and delivers o as output to all honest parties.

4.1 Multiparty Randomized Encoding

Ishai and Kushilevitz introduced the notion of randomizing polynomials in [IK00] which was later generalized to *Randomized Encoding (RE)* by Applebaum, Ishai and Kushilevitz in [AIK04]. A function f is encoded by a randomized function $\hat{f}$ if the output of $\hat{f}$ allows reconstruction of the output of f and nothing more. The function f is trivially a randomized encoding of itself. The utility of this notion is rooted in the observation that it is possible to obtain randomized encodings which are *simpler* than the function itself. A well studied notion of simplicity is the degree of the randomized encoding when the output of $\hat{f}$ is viewed as a multi-variate polynomial of $x_1, \ldots, x_n, r$, where $x_1, \ldots, x_n$ are the inputs to f and r is the randomness used in the encoding.

Multiparty randomized encoding (MPRE) was introduced by Applebaum, Brakerski and Tsabary in [ABT18] as a generalization of randomized encoding. We are intereseted in MPRE that remains secure against an arbitrary number of corruption, which we define below:

Definition 4.2 (Degree-d MPRE [ABT18]). *Consider an n-party function $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \to \mathcal{Y}$. $\hat{f}$ is a degree d* multiparty randomized encoding *of f if $\hat{f}$ can be described as a polynomial of degree-d over $\mathsf{GF}[2]$, and the following conditions hold for a set of preprocessing functions $h_1, \ldots, h_n$.*

- ***Correctness.*** *There is a decoder Dec such that, for every input $x_1, \ldots, x_n$, and every choice of private randomness $r_1, \ldots, r_n$, we have:*

$$\mathsf{Dec}\left(\hat{f}\left(h_1(1^\lambda, x_1; r_1), \ldots, h_n(1^\lambda, x_n; r_n)\right)\right) = f(x_1, \ldots, x_n)$$

- ***Privacy.*** *For every subset $M \subset [n]$ and for any non-uniform polynomial time adversary $\mathcal{A}$ corrupting parties in M, we have a simulator Sim such that, when $\{r_i\}_{i \in [n] \setminus M}$ are uniformly sampled, for all $x_1, \ldots, x_n$ and $\{r_i\}_{i \in M}$,*

$$\hat{f}\left(h_1(1^\lambda, x_1; r_1), \ldots, h_n(1^\lambda, x_n; r_n)\right) \approx \mathsf{Sim}(f(x_1, \ldots, x_n), \{x_i, r_i\}_{i \in M}). \quad (2)$$

Here $\approx$ denotes computational indistinguishability.

In the general definition of MPRE, each party uses a private decoder which, in addition to the encoding, uses the party's input and private randomness to compute the output of the function. However, in the above definition, the decoder is public. Such an MPRE was constructed in [GS17, GS18, ABT18] by transforming a secure computation protocol that computes the function with semi-malicious security–security is guaranteed even when the adversary chooses arbitrary private randomness for corrupt parties. To arrange for public decoding

of this MPRE, it suffices to append an additional round of communication to the corresponding protocol, in which the output of the function computation is broadcasted. We note that such a multi-round semi-malicious secure protocol can be constructed from any multi-round semi-honest secure OT protocol via the GMW transformation [GMW87] or making black-box use of semi-honest OT [HIK+11].

4.2 One-Time Digital Signature

Definition 4.3 (One-time digital signature scheme). *Let λ be a security parameter. A triple of algorithms* $(\mathrm{Gen}, \mathrm{Sig}, \mathrm{Ver})$ *is a one-time digital signature scheme if the following properties are met:*

Authentication. *When r is chosen uniformly, for all $x \in \{0,1\}^m$,*

$$\Pr\left[\mathrm{Ver}(x, \mathrm{Sig}(x, \mathsf{sk}), \mathsf{vk}) = 1 | (sk, vk) \leftarrow \mathrm{Gen}(1^\lambda, r)\right] = 1$$

Unforgeability. *For any non-uniform polynomial time adversary $\mathcal{A}$, for any x,*

$$\Pr\left[(x \neq x') \wedge (\mathrm{Ver}(x', t', \mathsf{vk}) = 1) \middle| \begin{array}{l} (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathrm{Gen}(1^\lambda, r) \\ t = \mathrm{Sig}(x, \mathsf{sk}) \\ (x', t') \leftarrow \mathcal{A}(1^\lambda, x, \mathsf{vk}, t) \end{array}\right] = \mathrm{negl}(\lambda).$$

One-time digital signatures exist if one-way functions exist [Lam16].

5 Impossibility of Statistical Security with Abort

We begin by exploring the limitations of this model owing to non-interactivity. Indeed, secure computation with abort is impossible for certain simple 3-party functions in this model against a computationally unbounded adversary. We will also show that this impossibility can be further extended to n-party setting when the adversary may corrupt more than half of the participants.

Theorem 5.1. *There exists a 3-party function f that can be computed using a degree 2-polynomial such that no protocol making parallel uses to 2-ary functions achieves statistically secure computation of f with abort against a malicious adversary corrupting arbitrarily many parties.*

Proof. We will show that the theorem holds for the 3-party function f that takes $x_i \in \mathbb{F}_4$ from each P_i and outputs 0 if $x_1 + x_2 + x_3 = 0$ and outputs 1 otherwise. When an element x_i in $\mathbb{F}_4$ is represented as a 2-dimensional vector $(x_{i,0}, x_{i,1})$, f is computed by the following degree-2 polynomial over $\mathbb{F}_2$:

$$f((x_{1,0}, x_{1,1}), (x_{2,0}, x_{2,1}), (x_{3,0}, x_{3,1})) = (x_{1,0} \oplus x_{2,0} \oplus x_{3,0}) \vee (x_{1,2} \oplus x_{2,2} \oplus x_{3,1}).$$

Suppose $\Pi = (\{\mathcal{O}_{\{i,j\}}\}, \mathsf{Enc}, \mathsf{Dec})$ is a secure protocol computing f with abort. Let $\epsilon \geq 0$ be some constant such that the advantage of any computationally unbounded adversary in breaking the protocol is at most ϵ. That is, when $\approx_\epsilon$ denotes statistical distance of at most ϵ, for any adversary $\mathcal{A}$ corrupting $M \subset [3]$,

$$(\mathsf{View}_{\mathcal{A}}, \mathsf{out}) \approx_\epsilon \mathsf{Ideal}(1^\lambda, \mathsf{Sim}^{\mathcal{F}^{\mathrm{SA}}(\{x_i\}_{i\in[n]\setminus M}, \cdot)}, \{x_i\}_{i\in M}).$$

Consider an honest execution of Π with each x_i is distributed uniformly and independently over $\mathbb{F}_4$. Let $z_{\{1,2\}}, z_{\{2,3\}}$ and $z_{\{1,3\}}$ be the output of 2-ary functions $\mathcal{O}_{\{1,2\}}, \mathcal{O}_{\{2,3\}}$ and $\mathcal{O}_{\{3,1\}}$, respectively. That is, when r_1, r_2, r_3 are sampled uniformly, and $y_{(i,j)} = \mathsf{Enc}((i,j), x_i, r_i)$ for distinct $i, j \in [3]$,

$$\begin{aligned} z_{\{1,2\}} &= \mathcal{O}_{\{1,2\}}(y_{(1,2)}, y_{(2,1)}) \qquad z_{\{2,3\}} = \mathcal{O}_{\{2,3\}}(y_{(2,3)}, y_{(3,2)}) \\ z_{\{1,3\}} &= \mathcal{O}_{\{3,1\}}(y_{(1,3)}, y_{(3,1)}), \end{aligned} \tag{3}$$

Claim 5.1.1 *Over the randomness of* $x_1, x_2, x_3, r_1, r_2, r_3$,

$$\begin{aligned} \Pr \Big[&\exists (x_1', r_1', x_2', r_2') \text{ s.t. } y'_{(1,2)} = \mathsf{Enc}((1,2), x_1', r_1'), y'_{(2,1)} = \mathsf{Enc}((2,1), x_2', r_2'), \\ &\text{and } \mathsf{Dec}(\mathcal{O}_{\{1,2\}}(y'_{(1,2)}, y'_{(2,1)}), z_{\{1,3\}}, z_{\{2,3\}}) \notin \{\perp, f(x_1, x_2, x_3)\} \Big] \leq 9\epsilon, \end{aligned} \tag{4}$$

and

$$\begin{aligned} \Pr \Big[&\exists (x_1', r_1', x_3', r_3') \text{ s.t. } y'_{(1,3)} = \mathsf{Enc}((1,3), x_1', r_1'), y'_{(3,1)} = \mathsf{Enc}((3,1), x_3', r_3'), \\ &\text{and } \mathsf{Dec}(z_{\{1,2\}}, \mathcal{O}_{\{1,3\}}(y'_{(1,3)}, y'_{(3,1)}), z_{\{2,3\}}) \notin \{\perp, f(x_1, x_2, x_3)\} \Big] \leq 9\epsilon. \end{aligned} \tag{5}$$

Before proving the claim, we will use this claim to prove the theorem. Consider an adversary $\mathcal{A}_1$ that corrupts P_1 and behaves as described in Figure 5.1

$\mathcal{A}_1$

1. Corrupt P_1. Sample x_1, r_1 uniformly and independently. Invoke $\mathcal{O}_{\{1,2\}}$ and $\mathcal{O}_{\{1,3\}}$ with $y_{(1,2)} = \mathsf{Enc}((1,2), x_1, r_1)$ and $\hat{y}_{(1,3)} = \mathsf{Enc}((1,3), x_1 + 1, r_1)$, respectively, as the inputs of P_1, and receive outputs $z_{\{1,2\}}$ and $\hat{z}_{\{1,3\}}$, respectively. Let $z_{\{2,3\}} = \hat{z}_{\{2,3\}}$ be the output of invocation of $\mathcal{O}_{\{2,3\}}$ by the honest parties P_2 and P_3.
2. If there exist $\hat{z}_{\{1,2\}} \in \mathcal{O}_{\{1,2\}}(\cdot,\cdot)$ and $z_{\{1,3\}} \in \mathcal{O}_{\{1,3\}}(\cdot,\cdot)$ such that $\mathrm{Dec}(\{z_{\{i,j\}}\}) = a_0, \mathrm{Dec}(\{\hat{z}_{\{i,j\}}\}) = a_1$ and $a_0, a_1 \in \{0,1\}$, output (a_0, a_1); otherwise, output $\perp$.

Fig. 5.1.

We now argue that Π is not secure against $\mathcal{A}_1$. Let r_2 and r_3 be the private randomness used by P_2 and P_3, respectively. Let $y_{(i,j)} = \mathsf{Enc}((i,j), x_i, r_i)$ for

each $i \in [3], j \neq i$. Then, $z_{\{i,j\}} = \mathcal{O}_{\{i,j\}}(y_{(i,j)}, y_{(j,i)})$ for each $\{i,j\} \neq \{1,3\}$. Defining $\tilde{z}_{\{1,3\}} = \mathcal{O}_{\{1,2\}}(y_{(1,3)}, y_{(3,1)})$, $z_{\{1,2\}}, z_{\{2,3\}}$ and $\tilde{z}_{\{1,3\}}$ are the outputs of the function calls in an honest execution of Π with x_i as input and r_i as randomness of each P_i. Hence,

$$\Pr\left[\mathsf{Dec}(z_{\{1,2\}}, z_{\{2,3\}}, z_{\{1,3\}}) = f(x_1, x_2, x_3)\right] \geq 1 - \epsilon.$$

This along with eq. (5) implies $\Pr[a_1 = f(x_1, x_2, x_3)] \geq 1 - 10\epsilon$. Using the same line of argument, $\Pr[a_0 = f(1 + x_1, x_2, x_3)] \geq 1 - 10\epsilon$. Thus, in the real execution, $\mathcal{A}_1$ successfully computes $f(x_1, x_2, x_3)$ and $f(1+x_1, x_2, x_3)$ with probability at least $1 - 20\epsilon$.

$\Pr[f(x_1 + 1, x_2, x_3) = 1 | f(x_1, x_2, x_3) = 1] = 2/3$, and $f(x_1, x_2, x_3) = 1$ with probability $3/4$ for any x_1, when of x_2 and x_3 are chosen uniformly. Hence, the simulator will fail to simultaneously guess the values of $f(x_1, x_2, x_3)$ and $f(x_1 + 1, x_2, x_3)$ with probability at least $1/4$. Thus, for sufficiently small ϵ, this contradicts security with at most ϵ distinguishing advantage against an adversary that corrupts at most 2 parties. We conclude the proof by proving the claim.

Proof (of Claim 5.1.1). We will prove Equation (4); Equation (5) can be proved similarly. Consider an adversary $\mathcal{A}_{1,2}$ that corrupts P_1 and P_2, and behaves as described in Figure 5.2.

$\mathcal{A}_{1,2}$

1. Corrupt P_1, P_2. Sample $r_1 \leftarrow \{0,1\}^*$, and invoke $\mathcal{O}_{\{1,3\}}$ with $y_{(1,3)} = \mathsf{Enc}((1,3), x_1, r_1)$ as P_1's input. Sample $r_2 \leftarrow \{0,1\}^*$ and invoke $\mathcal{O}_{\{2,3\}}$ with $y_{(2,3)} = \mathsf{Enc}((2,3), x_2, r_2)$ as P_2's input. Let $z_{\{1,3\}}$ and $z_{\{2,3\}}$ be the outputs of $\mathcal{O}_{\{1,3\}}$ and $\mathcal{O}_{\{2,3\}}$, respectively.
2. Define $y_{(1,2)} = \mathsf{Enc}((1,2), x_1, r_1)$ and $y_{(2,1)} = \mathsf{Enc}((2,1), x_2, r_2)$. Without invoking $\mathcal{O}_{\{1,2\}}$, compute $z_{\{1,2\}} = \mathcal{O}_{\{1,2\}}(y_{(1,2)}, y_{(2,1)})$. If there exist $y'_{(1,2)} = \mathsf{Enc}((1,2), x'_1, r'_1)$ and $y'_{(2,1)} = \mathsf{Enc}((2,1), x'_2, r'_2)$, for some x'_1, r'_1, x'_2, r'_2, such that

$$\mathsf{Dec}(\mathcal{O}_{\{1,2\}}(y'_{(1,2)}, y'_{(2,1)}), z_{\{1,3\}}, z_{\{2,3\}}) \notin \{\perp, \mathsf{Dec}(z_{\{1,2\}}, z_{\{1,3\}}, z_{\{2,3\}})\},$$

   then call $\mathcal{O}_{\{1,2\}}$ with $y'_{(1,2)}$ and $y'_{(2,1)}$ as inputs of P_1 and P_2; else use $y_{(1,2)}$ and $y_{(2,1)}$ as inputs of P_1 and P_2.
3. Output $\mathsf{Dec}(z_{\{1,2\}}, z_{\{1,3\}}, z_{\{2,3\}})$ and terminate.

Fig. 5.2.

We define a couple of distinguishers D_1 and D_2 that take the view of $\mathcal{A}_{1,2}$ and inputs/outputs of P_3 and does the following:

1. D_1 extracts $z_{\{1,3\}}$ and $z_{\{2,3\}}$ from the view of $\mathcal{A}_{1,2}$. It then extracts (x_1, r_1, x_2, r_2) from $\mathcal{A}_{1,2}$'s view and computes $z_{\{1,2\}}$. If $\mathsf{Dec}(z_{\{1,2\}}, z_{\{1,3\}}, z_{\{2,3\}}) = f(x_1, x_2, x_3)$, D_1 outputs 1, and outputs 0 otherwise.

2. If the output of P_3 is not equal to $f(x_1, x_2, x_3)$ or $\perp$, D_2 outputs 1, and outputs 0 otherwise.

We prove that, when ϵ is sufficiently small, there exists no simulator that guarantees at most ϵ distinguishability advantage against both D_1 and D_2 if

$$\Pr\Big[\exists(x_1', r_1', x_2', r_2') \text{ s.t. } y'_{(1,2)} = \mathsf{Enc}((1,2), x_1', r_1'), y'_{(2,1)} = \mathsf{Enc}((2,1), x_2', r_2'), \\ \text{and } \mathsf{Dec}(\mathcal{O}_{\{1,2\}}(y'_{(1,2)}, y'_{(2,1)}), z_{\{1,3\}}, z_{\{2,3\}}) \notin \{\perp, f(x_1, x_2, x_3)\}\Big] > 9\epsilon. \tag{6}$$

Suppose such a simulator Sim exists. Let $\hat{x}_1$ and $\hat{x}_2$ be the inputs of P_1 and P_2 used by Sim while invoking the functionality. We use security against D_2 to argue that $\hat{x}_2 + \hat{x}_1 \neq x_2 + x_1$ with substantial probability. If $\hat{x}_2 + \hat{x}_1 = x_2 + x_1$, then $f(x_1, x_2, x_3) = f(\hat{x}_1, \hat{x}_2, x_3)$ for any x_3. Hence,

$$\begin{aligned}\Pr[D_2 \text{ outputs 1 in the ideal execution}] &= \Pr[f(x_1, x_2, x_3) \neq f(\hat{x}_1, \hat{x}_2, x_3)] \\ &\leq \Pr[\hat{x}_2 + \hat{x}_1 \neq x_2 + x_1].\end{aligned}$$

Whereas, by our assumption in eq. (6), and the property of $\mathcal{A}_{1,2}$, D_2 outputs 1 in the real execution with probability more than 9ϵ. Since the probability with which D_2 outputs 1 in real and ideal executions are at most ϵ apart, the above observations imply that

$$\begin{aligned}8\epsilon &< \Pr[D_2 \text{ outputs 1 in the real execution}] - \epsilon \\ &\leq \Pr[D_2 \text{ outputs 1 in the ideal execution}] \leq \Pr[\hat{x}_2 + \hat{x}_1 \neq x_2 + x_1].\end{aligned} \tag{7}$$

Next, we use security against D_1 to argue that $\hat{x}_2 + \hat{x}_1 \neq x_2 + x_1$ can only occur with very low probability, reaching a contradiction. If $\hat{x}_2 + \hat{x}_1 \neq x_2 + x_1$, over the randomness of x_3,

$$\Pr[f(x_1, x_2, x_3) = 0 \mid \hat{x}_2 + \hat{x}_1 \neq x_2 + x_1, f(\hat{x}_1, \hat{x}_2, x_3) = 1] = 1/3.$$

Note that $\mathrm{Dec}(\mathcal{O}_{\{1,2\}}(y_{(1,2)}, y_{(2,1)}), \hat{z}_{\{1,3\}}, \hat{z}_{\{2,3\}})$ is a value that the simulator computes using $x_1, x_2, \hat{x}_1, \hat{x}_2$ and $f(\hat{x}_1, \hat{x}_2, x_3)$ and its own private randomness. Since an 2/3-biased coin cannot be guessed with non-zero advantage, the above condition implies that, when E is the event $\hat{x}_2 + \hat{x}_1 \neq x_2 + x_1, f(\hat{x}_1, \hat{x}_2, x_3) = 1$,

$$\Pr\big[\mathrm{Dec}(\mathcal{O}_{\{1,2\}}(y_{(1,2)}, y_{(2,1)}), \hat{z}_{\{1,3\}}, \hat{z}_{\{2,3\}}) \neq f(x_1, x_2, x_3) \mid E\big] \geq 1/3.$$

The above equality implies

$$\begin{aligned}&\Pr[D_1 \text{ outputs 1 in the ideal execution}] \\ &\quad= \Pr\big[\mathrm{Dec}(\mathcal{O}_{\{1,2\}}(y_{(1,2)}, y_{(2,1)}), \hat{z}_{\{1,3\}}, \hat{z}_{\{2,3\}}) = f(x_1, x_2, x_3)\big] \\ &\quad\leq 1 - \frac{1}{3}\Pr[\hat{x}_2 + \hat{x}_1 \neq x_2 + x_1, f(\hat{x}_1, \hat{x}_2, x_3) = 1] \\ &\quad\leq 1 - \frac{1}{4}\Pr[\hat{x}_2 + \hat{x}_1 \neq x_2 + x_1].\end{aligned} \tag{8}$$

The last inequality used $f(\hat{x}_1, \hat{x}_2, x_3) = 1$ with probability 3/4 over the randomness of x_3.

Next, we bound the probability with which D_1 outputs 1 in the real execution. Since r_1 and r_2 are sampled uniformly from $\{0,1\}^*$ and P_3 is honest, random variables $(z_{\{1,2\}}, z_{\{2,3\}}, z_{\{1,3\}})$ are identically distributed as in an honest execution of Π (as described in Equation (3)). By correctness of the protocol when all parties are honest, the output of the decoder is $f(x_1, x_2, x_3)$ with probability at least $1 - \epsilon$. In other words,

$$\begin{aligned} &\Pr\left[D_1 \text{ outputs 1 in the real execution}\right] \\ =&\Pr\left[\mathsf{Dec}(z_{\{1,2\}}, z_{\{1,3\}}, z_{\{2,3\}}) = f(x_1, x_2, x_3)\right] \geq 1 - \epsilon. \end{aligned} \tag{9}$$

The probability with which D_1 outputs 1 in real and ideal executions are at most ϵ apart. Hence, by eq. (8) and eq. (9),

$$\begin{aligned} 1 - 2\epsilon &\leq \Pr\left[D_1 \text{ outputs 1 in the real execution}\right] - \epsilon \\ &\leq \Pr\left[D_1 \text{ outputs 1 in the ideal execution}\right] \leq 1 - \frac{1}{4}\Pr[\hat{x}_2 + \hat{x}_1 \neq x_2 + x_1]. \end{aligned}$$

Hence, $\Pr[\hat{x}_2 + \hat{x}_1 \neq x_2 + x_1] \leq 8\epsilon$. This contradicts eq. (7). We conclude that Sim does not exist; proof is complete. □

This completes the proof of the theorem. □

Using a similar analysis we can extend the above result to n-party functions against an adversary that corrupts a majority of the parties. The following theorem is proved in Theorem A.1.

Theorem 5.2. *There exists an n-party function f such that no protocol making parallel uses of 2-ary functions that securely computes f with statistical security with abort against a malicious adversary corrupting at most $\lceil n/2 \rceil$ parties.*

6 Positive Results

In this section, we give our two positive results, namely, a statistical PwKO protocol for computing degree-2 functions and computationally secure protocol for computing arbitrary functions. A key building block used in these constructions is conditional disclosure of secrets protocol that is described next.

6.1 Conditional Disclosure Protocol

A crucial building block in our later constructions is the so-called conditional disclosure protocol which ensures that the inputs used by any (corrupt) party P_i during their oracle calls to the 2-ary functions in the larger non-interactive protocol are consistent.

A conditional disclosure protocol in which *parties P_i and P_j verify the consistency of P_k's inputs to $\mathcal{O}_{\{i,k\}}$ and $\mathcal{O}_{\{j,k\}}$* is a n-party non-interactive protocol

with parallel use of 2-ary functions that effectively delivers a secret that is additively secret shared between P_i and P_j to all parties if P_k's input to $\mathcal{O}_{\{i,k\}}$ and $\mathcal{O}_{\{j,k\}}$ are consistent. If P_k uses inconsistent inputs, then all parties detect malpractice and abort; furthermore, the secret inputs of P_i and P_k are kept hidden from the adversary. On the other hand, when P_i and P_j are corrupt, the protocol ensures perfect privacy of P_k's input. The n-party conditional disclosure protocol in which P_i and P_j verify P_k is denoted by $\mathrm{CD}_{\{i,j\},k}$.

Definition 6.1. *Let P_i, P_j, P_k be distinct parties. Let $\mathrm{CD}_{\{i,j\},k}$ be an n-party protocol using parallel use of 2-ary functions with $x_i, x_j \in \{0,1\}^m$, respectively, as inputs of P_i, P_j and $x_k \in \{0,1\}^\ell$ as the input of P_k, and no inputs from the remaining parties, described by encoder Enc, 2-ary functions $\{\mathcal{O}_{\{i,j\}}, \mathcal{O}_{\{j,k\}}, \mathcal{O}_{\{i,k\}}\}$ and a decoder Dec. $\mathrm{CD}_{\{i,j\},k}$ is said to be a conditional disclosure protocol with P_i and P_j verifying P_k with $\epsilon(\lambda)$ soundness if the following properties are met.*

1. *If P_i, P_j, P_k are honest, all honest parties output $x_i \oplus x_j$ at the end of the protocol.*
2. *If P_k is honest, a malicious adversary corrupting P_i and P_j does not learn x_k. That is, the view of the adversary corrupting P_i, P_j is identically distributed irrespective of the value of x_k.*
3. *Π offers $\epsilon(\lambda)$ soundness if for any computationally unbounded adversary $\mathcal{A}$ that corrupts a set of parties $\{P_a\}_{a\in C}$ such that $\{i,j\} \cap C = \emptyset$, there exists an ideal world PPT simulator Sim such that the following conditions are met:*

(a) *For every choice of inputs $\{x_a\}_{a\in\{i,j,k\}}$*

$$(\mathsf{View}_{\mathcal{A}}, \mathsf{out}) \equiv \mathsf{Ideal}(1^\lambda, \mathsf{Sim}^{\mathcal{F}_{\mathrm{CD}_{\{i,j\},k}}(x_i,x_j,\cdot)}, x_k). \tag{10}$$

Here, $\mathsf{View}_{\mathcal{A}}$ refers to the view of the adversary $\mathcal{A}$ (which includes the input, randomness, and the outputs of the function calls) and out refers to the output of all honest parties which coincide. Ideal refers to the ideal execution of the protocol. This starts with the Sim run on $1^\lambda, x_k$. If $\mathcal{F}(x_i, x_j, \cdot)$ receives a $b = 1$ from Sim, it delivers $x_i \oplus x_j$ to Sim, and all the honest parties; if $b = 0$, it delivers $\perp$ to Sim and all honest parties. The output of Ideal corresponds to the output of Sim and the output of all the honest parties (which coincide).

(b) *Conditioned on the event that the input of Sim to $\mathcal{O}_{\{i,k\}}$ and $\mathcal{O}_{\{j,k\}}$ are, respectively, $y_{\{i,k\}}$ and $y_{\{j,k\}}$ such that $y_{\{i,k\}}$ belongs to the domain of $\mathsf{Enc}((i,k), x'_k, \cdot)$ and $y_{\{j,k\}}$ belongs to the domain of $\mathsf{Enc}((j,k), x''_k, \cdot)$, and $x'_k \neq x''_k$, the input of Sim to $\mathcal{F}(x_i, x_j, \cdot)$ is 0 with probability $1 - \epsilon(\lambda)$.*

Remark 6.2. Consider an execution of $\mathrm{CD}_{\{i,j\},k}$ in which an adversary corrupts P_k, while both P_i and P_j are honest. In our definition, conditioned on Sim sending $b = 0$ to $\mathcal{F}_{\mathrm{CD}}$, Sim perfectly simulated the view of the adversary without knowing the secret, and the output of all honest parties is $\perp$. This models the soundness requirement that the secret that is shared between P_i and P_j remains hidden from the adversary, and that all honest parties output $\perp$. This justifyies defining soundness as the probability with which Sim sends $b = 1$ conditioned on the event that the adversary sends inconsistent values of x_k to $\mathcal{O}_{\{j,k\}}$ and $\mathcal{O}_{\{i,k\}}$ on behalf of P_k.

Figure 6.1 provides a formal description of the protocol which clarifies how the aforementioned masking is carried out. The protocol also hides the secret in the degree 2 term instead of the constant term allowing the construction to be based on $\mathbb{F}_3$ rather than $\mathbb{F}_4$. Lemma 6.3 formally proves that the construction is a 1/9 sound conditional disclosure protocol where P_1 and P_2 verify P_3's inputs.

$\mathrm{CD}_{\{12\},3}$

Let the inputs of P_1, P_2 and P_3 be $x_1, x_2, x_3 \in \{0,1\}$, respectively. Let $\mathbb{F}_3^2[u]$ be the set of all univariate polynomials over the variable u with coefficients in $\mathbb{F}_3$ and of degree at most 2 (there are 27 of them).

Execution of Enc

1. For $i \in \{2,3\}, \mathsf{Enc}((1,i), x_1, r_1) = (x_1, r_1)$.
2. For $i \in \{1,3\}, \mathsf{Enc}((2,i), x_2, r_2) = (x_2, r_2)$.
3. For $i \in \{1,2\}, \mathsf{Enc}((3,i), x_3, r_3) = (x_3, r_3)$.

Execution of $\mathcal{O}_{\{i,j\}}$

Interpret r_1 as $(\alpha_1, \{(\theta_{p,0}, \theta_{p,1}, a_p, a'_p)\}_p)$, where $\alpha_1 \leftarrow \mathbb{F}_3$, and for each $p \in \mathbb{F}_3^2[u]$, $\theta_{p,0}, \theta_{p,1}$, a_p and a'_p are uniform independent bits. Interpret r_2 as $(\alpha_2, \{(\phi_{p,0}, \phi_{p,1}, b_p, b'_p)\}_p)$, where $\alpha_2 \leftarrow \mathbb{F}_3$, and for each $p \in \mathbb{F}_3^2[u]$, $\phi_{p,0}, \phi_{p,1}$, b_p and b'_p are uniform independent bits. Interpret r_3 as (c_0, c_1), where $c_0, c_1 \leftarrow \mathbb{F}_3$, and let $q(u) = x_3 u^2 + c_1 u + c_0$; here we typecast x_3 in $\mathcal{F}_3$.

1. $\mathcal{O}_{\{1,2\}}((x_1, r_1), (x_2, r_2))$: Output $\{x_1 \oplus x_2 \oplus \theta_{p,1} \oplus \phi_{p,1}, a_p \oplus b'_p, b_p \oplus a'_p\}_p$.
2. $\mathcal{O}_{\{1,3\}}((x_1, r_1), (x_3, r_3))$: For each $p \in \mathbb{F}_3^2[u]$, if $p(\alpha_1) = q(\alpha_1)$, set $\gamma_p = 1$; else set $\gamma_p = 0$. Output $\{(\gamma_p \oplus a_p, \theta_{p,\gamma_p}, a'_p)\}_p$.
3. $\mathcal{O}_{\{2,3\}}((x_2, r_2), (x_3, r_3))$: For each $p \in \mathbb{F}_3^2[u]$, if $p(\alpha_2) = q(\alpha_2)$, set $\gamma'_p = 1$; else set $\gamma'_p = 0$. Output $\{(\gamma'_p \oplus b_p, \phi_{p,\gamma'_p}, b'_p)\}_p$.

Execution of Dec

1. For each $p \in \mathbb{F}_3^2[u]$, recover γ_p from $\gamma_p \oplus a_p, b'_p$ and $a_p \oplus b'_p$, which are available in the output of $\mathcal{O}_{\{1,3\}}, \mathcal{O}_{\{2,3\}}$ and $\mathcal{O}_{\{1,2\}}$, respectively.
2. Similarly, for each $p \in \mathbb{F}_3^2[u]$, recover γ'_p from $\gamma'_p \oplus b_p, a'_p$ and $b_p \oplus a'_p$.
3. If there exists p such that $\gamma_p = \gamma'_p = 1$, the recover $x_1 \oplus x_2$ from $x_1 \oplus x_2 \oplus \theta_{p,1} \oplus \phi_{p,1}, \theta_{p,1}$ and $\phi_{p,1}$ which are available in the outputs of $\mathcal{O}_{\{1,2\}}$, $\mathcal{O}_{\{1,3\}}$ and $\mathcal{O}_{\{2,3\}}$, respectively. If no such p exists, output $\perp$.

Fig. 6.1. Conditional disclosure protocol with 8/9 soundness in which P_1 and P_2 sharing a 1-bit secret verify P_3's 1-bit input to $\mathcal{O}_{\{1,3\}}$ and $\mathcal{O}_{\{2,3\}}$.

Lemma 6.3. *The protocol* $\mathrm{CD}_{\{12\},3}$ *in Figure 6.1 is a conditional disclosure protocol where* P_1 *and* P_2 *with single bit secret verify the single-bit input of* P_3 *with* $8/9$ *soundness.*

Proof. When P_3 is honest, for any α_1 and α_2 chosen by P_1 and P_2, there exists $p \in \mathbb{F}_3^2$ (specifically $p(u) = x_3u^2 + c_1u + c_0$) such that $p(\alpha_i) = x_3(\alpha_i)^2 + c_1\alpha_i + c_0$ for $i = 1, 2$. Hence, γ_p computed by $\mathcal{O}_{\{1,3\}}$ and γ'_p computed by $\mathcal{O}_{\{2,3\}}$ are both 1. Hence, the protocol satisfies the first condition in the definition of conditional disclosure protocol in Definition 6.1.

Next, we prove that it satisfies the second condition in Definition 6.1. The view of the adversary consists of x_1, x_2, the set of masks, α_1 and α_2 chosen by P_1 and P_2, and the class of polynomials $\mathcal{P}_1$ and $\mathcal{P}_1$, where, for each $i \in \{1, 2\}$, $\mathcal{P}_i$ contains the set of all degree-2 polynomials p such that $p(\alpha_i) = x_3(\alpha_i)^2 + c_1\alpha_i + c_0$. It is easy to see that this view only reveals $x_3\alpha_i^2 + c_1\alpha_i + c_0$ for $i = 1, 2$. Since c_0 and c_1 are uniformly and independently sampled by P_3, irrespective of the values of α_1 and α_2, these values are identically distributed for $x_3 = 0, 1$. In other words, the view of the adversary can be perfectly simulated irrespective of the value of x_3, satisfying the second condition.

Next, we show that the protocol is $8/9$-sound by showing that the simulator Sim in Figure 6.2 satisfies the conditions in Definition 6.1.

Sim

1. On behalf of P_1, choose uniformly random r_1 and interpret it as $(\alpha_1, \{(\theta_{p,0}, \theta_{p,1}, a_p, a'_p)\}_p)$. On behalf of P_2, choose uniformly random r_2 and interpret it as $(\alpha_2, \{(\phi_{p,0}, \phi_{p,1}, b_p, b'_p)\}_p)$.
2. Send $\mathcal{O}_{\{1,2\}}((0, r_1), (0, r_2)) = \{(\theta_{p,1} \oplus \phi_{p,1}, a_p \oplus b'_p, b_p \oplus a'_p)\}_p$, when $\mathcal{A}$ demands the output of $\mathcal{O}_{\{1,2\}}$,
3. When $\mathcal{A}$ queries $\mathcal{O}_{\{1,3\}}$ with input $(x_{3,\{1,3\}}, r_{3,\{1,3\}})$, check if $\mathcal{O}_{\{2,3\}}$ is already queried. If false, send $\mathcal{O}_{\{13\}}((0, r_1), (x_{3,\{1,3\}}, r_{3,\{1,3\}}))$. If true, let $(x_{3,\{2,3\}}, r_{3,\{2,3\}})$ be P_3's input to $\mathcal{O}_{\{2,3\}}$ and $\mathcal{O}_{\{1,3\}}$, respectively. Interpret $r_{3,\{1,3\}}$ as $(c_{0,\{1,3\}}, c_{1,\{1,3\}})$ and $r_{3,\{2,3\}}$ as $(c_{0,\{2,3\}}, c_{1,\{2,3\}})$. Check if $x_{2,\{1,3\}} \neq x_{3,\{1,3\}}$, $\alpha_1 = \alpha_2$ and

$$x_{3,\{1,3\}}\alpha_1^3 + c_{1,\{1,3\}}\alpha_1 + c_{0,\{1,3\}} \neq x_{3,\{2,3\}}\alpha_2^3 + c_{1,\{2,3\}}\alpha_2 + c_{0,\{2,3\}}. \quad (11)$$

   (a) If true, send $\mathcal{O}_{\{13\}}((0, r_1), (x_{3,\{1,3\}}, r_{3,\{1,3\}}))$, and send $b = 0$ as input to $\mathcal{F}_{\mathrm{CD}_{\{1,2\},3}}(x_1, x_2, \cdot)$.
   (b) If false, invoke $\mathcal{F}_{\mathrm{CD}_{\{1,2\},3}}(x_1, x_2, \cdot)$ with $b = 1$ to receive $x_1 \oplus x_2$. Send $\mathcal{O}_{\{13\}}((0, r'_1), (x_{3,\{1,3\}}, r_{3,\{1,3\}}))$ where r'_1 is obtained from r_1 by replacing $\theta_{p,1}$ with $\theta_{p,1} \oplus x_1 \oplus x_2$ for each p.
4. Behave analogously when $\mathcal{A}$ queries $\mathcal{O}_{\{2,3\}}$.

Fig. 6.2. Conditional disclosure protocol with $8/9$ soundness in which P_1 and P_2 sharing a 1-bit secret verify P_3's 1-bit input to $\mathcal{O}_{\{1,3\}}$ and $\mathcal{O}_{\{2,3\}}$.

We show that w.r.t. any adversary $\mathcal{A}$, the real world and the ideal world executions are computationally indistinguishable via a hybrid argument.

- Hyb_0 : This corresponds to the real world execution of the protocol.
- Hyb_1 : Let $(x_{3,\{1,3\}}, r_{3,\{1,3\}})$ and $(x_{3,\{2,3\}}, r_{3,\{2,3\}})$ be P_3's input to $\mathcal{O}_{\{2,3\}}$ and $\mathcal{O}_{\{1,3\}}$, respectively. Interpret $r_{3,\{1,3\}}$ as $(c_{0,\{1,3\}}, c_{1,\{1,3\}})$ and $r_{3,\{2,3\}}$ as $(c_{0,\{2,3\}}, c_{1,\{2,3\}})$. In this hybrid, check if $x_{2,\{1,3\}} \neq x_{3,\{1,3\}}$, $\alpha_1 = \alpha_2$ and eq. (11) are together satisfied. If not, change the output of $\mathcal{O}_{\{1,2\}}$ to $\{(\theta_{p,1} + \phi_{p,1}, a_p \oplus b'_p, b_p \oplus a'_p)\}$. If $\mathcal{O}_{\{1,3\}}$ is invoked after $\mathcal{O}_{\{2,3\}}$, replace $\theta_{p,1}$ with $x_1 \oplus x_2 \oplus \theta_{p,1}$ before computing the output of $\mathcal{O}_{\{1,3\}}$. Likewise, if $\mathcal{O}_{\{2,3\}}$ is invoked after $\mathcal{O}_{\{1,3\}}$, replace $\phi_{p,1}$ with $x_1 \oplus x_2 \oplus \theta_{p,1}$ before computing the output of $\mathcal{O}_{\{2,3\}}$. If the check succeeds, make no changes.
- Hyb_2 : In this hybrid, make the following change if the same check succeeds: change the output of $\mathcal{O}_{\{1,2\}}$ to $(\theta_{p,1} \oplus \phi_{p,1}, a_p \oplus b'_p, b_p \oplus a'_p)$.

Hyb_0 is perfectly indistinguishable from Hyb_1 because

$$\begin{aligned}
&\{x_1 \oplus x_2 \oplus (\theta_{p,1} + \phi_{p,1}, a_p \oplus b'_p, b_p \oplus a'_p), \theta_{p,1}, \phi_{p,1}, a_p, b_p, a'_p, b'_p\}_p \\
\equiv\ &\{(\theta_{p,1} + \phi_{p,1}, a_p \oplus b'_p, b_p \oplus a'_p), x_1 \oplus x_2 \oplus \theta_{p,1}, \phi_{p,1}, a_p, b_p, a'_p, b'_p\}_p \\
\equiv\ &\{(\theta_{p,1} + \phi_{p,1}, a_p \oplus b'_p, b_p \oplus a'_p), \theta_{p,1}, x_1 \oplus x_2 \oplus \phi_{p,1}, a_p, b_p, a'_p, b'_p\}_p.
\end{aligned}$$

When the check succeeds, for each p, $\gamma_p = 1 - \gamma'_p$. Since $\theta_{p,1}, \phi_{p,1}$ are uniformly and independently chosen,

$$\begin{aligned}
&\{x_1 \oplus x_2 \oplus (\theta_{p,1} + \phi_{p,1}, a_p \oplus b'_p, b_p \oplus a'_p), a_p, b_p, a'_p, b'_p, \theta_{p,\gamma_p}, \phi_{p,\gamma'_p}\}_p \\
&\qquad \equiv \{(\theta_{p,1} + \phi_{p,1}, a_p \oplus b'_p, b_p \oplus a'_p), a_p, b_p, a'_p, b'_p, \theta_{p,\gamma_p}, \phi_{p,\gamma'_p}\}_p.
\end{aligned}$$

Hence $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2$. It can be verified that Hyb_2 corresponds to the ideal execution using Sim. To show that the soundness error of the scheme is at most $8/9$, we need to show that Sim sends $b = 0$ to the functionality with probability at least $1/9$ when $x_{2,\{1,3\}} \neq x_{3,\{1,3\}}$. This event occurs if and only if $\alpha_1 = \alpha_2$ and eq. (11) are simultaneously satisfied, when $x_{2,\{1,3\}} \neq x_{3,\{1,3\}}$.

Owing to the symmetry, we assume, without loss of generality, that adversary rushes to obtain the output of $\mathcal{O}_{\{1,2\}}$; then adaptively chooses the input $(x_{3,\{1,3\}}, c_{0,\{1,3\}}, c_{1,\{1,3\}})$ to $\mathcal{O}_{\{1,3\}}$, obtains the output; then adaptively chooses the input $(1 \oplus x_{3,\{1,3\}}, c_{0,\{2,3\}}, c_{1,\{2,3\}})$ to $\mathcal{O}_{\{2,3\}}$, obtains its output. For any choice of $c_{0,\{1,3\}}, c_{1,\{1,3\}}$, the adversary only learns $(x_1 \oplus x_2 \oplus \theta_{p,1} \oplus \phi_{p,1}, a_p \oplus b'_p, b_p \oplus a'_p)$ for each p as the output of $\mathcal{O}_{\{1,2\}}$, and $(\gamma_p \oplus a_p, \theta_{p,\gamma_p}, a'_p)$ for each p from the output of $\mathcal{O}_{\{1,3\}}$. For any $p \in \mathbb{F}_3[u]$, a_p and b'_p are uniform independent bits. Hence, the value of γ_p is completely hidden from the adversary who knows $\gamma_p \oplus a_p$ and $a_p \oplus b'_p$. Thus, the above set of values are identically distributed irrespective of the values of α_1 (and, trivially, irrespective of α_2 since no function of α_2 has been revealed by $\mathcal{O}_{\{1,2\}}$ and $\mathcal{O}_{\{1,3\}}$). We conclude that, α_1 and α_2 are uniformly and independently distributed in $\mathbb{F}_3$, conditioned on adversary's view after the evaluation of $\mathcal{O}_{\{1,2\}}$ and $\mathcal{O}_{\{2,3\}}$. In other words, the distribution of $c_{0,\{1,3\}}, c_{1,\{1,3\}}, c_{0,\{2,3\}}$ and $c_{1,\{2,3\}}$ chosen by the adversary is independent of the

distribution of α_1, α_2. Thus, when $x_{2,\{1,3\}} \neq x_{3,\{1,3\}}$, the event where $\alpha_1 = \alpha_2$ and eq. (11) are simultaneously satisfied occurs with probability at least $1/9$. This concludes the proof of the lemma. □

Boosting soundness of Conditional Disclosure The protocol presented in the previous section only guarantees that a corrupt P_3 which provides inconsistent inputs to $\mathcal{O}_{\{1,3\}}$ and $\mathcal{O}_{\{2,3\}}$ would be detected with a constant probability, in which event the decoder reports an abort and the secret is hidden from the adversary. The following simple construction boosts the probability of this event such that it occurs with overwhelming probability.

To achieve soundness of $\mathrm{negl}(\lambda)$, the construction repeats the simple protocol λ times. If every execution succeeds, we require the parties to correctly recover the secrets. However, if the decoder reports an abort in any of the execution, which occurs independently with probability at least $1/9$ in each execution if the bit being verified is inconsistent, we require the secret to be completely hidden from the adversary, and the parties to report an abort. This is easy to arrange by having each repetition reveal an additive secret share of $x_1 \oplus x_2$; for this, P_1 and P_2 provide an additive secret share of their respective inputs as inputs to each execution. Crucially, all λ repetitions of the protocol takes the same input x_3 from P_3; Formally, the input of P_3 to both $\mathcal{O}_{\{1,3\}}$ and $\mathcal{O}_{\{2,3\}}$ is $x_3, \{c_{0,i}, c_{1,i}\}_{i\in[\lambda]}$, where $c_{b,j}$ is uniformly and independently chosen from $\mathbb{F}_3$ for each $i \in [\lambda]$ and $b \in \{0,1\}$. The polynomial used in repetition i is the polynomial $q(u) = x_3 \cdot u^2 + c_{1,\ell} \cdot u + c_{0,\ell}$. This ensures that each execution (independently) has at least $1/9$ probability of failing if (corrupt) P_3 uses inconsistent values of x_3 in $\mathcal{O}_{\{1,3\}}$ and $\mathcal{O}_{\{2,3\}}$.

The above protocol can be further modified to handle string inputs from all parties. Let $x_1, x_2 \in \{0,1\}^m$ be the inputs of P_1, P_2, and $x_3 \in \{0,1\}^\ell$ be the input of P_3. Modify the previous construction to verify each of the ℓ-bits in x_3 with $\mathrm{negl}(\lambda)$ soundness error, and reveal $x_1 \oplus x_2$ only if P_3 provides consistent values for every bit in x_3. As in the previous construction, this is arranged by having the verification of each bit reveal an additive secret share of $x_1 \oplus x_2$.

By modifying the $\mathrm{CD}_{\{12\},3}$ described in Figure 6.1 as discussed above, we obtain the following result as a consequence of Lemma 6.3.

Theorem 6.4. *For any m, ℓ, λ, there exists a conditional disclosure protocol $\mathrm{CD}_{\{12\},3}$ with P_1 and P_2 verifying P_3 with $\ell(8/9)^\lambda$ soundness when P_1 and P_2 have m bit inputs and P_3 has ℓ bit input.*

A formal description of $\mathrm{CD}_{\{12\},3}$ is provided in Figure A.3 in Appendix A.1. We prove Theorem 6.4 in Appendix A.1. We stress that, this protocol can be modified to obtain a conditional disclosure protocol in which inputs of parties belong to arbitrary sets: it suffices to *flatten* the input being verified and the secret into strings.

6.2 Computing Degree-2 Functions with PwKO

A deterministic n-party function f that takes input $x_i \in \{0,1\}^m$ from party $P_i, i \in [n]$ has effective degree 2 if it can be decomposed into functions $\{h_{\{i,j\}}\}$, where, for each $\{i,j\} \in \binom{[n]}{2}$, $h_{\{i,j\}} : \{0,1\}^m \times \{0,1\}^m \to \mathbb{G}$ for an additive finite group $\mathbb{G}$, such that, for all $x_1, \ldots, x_n$,

$$f(x_1, \ldots, x_n) = \sum_{\{i,j\}} h_{\{i,j\}}(x_i, x_j). \tag{12}$$

In this section, we construct a protocol that securely computes any function with effective degree 2. The construction follows the intuition sketched in Section 2.2.

Description of the Protocol. Let f be a deterministic n-party function of effective degree 2, and let $\{h_{\{i,j\}}\}$ be as described in eq. (12). Let $\lambda \in \mathbb{N}$ be a security parameter. The Figure 6.3 provides a formal description of a protocol that securely computes f with PwKO when the input of each party P_i is x_i.

The construction uses the following resources and notations: Let $\mathrm{CD}_{\{i,j\},k} = (\{\mathrm{CD}_{\{i,j\},k}.\mathcal{O}_{\{i,j\}}\}, \mathrm{CD}_{\{i,j\},k}.\mathsf{Enc}, \mathrm{CD}_{\{i,j\},k}.\mathsf{Dec})$ be a conditional disclosure protocol with $\mathrm{negl}(\lambda)$ soundness, conditionally reveals the secret that is shared between P_i and P_j after verifying the consistency of P_k's input. For conciseness, we will drop 1^λ in the argument, and denote $\mathrm{CD}_{\{i,j\},k}.\mathsf{Enc}(1^\lambda, (i,\ell), \cdot, \cdot)$ by $\mathrm{CD}_{\{i,j\},k}.\mathsf{Enc}_{(i,\ell)}(\cdot,\cdot)$, for all $\ell \in \{j,k\}$, and so on.
For legibility, we suppress the private randomness used in the encoder of Π_f as well as the conditional disclosure protocols invoked as sub-protocols. But, we stress that, an honest party uses the same private randomness while invoking the encoder to obtain their input to different 2-ary functions.

Correctness of the protocol. When all parties behave honestly, the output of $\mathrm{CD}_{\{i,j\},k}$ is $s_{\{i,j\},k} = s_i[j,k] + s_j[i,k]$ for each $\{i,j,k\} \in \binom{[n]}{3}$. This follows from the correctness of conditional disclosure protocol. Furthermore, $\{\gamma_i[j]\}_{j \neq i}$ forms an additive secret sharing of 0. Hence, the decoder outputs the following ensuring correctness:

$$\hat{z}_{\{i,j\}} - \sum_{k \notin \{i,j\}} s_i[j,k] + s_j[i,k] = \sum_{\{i,j\}} h_{\{i,j\}}(x_i, x_j) + \sum_i \sum_{j \neq i} \gamma_i[j] = f(x_1, \ldots, x_n).$$

The security with PwKO of the protocol follows the outline presented in the technical overview. A formal proof is provided in Appendix A.2.

Theorem 6.5. *The protocol Π_f in Figure 6.3 computes any n-party function f of effective degree 2 with statistical security while guaranteeing privacy with knowledge output against a malicious adversary that corrupts any set of parties.*

6.3 Achieving General Secure Computation with Abort

We use the protocol developed in the previous section to realize general secure computation with abort against computationally bounded adversaries. For this,

Π_f

(i). Execution of Enc

For any $j \neq i$, $(1^\lambda, x_i, \gamma_i[j], \{s_i[j,k], s_i[k,j]\}_{k \notin \{i,j\}}, R_i) \leftarrow \mathsf{Enc}(1^\lambda, (i,j), x_i, \cdot)$.
Here, $\gamma_i[j] \in \{0,1\}^l$ is such that $\{\gamma_i[k]\}_{k \neq i}$ is an additive secret sharing of $0 \in \mathbb{G}$; for each k, $s_i[j,k]$ and $s_i[k,j]$ are uniform in $\mathbb{G}$; R_i is a uniformly random string used to derive private randomness for the encoders of various parallel invocations of conditional disclosure protocols; see (ii).2.(a-c).
Recall, an honest P_i uses the same randomness (and input x_i) while invoking $\mathsf{Enc}(1^\lambda, (i,j), \cdot, \cdot)$ for all $j \neq i$. This is crucial in ensuring the required correlation among P_i's inputs to $\{\mathcal{O}_{\{i,j\}}\}_{j \neq i}$.

(ii). Execution of $\mathcal{O}_{\{i,j\}}$

When P_i's input is $(\lambda_i, x_i, \gamma_i[j], \{s_i[j,k], s_i[k,j]\}_{k \notin \{i,j\}}, R_i)$, and P_j's input is $(\lambda_j, x_j, \gamma_j[i], \{s_j[i,k], s_j[k,i]\}_{k \notin \{i,j\}}, R_j)$, $\mathcal{O}_{\{i,j\}}$ behaves as follows:

1. Compute $z_{\{i,j\}} = h_{\{i,j\}}(x_i, x_j)$. Output $\hat{z}_{\{i,j\}}$ where $\hat{z}_{\{i,j\}} = z_{\{i,j\}} + \gamma_i[j] + \gamma_j[i] + \sum_{k \notin \{i,j\}} (s_i[j,k] + s_j[i,k])$.
2. If $\lambda_j \neq \lambda_i$, output $\perp$ and terminate; otherwise, let $\lambda = \lambda_i$; for each $k \notin \{i,j\}$:
   (a) Output $\mathrm{CD}_{\{i,j\},k}.\mathcal{O}_{\{i,j\}}(y_{(i,j)}, y_{(j,i)})$, where $y_{(i,j)} \leftarrow \mathrm{CD}_{\{i,j\},k}.\mathsf{Enc}_{(i,j)}(s_i[j,k], \cdot)$ and $y_{(j,i)} \leftarrow \mathrm{CD}_{\{i,j\},k}.\mathsf{Enc}_{(j,i)}(s_j[i,k], \cdot)$. Note, fresh randomness from R_i, say $r_{i,(\{i,j\},k)}$, is used to compute P_i's encodings $\mathrm{CD}_{\{i,j\},k}.\mathsf{Enc}_{(i,j)}$ and $\mathrm{CD}_{\{i,j\},k}.\mathsf{Enc}_{(i,k)}$ (in $\mathcal{O}_{\{i,k\}}$); same for P_j.
   (b) Output $\mathrm{CD}_{\{i,k\},j}.\mathcal{O}_{\{i,j\}}(y_{(i,j)}, y_{(j,i)})$, where $y_{(i,j)} \leftarrow \mathrm{CD}_{\{i,k\},j}.\mathsf{Enc}_{(i,j)}(s_i[k,j], \cdot)$ and $y_{(j,i)} \leftarrow \mathrm{CD}_{\{i,k\},j}.\mathsf{Enc}_{(j,i)}(x_j, \cdot)$.
   (c) Output $\mathrm{CD}_{\{j,k\},i}.\mathcal{O}_{\{i,j\}}(y_{(i,j)}, y_{(j,i)})$, where $y_{(i,j)} \leftarrow \mathrm{CD}_{\{j,k\},i}.\mathsf{Enc}_{(i,j)}(x_i, \cdot)$, and $y_{(j,i)} \leftarrow \mathrm{CD}_{\{j,k\},i}.\mathsf{Enc}_{(j,i)}(s_j[k,i], \cdot)$.

(iii). Execution of Dec.

For each $\{i,j,k\} \in \binom{[n]}{3}$, let $s_{\{i,j\},k}$ be the output of $\mathrm{CD}_{\{i,j\},k}$. If $\hat{z}_{\{i,j\}} = \perp$ for some $\{i,j\}$ or $s_{\{i,j\},k} = \perp$ for some $\{i,j,k\}$, output $\perp$ and terminate; otherwise set $\hat{w}_{\{i,j\}} = \hat{z}_{\{i,j\}} - \sum_{k \notin \{i,j\}} s_{\{i,j\},k}$. Output $\sum_{\{i,j\}} \hat{w}_{\{i,j\}}$.

Fig. 6.3. Π_f computes f of effective degree 2 with privacy with knowledge of output.

we will rely on a computational semi-malicious MPRE, as defined in Definition 4.2. In [ABT18, Theorem 7.3], Applebaum, Brakerski and Tsabary showed that every n-party functionality f can be encoded by a computational degree-2 MPRE $\hat{f}$ with complexity polynomial in n and size of the circuit, by making non-black box use of a (possibly multi-round) oblivious transfer.

Since degree-2 MPRE has effective degree-2, we can compute f by first computing MPRE using the protocol in Figure 6.3, and then decode the MPRE to compute the function. Owing to computational security of the MPRE, the

resulting protocol securely computes f with PwKO against a computationally bounded adversary corrupting any subset of parties.

We use a standard approach to bootstrap this protocol and achieve security with abort. Instead of computing f in the aforementioned manner, compute a signed version of f in which the output of f is appended with one-time signatures (See Definition 4.3) of this value with respect to a signing key provided by each party. All parties can verify the validity of the signatures using the respective verifying keys. For this, every party broadcasts their verifying key in parallel with the function computation. If at least one of the signatures is found to be inconsistent, the parties unanimously declare abort instead of outputting the candidate output. The resulting protocol securely computes f with abort against a computationally unbounded adversary.

When all parties behave honestly, authenticity of the signature scheme (Definition 4.3) guarantees that the verification succeeds and the parties output the function output, ensuring correctness. Whereas, an adversary that attempts to modify the function output will fail to provide consistent signatures with overwhelming probability by the unforgeability property (Definition 4.3) of the one-time signature scheme, resulting in all honest parties issuing abort. Here, we have crucially relied on the PwKO guarantee of the underlying protocol to ensure that the adversary does not learn the signing key of any of the honest parties.

We obtain the following result. The formal description of the protocol achieving security with abort, and the proof of its security and correctness is provided in Appendix A.3.

Theorem 6.6. *Assuming the existence of (possibly multi-round) oblivious transfer that is secure against semi-honest adversaries, for every n-party functionality f, there exists a non-interactive protocol using parallel calls to 2-ary functions that achieves security with unanimous abort against a computationally bounded malicious adversary that corrupts an arbitrary number of parties.*

6.4 Extensions

Honest Majority Setting. The following result is obtained by instantiating the same construction with semi-maliciously secure 2-MPRE against a dishonest minority, which exists if one-way functions exist [ABT18]:

Theorem 6.7. *Assuming the existence of one-way functions, for every n-party functionality f, there exists a non-interactive protocol using parallel calls to 2-ary functions that achieves security with unanimous abort against a computationally bounded malicious adversary that corrupts a strict minority of parties.*

3-ary functions. We consider the same problem in the setting where the parties have access to 3-ary functions instead of 2-ary. We obtain the following result.

Theorem 6.8. *Assuming the existence of one-way functions, for every n-party functionality f, there exists a non-interactive protocol using parallel calls to 3-ary*

functions that achieves security with unanimous abort against a computationally bounded malicious adversary that corrupts an arbitrary number of parties.

This theorem is obtained by first constructing a protocol that securely computes degree-3 polynomials with PwKO. Then, it uses the MPRE construction from Beaver, Micali, and Rogaway [BMR90] in the bootstrapping step. We present a construction and its analysis in Appendix B.

References

ABG+20. Benny Applebaum, Zvika Brakerski, Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Separating two-round secure computation from oblivious transfer. In Thomas Vidick, editor, *ITCS 2020*, volume 151, pages 71:1–71:18. LIPIcs, January 2020.

ABT18. Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Perfect secure computation in two rounds. In Beimel and Dziembowski [BD18], pages 152–174.

ABT19. Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Degree 2 is complete for the round-complexity of malicious MPC. In Ishai and Rijmen [IR19], pages 504–531.

ACGJ18. Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Round-optimal secure multiparty computation with honest majority. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 395–424. Springer, Heidelberg, August 2018.

ACGJ19. Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Two round information-theoretic MPC with malicious security. In Ishai and Rijmen [IR19], pages 532–561.

AG21. Benny Applebaum and Aarushi Goel. On actively-secure elementary MPC reductions. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part I*, volume 13042 of *LNCS*, pages 717–749. Springer, Heidelberg, November 2021.

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

AIK07. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography with constant input locality. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 92–110. Springer, Heidelberg, August 2007.

BD18. Amos Beimel and Stefan Dziembowski, editors. *TCC 2018, Part I*, volume 11239 of *LNCS*. Springer, Heidelberg, November 2018.

BL18. Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Nielsen and Rijmen [NR18], pages 500–532.

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.

BOSS20. Carsten Baum, Emmanuela Orsini, Peter Scholl, and Eduardo Soria-Vazquez. Efficient constant-round MPC with identifiable abort and public verifiability. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 562–592. Springer, Heidelberg, August 2020.

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

CGH+85. Benny Chor, Oded Goldreich, Johan Håstad, Joel Friedman, Steven Rudich, and Roman Smolensky. The bit extraction problem of t-resilient functions (preliminary version). In *26th FOCS*, pages 396–407. IEEE Computer Society Press, October 1985.

Cle86. Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *18th ACM STOC*, pages 364–369. ACM Press, May 1986.

DI05. Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 378–394. Springer, Heidelberg, August 2005.

FGMO01. Matthias Fitzi, Juan A. Garay, Ueli M. Maurer, and Rafail Ostrovsky. Minimal complete primitives for secure multi-party computation. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 80–100. Springer, Heidelberg, August 2001.

FOC86. *27th FOCS*. IEEE Computer Society Press, October 1986.

GIS18. Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Two-round MPC: Information-theoretic and black-box. In Beimel and Dziembowski [BD18], pages 123–151.

GMW86. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In FOCS 1986 [FOC86], pages 174–187.

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.

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

GS18. Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Nielsen and Rijmen [NR18], pages 468–499.

HIK+11. Iftach Haitner, Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions of protocols for secure computation. *SIAM J. Comput.*, 40(2):225–266, 2011.

HILL99. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

HKW20. Susan Hohenberger, Venkata Koppula, and Brent Waters. Chosen ciphertext security from injective trapdoor functions. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 836–866. Springer, Heidelberg, August 2020.

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

IKP10. Yuval Ishai, Eyal Kushilevitz, and Anat Paskin. Secure multiparty computation with minimal interaction. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 577–594. Springer, Heidelberg, August 2010.

IKSS22. Yuval Ishai, Dakshita Khurana, Amit Sahai, and Akshayaram Srinivasan. Round-optimal black-box secure computation from two-round malicious OT. In Eike Kiltz and Vinod Vaikuntanathan, editors, *Theory of Cryptography - 20th International Conference, TCC 2022, Chicago, IL, USA, November 7-10, 2022, Proceedings, Part II*, volume 13748 of *Lecture Notes in Computer Science*, pages 441–469. Springer, 2022.

IPP+22. Yuval Ishai, Arpita Patra, Sikhar Patranabis, Divya Ravi, and Akshayaram Srinivasan. Fully-secure MPC with minimal trust. In Eike Kiltz and Vinod Vaikuntanathan, editors, *Theory of Cryptography - 20th International Conference, TCC 2022, Chicago, IL, USA, November 7-10, 2022, Proceedings, Part II*, volume 13748 of *Lecture Notes in Computer Science*, pages 470–501. Springer, 2022.

IPS08. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Heidelberg, August 2008.

IR19. Yuval Ishai and Vincent Rijmen, editors. *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*. Springer, Heidelberg, May 2019.

Kil88. Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988.

Lam16. Leslie Lamport. Constructing digital signatures from a one way function. 2016.

NR18. Jesper Buus Nielsen and Vincent Rijmen, editors. *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*. Springer, Heidelberg, April / May 2018.

Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In FOCS 1986 [FOC86], pages 162–167.

Appendix

A Proof of Theorem 5.2

Theorem A.1 (Theorem 5.2 Restated.). *There exists a n-party function f that can be computed using a degree 2 polynomial such that no protocol making parallel calls to 2-ary functions can securely compute f with statistical security with abort against a malicious adversary corrupting at most $\lceil n/2 \rceil$ parties.*

Proof. Let f_n be an n-party functionality that takes input $x_i \in \mathbb{F}_4$ from each P_i, and outputs 0 if $x_1 + \ldots + x_n = 0$, and outputs 1 otherwise. Given a protocol Π that securely computes f_n with abort against $\lceil n/2 \rceil$ corruptions we will construct a protocol Π' for f_4 with abort as follows. Let $n = 4t+u$ where $u < 4$. There exist t_1, t_2, t_3, t_4 such that $t_1+\ldots+t_4 = n$, $t_1+t_2 \leq \lceil n/2 \rceil$, $t_1+t_3 \leq \lceil n/2 \rceil$ and $t_1+t_4 \leq \lceil n/2 \rceil$. In Pi', the parties $Q_1, \ldots, Q_4$ with inputs $x_1, \ldots, x_4$ emulate Π. For this, Q_1 emulates $\{P_i\}_{1 \leq i \leq t_1}$ with inputs $x_{1,1}, \ldots, x_{1,t_1}$ randomly chosen subjected to their sum being x_1; Q_2 emulates $\{P_i\}_{t_1+1 \leq i \leq t_2}$ with inputs $x_{2,1}, \ldots, x_{2,t_2}$ randomly chosen subjected to their sum being x_2; and so on. If P_i and $P_{i'}$ are being emulated by the same Q_j, that party computes $\mathcal{O}_{\{i,i'\}}$ according to Π and broadcasts it (in one of its oracle calls); if P_i and $P_{i'}$ belong to distinct Q_j and $Q_{j'}$, then the output of $\mathcal{O}_{\{i,i'\}}$ in Π is computed inside $\mathcal{O}_{\{j,j'\}}$. Finally, each party outputs whatever the emulated parties output (all honest parties output the same value in Π). It is easy to see that the correctness and security of Π' against at most 2 corruptions reduces to correctness and security of Π against at most $\lceil n/2 \rceil$ corruptions. We will now show that Π' is impossible, proving the theorem.

Lemma A.2. *There exists no protocol using parallel calls to 2-ary functions that computes f_4 with statistical security with abort against a malicious adversary corrupting at most 2 parties.*

Proof. Suppose $\Pi' = (\{\mathcal{O}_{\{i,j\}}\}, \mathsf{Enc}, \mathsf{Dec})$ is a secure protocol computing f_4 with abort. Let $\epsilon \geq 0$ be some constant such that the advantage with which any computationally unbounded adversary can break the protocol is at most ϵ. Consider an honest execution of Π' with each x_i is distributed uniformly and independently over $\mathbb{F}_4$. We will denote the output of $\mathcal{O}_{\{i,j\}}$ by $z_{\{i,j\}}$.

Claim A.2.1 *Over the randomness of x_1, x_2, x_3, x_4 and the private randomness of $Q_1, \ldots, Q_4$,*

$$\Pr\Big[\exists (x_1', r_1', x_2', r_2') \text{ s.t. } y'_{(1,2)} = \mathsf{Enc}((1,2), x_1', r_1'), y'_{(2,1)} = \mathsf{Enc}((2,1), x_2', r_2'), \\ \text{and } \mathsf{Dec}(\mathcal{O}_{\{1,2\}}(y'_{(1,2)}, y'_{(2,1)}), \{z_{\{i,j\}}\}_{\{i,j\} \neq \{1,2\}}) \notin \{\perp, f(x_1, x_2, x_3, x_4)\}\Big] \leq 9\epsilon, \tag{13}$$

and

$$\Pr\Big[\exists(x'_1,r'_1,x'_3,r'_3)\ s.t.\ y'_{(1,3)} = \mathsf{Enc}((1,3),x'_1,r'_1), y'_{(3,1)} = \mathsf{Enc}((3,1),x'_3,r'_3),$$
$$and\ \mathsf{Dec}(\mathcal{O}_{\{1,3\}}(y'_{(1,3)},y'_{(3,1)}),\{z_{\{i,j\}}\}_{\{i,j\}\neq\{1,3\}}) \notin \{\perp, f(x_1,x_2,x_3,x_4)\}\Big] \leq 9\epsilon, \tag{14}$$

We first use this claim to prove the theorem. Consider another adversary $\mathcal{A}_{1,4}$ that corrupts P_1 and behaves as described in Figure A.1

$\mathcal{A}_{1,4}$

1. Corrupt P_1, P_4. Sample x_1, x_4, r_1, r_4 uniformly and independently. Invoke all 2-ary functions except $\mathcal{O}_{\{1,3\}}$ using these values to compute the respective outputs. For each $\{i,j\} \neq \{1,3\}$, let the output of $\mathcal{O}_{\{i,j\}}$ be denoted by $z_{\{i,j\}}$.
2. Invoke $\mathcal{O}_{\{1,3\}}$ after computing P_1's encoding using input x_1+1 and randomness r_1; let the output be $\hat{z}_{\{1,3\}}$. Invoke $\mathcal{O}_{\{1,4\}}$ after computing P_1's (resp. P_4's) encoding using input x_1+1 and randomness r_1 (resp. x_4 and r_4); let the output be $\hat{z}_{\{1,4\}}$. $\mathcal{A}_{1,4}$ can compute the output of $\mathcal{O}_{\{1,4\}}$ a second time (in the head) since P_1 and P_4 are corrupted. For each $\{i,j\} \notin \{\{1,2\},\{1,3\},\{1,4\}\}$, set $\hat{z}_{\{i,j\}} = z_{\{i,j\}}$.
3. If there exist $\hat{z}_{\{1,2\}} \in \mathcal{O}_{\{1,2\}}(\cdot,\cdot)$ and $z_{\{1,3\}} \in \mathcal{O}_{\{1,3\}}(\cdot,\cdot)$ such that $\mathrm{Dec}(\{z_{\{i,j\}}\}) = a_0$, $\mathrm{Dec}(\{\hat{z}_{\{i,j\}}\}) = a_1$, and $a_0, a_1 \in \{0,1\}$, output (a_0,a_1); otherwise, output $\perp$.

Fig. A.1.

We argue that Π' is not secure against $\mathcal{A}_{1,4}$ along the same lines as in our proof of Theorem 5.1. Let $\tilde{z}_{\{1,3\}}$ be the output of $\mathcal{O}_{\{1,3\}}$ when invoked with P_1's (resp. P_3's) input being the encoding using input x_1 (resp. x_3) and randomness r_1 (resp. r_3). Then, $\{z_{\{i,j\}}\}_{\{i,j\}\neq\{1,3\}}$ along with $\tilde{z}_{\{1,3\}}$ form the output of all oracle calls in an honest execution of Π' with x_i as input of P_i for each $i \in [4]$. Hence,

$$\Pr\left[\mathsf{Dec}(\{z_{\{i,j\}}\}_{\{i,j\}\neq\{1,3\}}, \tilde{z}_{\{1,3\}}) = f(x_1,x_2,x_3)\right] \geq 1-\epsilon.$$

This along with eq. (14) implies $\Pr[a_1 = f(x_1,x_2,x_3)] \geq 1-10\epsilon$. Using the same line of argument, $\Pr[a_0 = f(1+x_1,x_2,x_3)] \geq 1-10\epsilon$. Thus, in the real execution, $\mathcal{A}_{1,4}$ successfully computes $f(x_1,x_2,x_3)$ and $f(1+x_1,x_2,x_3)$ with probability at least $1-20\epsilon$. Over the randomness of x_2, x_3, for any x_1, $\Pr[f(x_1+1,x_2,x_3) = 1 | f(x_1,x_2,x_3) = 1] = 1/3$, and $f(x_1,x_2,x_3) = 0$ with probability $3/4$. Hence, the simulator fails to simultaneously guess the values of $f(x_1,x_2,x_3)$ and $f(x_1+1,x_2,x_3)$ with probability at least $1/4$. Thus, for sufficiently small ϵ, this contradicts security with at most ϵ distinguishing advantage against an adversary that corrupts at most 2 parties. This proves the theorem. □

Proof. We will prove Equation (4); Equation (5) can be proved similarly. Consider an adversary $\mathcal{A}_{1,2}$ that corrupts P_1 and P_2, and behaves as described in Figure A.2. With respect to $\mathcal{A}_{1,2}$, the proof of the claim is almost identical to

$\mathcal{A}_{1,2}$

1. Corrupt P_1, P_2. Invoke all 2-ary functions except $\mathcal{O}_{\{1,2\}}$ by emulating P_1 and P_2 honestly after setting their inputs to be x_1 and x_2, respectively. Let $z_{\{i,j\}}$ be the output of $\mathcal{O}_{\{i,j\}}$ for eacj $\{i,j\} \neq \{1,2\}$.
2. Compute the output of $\mathcal{O}_{\{1,2\}}$ *in the head* by emulating P_1 and P_2 honestly after setting their inputs to be x_1 and x_2, respectively. Let the output be $z_{\{i,j\}}$. If there exist x'_1, r'_1, x'_2, r'_2 such that, when $y'_{(1,2)} = \mathsf{Enc}((1,2), x'_1, r'_1)$ and $y'_{(2,1)} = \mathsf{Enc}((2,1), x'_2, r'_2)$,

$$\mathsf{Dec}(\mathcal{O}_{\{1,2\}}(y'_{(1,2)}, y'_{(2,1)}), z_{\{1,3\}}, z_{\{2,3\}}) \notin \{\bot, \mathsf{Dec}(z_{\{1,2\}}, z_{\{1,3\}}, z_{\{2,3\}})\},$$

   then call $\mathcal{O}_{\{1,2\}}$ with $y'_{(1,2)}$ and $y'_{(2,1)}$ as inputs of P_1 and P_2 to compute $\widetilde{z}_{\{1,2\}} = \mathcal{O}_{\{1,2\}}(y'_{(1,2)}, y'_{(2,1)})$; else invoke with $y_{(1,2)}$ and $y_{(2,1)}$ as inputs of P_1 and P_2.
3. Output $\mathsf{Dec}(z_{\{1,2\}}, z_{\{1,3\}}, z_{\{2,3\}})$ and terminate.

Fig. A.2.

that of Claim 5.1.1. We leave the details to the reader.

A.1 Conditional disclosure protocol with negligible soundness

In this section, we use the conditional disclosure protocol for verifying one bit input to conditionally reveal one bit secret to build a general conditional disclosure protocol with negiligible soundness.

Description of the Protocol. The protocol is formally described in Figure A.3.

Proof of Security. We prove security against a computationally unbounded non-adaptive adversary in the standalone setting.

Theorem A.3 (Theorem 6.4 Restated.). *For any m, ℓ, λ, $\mathrm{CD}^{\lambda}_{\{1,2\},3}$ in Figure A.3 is a conditional disclosure protocol with P_1 and P_2 verifying P_3 with $\ell(8/9)^{\lambda}$ soundness when P_1 and P_2 have m-bit inputs and P_3 has ℓ-bit input.*

Proof. When all parties are honest, for each $d \in [\ell]$ and $e \in [\lambda]$, the output of $\mathrm{CD}_{\{1,2\},3}$ with input $x_1[d,e]$ and $x_2[d,e]$, respectively, from P_1 and P_2, verifying the input $x_3[d]$ outputs $x_1[d,e] + x_2[d,e]$. Hence, the output of the protocol is $\sum_{d,e} x_1[d,e] + x_2[d,e] = x_1 + x_2$, as required in condition 1 of Definition 6.1.

$\mathrm{CD}^{\lambda}_{\{1,2\},3}$

Let $\lambda \in \mathbb{N}$ be a security parameter, let the inputs of P_1, P_2 and P_3 be $x_1, x_2 \in \{0,1\}^m$ and $x_3 \in \{0,1\}^\ell$, respectively.
Let $\mathrm{CD}_{\{1,2\},3} = (\mathrm{CD}_{\{1,2\},3}.\mathsf{Enc}, \{\mathrm{CD}_{\{1,2\},3}.\mathcal{O}_{\{i,j\}}\}, \mathrm{CD}_{\{1,2\},3}.\mathsf{Dec})$ be the conditional disclosure protocol with 1/9 soundness described in Figure 6.1.

Execution of Enc

1. $\mathsf{Enc}((1,i),x_1)$ for $i \in \{2,3\}$: Sample $\{x_1[d,e]\}_{d,e}$ as additive secret shares of x_1. Output $\{y_{1,i}[d,e]\}_{d,e}$ where $y_{1,i}[d,e] \leftarrow \mathrm{CD}_{\{1,2\},3}.\mathsf{Enc}((1,i), x_1[d,e])$ for all $d \in [\ell]$ and $e \in [\lambda]$.
2. $\mathsf{Enc}((2,i),x_2)$ for $i \in \{1,3\}$: Sample $\{x_2[d,e]\}_{d,e}$ as additive secret shares of x_2. Output $\{y_{2,i}[d,e]\}_{d,e}$ where $y_{2,i}[d,e] \leftarrow \mathrm{CD}_{\{1,2\},3}.\mathsf{Enc}((2,i), x_2[d,e])$ for all $d \in [\ell]$ and $e \in [\lambda]$.
3. $\mathsf{Enc}((3,i),x_3)$ for $i \in \{1,2\}$: Output $\{(x_3[d], \{(c_0[d,e], c_1[d,e])\}_e)\}_d$, where $x_3[d]$ is the d^{th} bit of x_3, and each $c_0[d,e]$ and $c_1[d,e]$ is uniformly and independently chosen from $\mathbb{F}_3$.

Execution of $\mathcal{O}_{\{i,j\}}$

1. $\mathcal{O}_{\{1,2\}}(\{y_{(1,2)}[d,e]\}_{d,e}, \{y_{(2,1)}[d,e]\}_{d,e})$: For each $d \in [\ell]$ and $e \in [\lambda]$, compute $z_{\{1,2\}}[d,e] = \mathrm{CD}_{\{1,2\},3}.\mathcal{O}_{\{1,2\}}(y_{(1,2)}[d,e], y_{(2,1)}[d,e])$. Output $\{z_{\{1,2\}}[d,e]\}$.
2. $\mathcal{O}_{\{1,3\}}(\{y_{(1,3)}[d,e]\}_{d,e}, \{(x_3[d], \{(c_0[d,e], c_1[d,e])\}_e)\}_d)$: For each d, e, compute $z_{\{1,3\}}[d,e] = \mathrm{CD}_{\{1,2\},3}.\mathcal{O}_{\{1,3\}}(y_{(1,3)}[d,e], (x_3[d], c_0[d,e], c_1[d,e]))$. Output $\{z_{\{1,3\}}[d,e]\}$.
3. $\mathcal{O}_{\{2,3\}}(\{y_{(2,3)}[d,e]\}_{d,e}, \{(x_3[d], \{(c_0[d,e], c_1[d,e])\}_e)\}_d)$: For each d, e, compute $z_{\{2,3\}}[d,e] = \mathrm{CD}_{\{1,2\},3}.\mathcal{O}_{\{2,3\}}(y_{(2,3)}[d,e], (x_3[d], c_0[d,e], c_1[d,e]))$. Output $\{z_{\{2,3\}}[d,e]\}$.

(v). Execution of Dec

1. For each d, e, compute $\hat{x}[d,e] = \mathrm{CD}_{\{1,2\},3}.\mathsf{Dec}(z_{\{1,2\}}[d,e], z_{\{1,3\}}[d,e], z_{\{2,3\}}[d,e])$. If $\hat{x}[d,e] = \perp$ for any e, d, output $\perp$ and terminate; else output $\sum_{e,d} \hat{x}[d,e]$.

Fig. A.3. A Conditional disclosure protocol with negligible soundness.

Next, we prove that it satisfies the second condition in Definition 6.1. Fix $d \in [\ell]$ and $e \in [\lambda]$. It is easy to see that the view on an adversary corrupting P_1 and P_2 only reveals the evaluation of the polynomial $x_3[d]u^2 + c_1[d,e]u + c_0[d,e]$ on at most 2 distinct points. Since $c_1[d,e], c_0[d,e]$ are chosen uniformly at random, this reveal no information about $x_3[d]$. In other words, the view of the adversary can be perfectly simulated irrespective of the value of x_3, satisfying the second condition.

Next, we show that the protocol is $\ell(8/9)^\lambda$-sound by via a simulator Sim described below that satisfies the conditions in Definition 6.1 when P_3 is corrupt and P_1 and P_2 are honest.

Let $\mathrm{CD}_{\{1,2\},3}.\mathsf{Sim}$ be the simulator for the 1/9-sound protocol that is described in Figure 6.2. For each $d \in [\ell]$ and $e \in [\lambda]$, Sim instantiates a copy of $\mathrm{CD}_{\{1,2\},3}.\mathsf{Sim}$. For each $d \in [\ell]$ and $e \in [\lambda]$, when $\mathcal{A}$ queries for the output of $\mathcal{O}_{\{1,2\}}$, Sim queries (d,e)'th copy of $\mathrm{CD}_{\{1,2\},3}.\mathsf{Sim}$ for the the output of $\mathcal{O}_{\{1,2\}}$ and forwards it to the adversary. When $\mathcal{A}$ queries for the output of $\mathcal{O}_{\{1,3\}}$ with input $\{(x_3[d], \{c_0[d,e], c_1[d,e]\}_e)\}_d$, for each (d,e), Sim queries (d,e)'th copy of $\mathrm{CD}_{\{1,2\},3}.\mathsf{Sim}$ with $(x_3[d], c_0[d,e], c_1[d,e])$ as the input of the adversary and forwards the response. Sim processes the query for the output of $\mathcal{O}_{\{1,3\}}$ analogously. For any (d,e), when (d,e)'th copy of $\mathrm{CD}_{\{1,2\},3}$ sends $b = 1$ to 'the functionality $\mathcal{F}_{\mathrm{CS}}$', check if, for all $e' in [\lambda] \setminus \{e\}$, (d,e')'th copy of $\mathrm{CD}_{\{1,2\},3}$ has already queried with $b = 1$: if not, respond with a uniformly random bit $\hat{x}_3[d,e]$. Otherwise, query $\mathcal{F}_{\mathrm{CDS}}(x_1, x_2, \cdot)$ with $b = 1$ and obtain $x_1 + x_2$ (if not obtained already), then respond to this query with $\hat{x}[d,e]$ such that $\{\hat{x}_3[d,e]\}_{e \in [\lambda]}$ form an additive secret sharing of $x_1[d] + x_2[d]$. If (d,e)'th copy sends $b = 0$, respond with $\perp$.

To show that this simulation is perfect, we consider a hybrid Hyb in which every real execution of $\mathrm{CD}_{\{1,2\},3}$ is replaced with an ideal execution in which $\mathcal{A}$ interacts with $\mathrm{CD}_{\{1,2\},3}.\mathsf{Sim}$ given access to $\mathcal{F}_{\mathrm{CS}}(\hat{x}_1[d,e], \hat{x}_2[d,e], \cdot)$, where $\{\hat{x}_1[d,e]\}_{d,e}$ and $\{\hat{x}_2[d,e]\}_{d,e}$ are sampled, respectively, as additive secret shares of x_1 and x_2. This hybrid can be shown to be indistinguishable from the real execution by progressively replacing a fresh (d,e)'th copy of $\mathrm{CD}_{\{1,2\},3}$ with a simulator in the real execution, and appealing to perfect indistinguishibility between the real and ideal execution of $\mathrm{CD}_{\{1,2\},3}$.

In detail, let $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{\ell \cdot \lambda}$ be a sequence of subsets of $[\ell] \times [\lambda]$ such that, for each i, $\mathcal{S}_i \setminus \mathcal{S}_{i-1}$ is a singleton set. For each $0 \leq i \leq \ell \cdot \lambda$, consider the Hyb_i in which we replace (d,e)'th copy of $\mathrm{CD}_{\{1,2\},3}$ with an ideal execution using $\mathrm{CD}_{\{1,2\},3}.\mathsf{Sim}$ with access to $\mathcal{F}_{\mathrm{CS}}(\hat{x}_1[d,e], \hat{x}_2[d,e], \cdot)$ for each $(d,e) \in S_i$. Since $\mathcal{S}_0 = \emptyset$, Hyb_0 is the real execution, and since $\mathcal{S}_{\ell \cdot \lambda} = [\ell] \times [\lambda]$, $\mathsf{Hyb}_{\ell \cdot \lambda}$ is identical to Hyb.

We will show that, for any $0 < i \leq \ell \cdot \lambda$, $\mathsf{Hyb}_i \equiv \mathsf{Hyb}_{i-1}$, which implies that the real execution is indistinguishable from Hyb. Let $(d,e) \in \mathcal{S}_i \setminus \mathcal{S}_{i-1}$. The only difference between Hyb_{i-1} and Hyb_i is that, for the unique (d,e) that belongs to $\mathcal{S}_i \setminus \mathcal{S}_{i-1}$, the real world interaction between $\mathcal{A}$ corrupting P_3 and interacting with honest P_1 and P_2 in (d,e)'th copy of $\mathrm{CD}_{\{1,2\},3}$ in the former is replaced with ideal world interaction between $\mathcal{A}$ and $\mathrm{CD}_{\{1,2\},3}.\mathsf{Sim}$ with access to $\mathcal{F}_{\mathrm{CD}}$. Suppose $\mathsf{Hyb}_i \not\equiv \mathsf{Hyb}_{i-1}$.

Consider an adversary $\mathcal{A}'$ that simulates the interaction in Hyb_{i-1} with one notable exception: in the (d,e)'th execution of $\mathrm{CD}_{\{1,2\},3}$, the adversary $\mathcal{A}'$ corrupts P_3, behaves according to $\mathcal{A}$, and interacts with a pair of honest parties P_1 and P_2 with inputs $x_1[d,e]$ and $x_2[d,e]$, respectively. This scenario is identical to Hyb_{i-1}. However, when the interaction between $\mathcal{A}'$ and honest P_1 and P_2 is replaced with interaction between $\mathcal{A}'$ and a simulator as in an ideal execution, the scenario is identical to Hyb_i. Thus, if $\mathsf{Hyb}_i \not\equiv \mathsf{Hyb}_{i-1}$, we contradict perfect indistinguishability of real and ideal execution of $\mathrm{CD}_{\{1,2\},3}$ in the standalone setting.

We conclude this part of the proof by arguing that Hyb is perfectly indistinguishable from the simulation we presented above. This follows from the fact that $\{x_1[d,e] + x_2[d,e]\}$ is identically distributed in both hybrids.

It remains to show that the protocol is $\ell(8/9)^\lambda$-sound. Fix $d \in [\ell]$. Conditioned on the event that the value of $x[d]$ is inconsistent between the query to $\mathcal{O}_{\{1,2\}}$ and $\mathcal{O}_{\{1,3\}}$, probability with which $\mathrm{CD}_{\{1,2\},3}.\mathsf{Sim}$ sends $b = 0$ to $\mathcal{F}_{\mathrm{CD}}$ in the (d,e)'th copy of the simulation is $1/9$, independent of every other copy. Hence, conditioned on this event, $\mathrm{CD}_{\{1,2\},3}.\mathsf{Sim}$ sends $b = 0$ to the functionality in (d,e)'th copy of the simulation for some $e \in [\lambda]$ with probability at least $1 - (8/9)^\lambda$. Which in turn implies that Sim sends $b = 0$ to the corresponding functionality with the same probability conditioned on this event. We obtain the required soundness error by a union bound. This concludes the proof. □

A.2 Proof of Privacy with Knowledge of Output of the Protocol in Figure 6.3

We prove security against a computationally unbounded non-adaptive adversary in the standalone setting via a simulator Sim. The proof extends to the UC setting, and against an adaptive computationally unbounded adversary in a straightforward manner since our simulator is straightline.

Let $\{P_i\}_{i \in C}$ for some $C \subset [n]$ be the set of parties corrupted by $\mathcal{A}$; C is known to Sim. Let $H = [n] \setminus C$. At any point during the interaction, the simulator Sim needs to respond to $\mathcal{A}$'s signal $(\texttt{oracle}, \{i,j\}, \mathrm{Inputs})$ with the output of $\mathcal{O}_{\{i,j\}}$ (if $\mathcal{O}_{\{i,j\}}$ has not be queried so far). Here, Inputs contains the inputs of all corrupt parties, (if any) among $\{P_i, P_j\}$, to $\mathcal{O}_{\{i,j\}}$. Simulator can send a signal $(\texttt{evaluate}, \mathrm{Inputs})$ to the functionality $\mathcal{F}^{\mathrm{PwKO}}(\{x_i\}_{i \in H}, \cdot)$ (once during the execution). Here, Inputs contains the input $\hat{x}_i$ chosen by Sim for each party P_i it has corrupted. In response, the functionality outputs $f(\{x_i\}_{i \in H}, \{\hat{x}_i\}_{i \in H})$. Simulator can send a signal $(\texttt{output}, \mathrm{Output})$ to $\mathcal{F}^{\mathrm{PwKO}}$ (once during the execution), where Output is $\perp$ or belongs to the co-domain of f. In response, $\mathcal{F}^{\mathrm{PwKO}}$ sets the output of all uncorrupted parties to Output.

We will use the following notations in the description of Sim: Let the set of 2-ary functions whose outputs have not yet been queried by $\mathcal{A}$ at any stage of the execution be denoted by $\{\mathcal{O}_{\{i,j\}}\}_{\{i,j\} \in \mathcal{T}}$. Initially, $\mathcal{T} = \binom{[n]}{2}$. The description of the simulator Sim is given in Figure A.4.

Proof of Indistinguishability. We show that w.r.t. any adversary $\mathcal{A}$, the real world and the ideal world executions are computationally indistinguishable via a hybrid argument.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the real world execution of the protocol.
- $\underline{\mathsf{Hyb}_1}$: This hybrid is obtained by making the following changes in the description of Sim in Figure A.4. All the variables used below are already set in the description.

Sim

(i). Initialization of Sim

For each $i \in H$, sample R_i uniformly at random. Sample $\{\gamma_i[k]\}_{k \neq i}$ as an additive secret sharing of $0 \in \mathbb{G}$. Further, for all distinct $j, k \neq i$, sample $s_i[j,k]$ uniformly from $\mathbb{G}$.

(ii). Responding to $(\texttt{oracle}, \{i,j\}, \text{Inputs})$

1. If $i \in C$, receive the input of P_i to $\mathcal{O}_{\{i,j\}}$ from $\mathcal{A}$. Let this be $(\lambda_i, x_{(i,j)}, \gamma_i[j], \{s_i[j,k], s_i[k,j]\}_{k \notin \{i,j\}}, R_{(i,j)})$. If $j \in C$, receive the input of P_j to $\mathcal{O}_{\{i,j\}}$ from $\mathcal{A}$. Let this be $(\lambda_j, x_{(j,i)}, \gamma_j[i], \{s_j[i,k], s_j[k,i]\}_{k \notin \{i,j\}}, R_{(j,i)})$.
2. If $\{i,j\} \subseteq C$, compute and output $\mathcal{O}_{\{i,j\}}$ as described in Π_f using these inputs.
3. For each $k \notin \{i,j\}$: Compute and output $\text{CD}_{\{i,j\},k}.\mathcal{O}_{\{i,j\}}$ as follows:
   (a) If $(\{i,j\} \subseteq H) \wedge (k \in C)$: initialize $\text{CD}_{\{i,j\},k}.\mathsf{Sim}^{\mathcal{F}_{\text{CDS}}(s_i[j,k], s_j[i,k], \cdot)}$ if uninitialized. Forward the response of this simulator when queried for the output of $\text{CD}_{\{i,j\},k}.\mathcal{O}_{\{i,j\}}$.
   (b) Otherwise, compute $y_{(i,j)} = \text{CD}_{\{i,j\},k}.\mathsf{Enc}_{(i,j)}(s_i[j,k], \cdot)$; fresh randomness from $R_{(i,j)}$ (R_i if $i \in H$) is used as private randomness here. Analogously compute $y_{(j,i)}$. Output $\text{CD}_{\{i,j\},k}.\mathcal{O}_{\{i,j\}}(y_{(i,j)}, y_{(j,i)})$.
4. For each $k \notin \{i,j\}$: Compute and output $\text{CD}_{\{j,k\},i}.\mathcal{O}_{\{i,j\}}$ as follows:
   (a) If $(\{j,k\} \subseteq H) \wedge (i \in C)$: initialize $\text{CD}_{\{j,k\},i}.\mathsf{Sim}^{\mathcal{F}_{\text{CDS}}(s_j[k,i], s_k[j,i], \cdot)}$ if uninitialized. Query this simulator for the output of $\text{CD}_{\{i,j\},k}.\mathcal{O}_{\{i,j\}}$ with P_i's input (as provided by $\mathcal{A}$), and forward the response.
   (b) Otherwise, compute $y_{(i,j)} = \text{CD}_{\{j,k\},i}.\mathsf{Enc}_{(i,j)}(x_i[j], \cdot)$. Here $x_i[j] = 0^m$ if $i \in H$. Compute $y_{(j,i)} = \text{CD}_{\{j,k\},i}.\mathsf{Enc}_{(i,j)}(s_j[k,i], \cdot)$. Output $\text{CD}_{\{i,j\},k}.\mathcal{O}_{\{i,j\}}(y_{(i,j)}, y_{(j,i)})$.

   Analogously, compute and output $\text{CD}_{\{i,k\},j}.\mathcal{O}_{\{i,j\}}$.
5. Finally, output $\hat{z}_{\{i,j\}} = z_{\{i,j\}} + \gamma_i[j] + \gamma_j[i] + \sum_{k \notin \{i,j\}} s_i[j,k] + s_j[i,k]$, where $z_{\{i,j\}}$ is sampled as follows: If there exists $\{i',j'\} \in \mathcal{T}$ such that $\{i',j'\} \not\subseteq C$, then sample $z_{\{i,j\}}$ uniformly at random. Otherwise, check if, for all $k \in C$, there exists $\hat{x}_k$ such that, for all $\ell \in H$, $x_k[\ell] = \hat{x}_k$.
   (a) If false, sample $z_{\{i,j\}}$ uniformly at random.
   (b) If true, send the signal $(\texttt{evaluate}, \text{Inputs})$ to $\mathcal{F}^{\text{PwKO}}(\{x_k\}_{k \in [n] \setminus C}, \cdot)$, where the input of each $P_k, k \in C$ is $\hat{x}_k$. Store the output as out. Choose $z_{\{i,j\}}$ such that $\sum_{\{k,\ell\} \not\subseteq C} z_{k,\ell} = \mathsf{out} - \sum_{\{k,\ell\} \subseteq C} h_{\{k,\ell\}}(\hat{x}_k, \hat{x}_\ell)$.

(iii). Choosing the output of the honest parties.

When $\mathcal{T} = \emptyset$, use the outputs of all 2-ary functions to compute the output σ' according to Π_f. Send $(\texttt{output}, \sigma')$ to $\mathcal{F}^{\text{PwKO}}$.

Fig. A.4. Simulator for Π_f.

- Replace instructions in step 3.(a) and 4.(a) as follows:
  In 3.(a), compute $y_{(i,j)} = \mathrm{CD}_{\{i,j\},k}.\mathsf{Enc}_{(i,j)}(s_i[j,k],\cdot)$; fresh randomness from $R_{(i,j)}$ (R_i if $i \in H$) is used as private randomness here. Analogously compute $y_{(j,i)}$. Output $\mathrm{CD}_{\{i,j\},k}.\mathcal{O}_{\{i,j\}}(y_{(i,j)}, y_{(j,i)})$. Make analogous changes in 4.(a).
- Replace instructions in step 4.(b) as follows:
  compute $y_{(i,j)} = \mathrm{CD}_{\{j,k\},i}.\mathsf{Enc}_{(i,j)}(x_i[j],\cdot)$ using fresh randomness from $R_{(i,j)}$; here $x_i[j] = x_i$ and $R_{(i,j)} = R_i$ if $i \in H$.
  Compute $y_{(j,i)} = \mathrm{CD}_{\{j,k\},i}.\mathsf{Enc}_{(j,i)}(s_j[k,i],\cdot)$ using randomness from $R_{(j,i)}$; $R_{(j,i)} = R_j$ if $j \in H$. Output $\mathrm{CD}_{\{i,j\},k}.\mathcal{O}_{\{i,j\}}(y_{(i,j)}, y_{(j,i)})$.
- Make analogous changes in the computation of $\mathrm{CD}_{\{i,k\},j}$.
- In step 5, use $z_{\{i,j\}} = h_{\{i,j\}}(x_i[j], x_j[i])$ while computing $\hat{z}_{\{i,j\}}$. Here $x_i[j]$ (resp. $x_j[i]$) is x_i (resp. x_j) if $i \in H$ (resp. $j \in H$).

- Hyb_2 : In this hybrid, revert changes made in step 4.(b) back to the instructions in the simulator. That is, compute $y_{(i,j)} = \mathrm{CD}_{\{j,k\},i}.\mathsf{Enc}_{(i,j)}(x_i[j],\cdot)$, where $x_i[j] = 0^m$ if $i \in H$. Compute $y_{(j,i)} = \mathrm{CD}_{\{j,k\},i}.\mathsf{Enc}_{(i,j)}(s_j[k,i],\cdot)$. Output $\mathrm{CD}_{\{i,j\},k}.\mathcal{O}_{\{i,j\}}(y_{(i,j)}, y_{(j,i)})$. Make analogous changes in the computation of $\mathrm{CD}_{\{i,k\},j}$.
- Hyb_3 : In this hybrid, revert changes made in steps 3.(a) and 4.(a) back to the instructions in the simulator. That is, in step 3.(a), if uninitialized, initialize $\mathrm{CD}_{\{i,j\},k}.\mathsf{Sim}^{\mathcal{F}(s_i[j,k],s_j[i,k],\cdot)}$. Forward the response of this simulator when queried for the output of $\mathrm{CD}_{\{i,j\},k}.\mathcal{O}_{\{i,j\}}$. In step 4.(a), if uninitialized, initialize $\mathrm{CD}_{\{j,k\},i}.\mathsf{Sim}^{\mathcal{F}(s_j[k,i],s_k[j,i],\cdot)}$. With P_i's input (as provided by $\mathcal{A}$), query this simulator for the output of $\mathrm{CD}_{\{i,j\},k}.\mathcal{O}_{\{i,j\}}$ and forward the response. Make analogous changes in the computation of $\mathrm{CD}_{\{i,k\},j}$.
- Hyb_4 : In this hybrid, revert changes made in steps 5 back to the instructions in the simulator. That is, sample $z_{\{i,j\}}$ as follows: If there exists $\{i',j'\} \in \mathcal{T}$ such that $\{i',j'\} \not\subseteq C$, then sample $\hat{z}_{\{i,j\}}$ uniformly at random. Otherwise, check if, for all $k \in C$, there exists $\hat{x}_k$ such that, for all $\ell \in H$, $x_k[\ell] = \hat{x}_k$.
  1. If false, sample $z_{\{i,j\}}$ uniformly at random.
  2. If true, send the signal $(\texttt{evaluate}, \text{Inputs})$ to $\mathcal{F}^{\mathrm{PwKO}}(\{x_k\}_{k\in[n]\setminus C},\cdot)$, where the input of each $P_k, k \in C$ is $\hat{x}_k$. Store the output as out. Choose $z_{\{i,j\}}$ such that $\sum_{\{k,\ell\}\not\subseteq C} = \mathsf{out} - \sum_{\{k,\ell\}\subseteq C} h_{\{k,\ell\}}(\hat{x}_k, \hat{x}_\ell)$.

In Lemma A.4 we show that Hyb_0 and Hyb_1 are perfectly indistinguishable.
In Lemma A.5 we show that Hyb_1 and Hyb_2 are perfectly indistinguishable.
In Lemma A.6 we show that Hyb_2 and Hyb_3 are perfectly indistinguishable.
In Lemma A.7 we show that Hyb_3 and Hyb_4 are statistically indistinguishable.
We note that Hyb_4 corresponds to the ideal world execution using the simulator.

Lemma A.4. $\mathsf{Hyb}_0 \equiv \mathsf{Hyb}_1$.

Proof. This can be verified by inspection. Note that the view of the adversary is composed of the outputs of all oracle calls, the inputs of corrupt parties and adversary's own private randomness. The description of the simulator has been

changed to involve the real inputs of honest parties in steps 3.(a), 4.(a) and 5. The use of simulator has been reverted back to the actual computation of conditional disclosure protocols. All the variables sampled by honest parties have been sampled according to the same distribution. Finally, the output is computed according to the instructions in the protocol.

Lemma A.5. $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2$.

Proof. Hyb_1 differs from Hyb_2 only in the input of P_i, where $i \in H$, in $\mathrm{CD}_{\{j,k\},i}$ for each $j, k \neq i$. In the former, the actual input x_i is used, whereas, in the latter, a dummy input 0^m is used. Fix $i \in H$, and $j, k \neq i$. Consider an intermediate hybrid $\mathsf{Hyb}_{1,1}$ obtained from Hyb_1 by changing the input of P_i that is verified in $\mathrm{CD}_{\{j,k\},i}$ from x_i to 0^m. We will first prove that $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_{1,1}$.

We will build an adversary $\mathcal{A}'$ for $\mathrm{CD}_{\{j,k\},i}$ that corrupts P_j, P_k in a standalone execution. Note, hybrids Hyb_1 and $\mathsf{Hyb}_{1,1}$ differ only in the input of P_i in $\mathrm{CD}_{\{j,k\},i}$. $\mathcal{A}'$ emulates all the parties (including P_i) according to Hyb_1, with one important difference: the sub-protocol $\mathrm{CD}_{\{j,k\},i}$ in the hybrid is executed by interacting with an (external) honest P_i. When the input of P_i to $\mathrm{CD}_{\{j,k\},i}$ is x_i, this scenario is identical to Hyb_1; whereas, when the input is 0^m, the scenario is identical to $\mathsf{Hyb}_{1,1}$. If there is a distinguisher $\mathcal{D}$ that distinguishes between these two hybrids, then the interaction between $\mathcal{A}'$ and $\mathrm{CD}_{\{j,k\},i}$ will distinguish between an instance where the input of honest P_i is x_i and another where the input of P_i is 0^m. This contradicts perfect security of P_i's input in $\mathrm{CD}_{\{j,k\},i}$ as given in Definition 6.1.

To go from Hyb_1 to $\mathsf{Hyb}_{1,1}$, we changed the input of an honest P_i to $\mathrm{CD}_{\{j,k\},i}$. We can next consider $\mathsf{Hyb}_{1,2}$ obtained from $\mathsf{Hyb}_{1,1}$ by changing the input of $P_{i'}$ in $\mathrm{CD}_{\{j',k'\},i'}$ for $\{i', j', k'\} \neq \{i, j, k\}$ such that $i' \in H$, and similarly argue that $\mathsf{Hyb}_{1,1} \equiv \mathsf{Hyb}_{1,2}$. By proceeding in this manner, we can show that $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2$. □

Lemma A.6. $\mathsf{Hyb}_2 \equiv \mathsf{Hyb}_3$.

Proof. For each $i \in C$ and $j, k \in H$, the real world execution of $\mathrm{CD}_{\{j,k\},i}$ in Hyb_2 has been replaced with an ideal world execution in Hyb_3. This is the only difference between Hyb_2 and Hyb_3. Define $\mathcal{S}$ to be $C \times \binom{H}{2}$. Enumerate the collection $\mathcal{S}$ according to some arbitrary total ordering. For $i = 0, \ldots, |\mathcal{S}|$, let $\mathcal{S}_i$ be the collection of the first i members of $\mathcal{S}$. Define $\mathsf{Hyb}_{2,t}$ to be the hybrid obtained by replacing the real world execution of $\mathrm{CD}_{\{j,k\},i}$ with a simulation for each $(i, \{j, k\}) \in \mathcal{S}_t$. Note, $\mathsf{Hyb}_{2,0} = \mathsf{Hyb}_2$ and $\mathsf{Hyb}_{2,|\mathcal{S}|} = \mathsf{Hyb}_3$. We will prove that, for each $t \in [|\mathcal{S}|]$, $\mathsf{Hyb}_{2,t} \equiv \mathsf{Hyb}_{2,t-1}$, proving the lemma.

To transform $\mathsf{Hyb}_{2,t-1}$ to $\mathsf{Hyb}_{2,t}$, a real world execution of $\mathrm{CD}_{\{j,k\},i}$ (where $(i, \{j, k\})$ is the t'th memeber of $\mathcal{S}$) in which $\mathcal{A}$ corrupts P_i and interacts with honest P_j and P_k with inputs $s_j[k, i]$ and $s_k[j, i]$ has been replaced with interaction of $\mathcal{A}$ with $\mathrm{CD}_{\{j,k\},i}.\mathsf{Sim}$ with access to $\mathcal{F}_{\mathrm{CD}}(s_j[k, i], s_k[j, k], \cdot)$. Consider an adversary $\mathcal{A}_{\{j,k\},i}$ that corrupts P_i in a standalone execution of $\mathrm{CD}_{\{j,k\},i}$ and behaves as follows: $\mathcal{A}_{\{j,k\},i}$ emulates all the parties (including P_j and P_k) according to $\mathsf{Hyb}_{2,t-1}$, with one important difference: the sub-protocol $\mathrm{CD}_{\{j,k\},i}$ in

the hybrid is executed by interacting with the (external) honest P_j and P_k with inputs $s_j[k,i]$ and $s_k[j,i]$, respectively, in Scenario 1, whereas in Scenario 2, it interacts with $\mathrm{CD}_{\{j,k\},i}.\mathsf{Sim}^{\mathcal{F}_{\mathrm{CD}}(s_j[k,i],s_k[j,i],\cdot)}$. Observe that Scenario 1 is equivalent to $\mathsf{Hyb}_{2,t-1}$ and Scenario 2 is equivalent to $\mathsf{Hyb}_{2,t}$. But, by Theorem 6.4, Scenario 1 $\equiv$ Scenario 2. This proves the lemma.

Lemma A.7. $\mathsf{Hyb}_3 \approx_{\epsilon(\lambda)} \mathsf{Hyb}_4$, *where* $\epsilon(\lambda)$ *is a negligible function.*

Proof. Consider a hybrid $\mathsf{Hyb}_{3,1}$ obtained by modifying the instructions for sampling $z_{\{i,j\}}$ in step 5 of Hyb_4 as follows: If there exists $\{i',j'\} \in \mathcal{T}$ such that $\{i',j'\} \not\subseteq C$, then sample $z_{\{i,j\}}$ uniformly at random. Otherwise, choose $z_{\{i,j\}}$ such that $\sum_{\{k,\ell\}\not\subseteq C} z_{\{k,\ell\}} = \sum_{\{k,\ell\}\not\subseteq C} h_{\{k,\ell\}}(x_k[\ell], x_\ell[k])$, where $x_k[\ell]$ is set to x_k for all ℓ if $k \in H$; similarly for $x_\ell[k]$.

The Hyb_3 and $\mathsf{Hyb}_{3,1}$ differ only in the manner in which $z_{\{i,j\}}$ is chosen while computing $\hat{z}_{\{i,j\}}$ for each $\{i,j\} \not\subseteq C$ in step 5. In the former, for each $\{i,j\} \not\subseteq C$, $z_{\{i,j\}} = h_{\{i,j\}}(x_i[j], x_j[i])$. Whereas, in the latter $\{z_{\{i,j\}}\}_{\{i,j\}\not\subseteq C}$ have been chosen as an additive secret sharing of $\sum_{\{i,j\}\not\subseteq C} z_{\{i,j\}} = h_{\{i,j\}}(x_i[j], x_j[i])$.

Claim A.7.1 *When I denotes the indicator function, for each $\{i,j\}$, define*

$$u_{\{i,j\}} = I(i \in H) \cdot \gamma_i[j] + I(j \in H) \cdot \gamma_j[i].$$

Then, $\{u_{\{i,j\}}\}_{\{i,j\}\not\subseteq C}$ form an additive secret sharing of 0.

We will first prove the lemma assuming this claim; the proof of the claim is deferred to the end of this section. It can be verified that, for each $\{i,j\}$, $\gamma_i[j]$ and $\gamma_j[i]$, and hence $u_{\{i,j\}}$ is used exclusively in step 5. The above claim implies that $\{z_{\{i,j\}} + u_{\{i,j\}}\}_{\{i,j\}\not\subseteq C}$ is an additive secret sharing of $\sum_{\{i,j\}\not\subseteq C} z_{\{i,j\}}$. This directly implies that $\mathsf{Hyb}_3 \equiv \mathsf{Hyb}_{3,1}$.

Next, we obtain $\mathsf{Hyb}_{3,2}$ by modifying $\mathsf{Hyb}_{3,1}$. Let $\mathcal{O}_{\{i,j\}}$ be the last 2-ary function that $\mathcal{A}$ queries such that $\{i,j\} \not\subseteq C$; i.e., there exists no $\{i',j'\} \in \mathcal{T}$ such that $\{i',j'\} \not\subseteq C$. We modify the choice of $z_{\{i,j\}}$ in step 5 as follows: If for all $k \in C$, there exists $\hat{x}_k$ such that, for all $\ell \in H$, $x_k[\ell] = \hat{x}_k$, then choose $z_{\{i,j\}}$ such that

$$\sum_{\{k,\ell\}\not\subseteq C} z_{\{k,\ell\}} = f(\hat{x}_1, \ldots, \hat{x}_n) - \sum_{\{k,\ell\}\subseteq C} h_{\{k,\ell\}}(\hat{x}_k, \hat{x}_\ell).$$

If not, make no changes to $\mathsf{Hyb}_{3,1}$. $\mathsf{Hyb}_{3,1} \equiv \mathsf{Hyb}_{3,2}$ by the above claim, since $\sum_{\{i,j\}\not\subseteq C} z_{\{i,j\}} = h_{\{i,j\}\not\subseteq}(\hat{x}_i, \hat{x}_j)$.

Finally, we show that $\mathsf{Hyb}_{3,2} \approx_{\epsilon(\lambda)} \mathsf{Hyb}_4$, where $\epsilon(\lambda)$ is a negligible function. $\mathsf{Hyb}_{3,2}$ and Hyb_4 differ only in the value of $z_{\{i,j\}}$ chosen in step 5 for the aforementioned $\{i,j\}$ when there exists $k \in C$ and $\ell, \ell' \in H$ such that $x_k[\ell] \neq x_k[\ell']$. In Hyb_4, $z_{\{i,j\}}$ is chosen at random, and, in $\mathsf{Hyb}_{3,2}$ it is chosen such that $\sum_{\{k,\ell\}\not\subseteq C} z_{\{k,\ell\}} = h_{\{k,\ell\}\not\subseteq C}(x_k[\ell], x_\ell[k])$.

Recall that $\mathrm{CD}_{\{\ell,\ell'\},k}$ is emulated by an ideal world interaction of $\mathcal{A}$ with $\mathrm{CD}_{\{\ell,\ell'\},k}.\mathsf{Sim}^{\mathcal{F}_{\mathrm{CD}}(s_\ell[\ell',k],s_{\ell'}[\ell,k],\cdot)}$. Further, the masks $s_\ell[\ell',k]$ and $s_{\ell'}[\ell,k]$ added to $z_{\{\ell,\ell'\}}$, which are chosen by Sim uniformly from $\mathbb{G}$ appears only in the emulation of $\mathrm{CD}_{\{\ell,\ell'\},k}$ in both hybrids. The following hold for both hybrids:

1. For each $\{i', j'\} \not\subseteq C$ such that $\{i', j'\} \neq \{i, j\}$, $z_{\{i',j'\}}$ is sampled uniformly from $\mathbb{G}$. In $\hat{z}_{\{\ell,\ell'\}}$, the value of $z_{\{\ell,\ell'\}}$ is masked by $s_\ell[\ell', k] + s_{\ell'}[\ell, k]$ in addition to other masks.
2. Define event E in which $x_k[\ell] \neq x_k[\ell']$ for $k \in C$ and $\ell, \ell' \in H$. The probability of event E is the same independent of the values of $s_\ell[\ell', k]$, $s_{\ell'}[\ell, k]$ and $z_{\{i,j\}}$.
   This can be shown as follows: Suppose w.l.o.g. $\mathcal{A}$ queries $\mathcal{O}_{\{k,\ell\}}$ after $\mathcal{O}_{\{k,\ell'\}}$. When $\mathcal{A}$ prepares the input to $\mathcal{O}_{\{k,\ell\}}$ on behalf of P_k (which fixes event E), the view of $\mathcal{A}$ is identically distributed irrespective of the value of $(s_\ell[\ell', k], s_{\ell'}[\ell, k])$ since $\mathrm{CD}_{\{\ell,\ell'\},k}.\mathsf{Sim}$ queries $\mathcal{F}_{\mathrm{CD}}$ only to respond to the final oracle call in $\mathrm{CD}_{\{\ell,\ell'\},k}$. Furthermore, by definition of $\{i, j\}$, the output of $\mathcal{O}_{\{i,j\}}$ has not been computed at this point. The observation now follows from observation 1.
3. There exists an event $F \subseteq E$, that occurs with the same probability irrespective of the values of $(s_\ell[\ell', k], s_{\ell'}[\ell, k])$ conditioned on E, in which $\mathrm{CD}_{\{\ell,\ell'\},k}.\mathsf{Sim}$ invokes $\mathcal{F}_{\mathrm{CD}}(s_\ell[\ell', k], s_{\ell'}[\ell, k], \cdot)$ with input $b = 0$. By definition of $\mathcal{F}_{\mathrm{CD}}$, in this event, the functionality sends $\perp$ instead of $s_\ell[\ell', k] + s_{\ell'}[\ell, k]$ to the simulator. By $\mathrm{negl}(\lambda)$ soundness of $\mathrm{CD}_{\{\ell,\ell'\},k}$, probability of F conditioned on E is at least $1 - \mathrm{negl}(\lambda)$.

From these observations, we conclude that, conditioned on F, over the randomness of $(s_\ell[\ell', k], s_{\ell'}[\ell, k])$, the view of the adversary is identically distributed when $z_{\{i,j\}}$ is chosen such that $\{z_{\{i,j\}} + u_{\{i,j\}}\}_{\{i,j\} \not\subseteq C}$ form an additive secret sharing of $\sum_{\{i,j\} \not\subseteq C} z_{\{i,j\}}$ and when $z_{\{i,j\}}$ is chosen uniformly at random. Since F occurs with probability $1 - \mathrm{negl}(\lambda)$ conditioned on E, we conclude that $\mathsf{Hyb}_{3,2} \approx_{\epsilon(\lambda)} \mathsf{Hyb}_4$, where $\epsilon(\lambda)$ is a negligible function.

We conclude the proof by proving Claim A.7.1

Proof (of Claim A.7.1). Since $\{\gamma_i[j]\}_{j \neq i}$ is an additive sharing of 0 for each $i \in [n]$, We have

$$\sum_{\{i,j\} \not\subseteq C} u_{\{i,j\}} = \sum_{i \in H} \sum_{j \neq i} \gamma_i[j] = 0.$$

We will show, for each $\emptyset \neq S \subsetneq \{\{i, j\} \text{ s.t. } \{i, j\} \not\subseteq C\}$, $\{u_{\{i,j\}}\}_{\{i,j\} \in S}$ is a tuple of uniform, independent bits. This will prove the claim. By the XOR lemma [CGH+85], this is true if, for each such S, $\sum_{\{i,j\} \in S} u_{\{i,j\}}$ is a uniform bit.

Fix S. Define $H = [n] \setminus C$. There exist distinct i^*, j^*, k^* such that $i^* \in H$, $\{i^*, j^*\} \in S$, and $\{i^*, k^*\} \notin S$. We have

$$\sum_{\{i,j\} \in S} u_{\{i,j\}} = \left(\sum_{j : \{i^*,j\} \in S} s_{i^*}[j] \right) + \left(\sum_{i \in H \setminus \{i^*\}} \sum_{j : \{i,j\} \in S} \gamma_i[j] \right).$$

The first term in RHS of the second equation is a uniform bit that is independent of the second term. This follows from the following observations: $\{i^*, j^*\} \in S$, $\{i^*, k^*\} \notin S$, and $\{s_{i^*}[j]\}_{j \neq i^*}$ is an additive secret sharing of 0. We conclude that the above sum is a uniformly random bit, proving the claim. □

This proves the lemma. □

A.3 Description of the Protocol Achieving General Secure Computation with Abort

Our starting point is the computational semi-malicious MPRE constructed by Applebaum, Brakerski and Tsabary in [ABT18]. We will use this result to effectively convert any given function to a function of effective degree 2.

Theorem A.8 (Theorem 7.3 in [ABT18]). *Assuming the existence of (possibly multi-round) oblivious transfer, every n-party functionality f can be encoded by a computational degree-2 MPRE $\hat{f}$ that is secure against semi-malicious corruption of any set of parties, with complexity polynomial in n and S where S is the size of the circuit that computes f. The MPRE makes a non-black-box use of the oblivious transfer protocol.*

Let f be the n-party function that needs to be computed. Let $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ be a one-time digital signature scheme as defined in Definition 4.3. We derive a signed version of f with respect to this signature scheme, denoted by f_{Sig}, takes input (x_i, sk_i) from each $P_i, i \in [n]$ and computes

$$f_{\mathrm{Sig}}((x_1, \mathsf{sk}_1), \ldots, (x_n, \mathsf{sk}_n)) = (\mathsf{out}, \mathrm{Sig}(\mathsf{out}, \mathsf{sk}_1), \ldots, \mathrm{Sig}(\mathsf{out}, \mathsf{sk}_1)), \quad \text{where } \mathsf{out} = f(x_1, \ldots, x_n) \tag{15}$$

By Theorem A.8, there exist a set of functions $\{h_{\{i,j\}}\}$ such that

$$g(1^\lambda, (x_1, \mathsf{sk}_1; r_1), \ldots, (x_n, \mathsf{sk}_n; r_n)) = \sum_{\{i,j\} \in \binom{[n]}{2}} h_{\{i,j\}}(1^\lambda, (x_i, \mathsf{sk}_i, r_i), (x_j, \mathsf{sk}_j, r_j)).$$

is a semi-malicious computationally secure MPRE.

Let $\Pi_g = (\Pi_g.\mathsf{Enc}, \{\Pi_g.\mathcal{O}_{\{i,j\}}\}, \Pi_g.\mathsf{Dec})$ be the statistically secure non-interactive protocol using 2-ary functions that computes g with PwKO, that is obtained by replacing f with g in Figure 6.3. The protocol is formally described in Figure A.5.

A.4 Correctness of the Protocol in Figure A.5

When all parties behave honestly, by the correctness of Π_g, σ is an MPRE of f_{Sig}, hence $\mathsf{out} = f(x_1, \ldots, x_n)$ and each $t_i = \mathrm{Sig}(\mathsf{out}, \mathsf{sk}_i)$. Hence, output of the decoder in $f(x_1, \ldots, x_n)$ by authenticity of the digital signature scheme.

A.5 Proof of Security the Protocol in Figure A.5

We prove security against a computationally bounded non-adaptive adversary in the standalone setting. The proof extends to the UC setting, and against an adaptive computationally bounded adversary in a straightforward manner since our simulator is straightline.

Let $\{P_i\}_{i \in C}$ for some $C \subset [n]$ be the set of parties corrupted by $\mathcal{A}$; C is known to Sim. At any point during the interaction, the simulator Sim needs to respond

Π_f^{UA}

Let λ be the security parameter. Let g be as defined above. For each $i \in [n]$, let x_i be the input of P_i to f. P_i prepares its input to g as $(x_i, \mathsf{sk}_i, r_i)$, where $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathrm{Gen}(1^\lambda, \cdot)$ and r_i is sampled uniformly.

(i). Execution of Enc

For each $i \in [n]$, and $j \neq i$, $\mathsf{Enc}(1^\lambda, (i,j), (x_i, \mathsf{sk}_i, r_i, vk_i), \cdot) = (y_{(i,j)}, \mathsf{vk}_i)$, where $y_{(i,j)} = \Pi_g.\mathsf{Enc}(1^\lambda, (x_i, \mathsf{sk}_i, r_i), \cdot)$.

(i). Execution of $\mathcal{O}_{\{i,j\}}$

On input $(y_{(i,j)}, \mathsf{vk}_i)$ and $(y_{(j,i)}, \mathsf{vk}_j)$ from P_i and P_j, respectively, $\mathcal{O}_{\{i,j\}}$ outputs $z_{\{i,j\}} = \Pi_g.\mathcal{O}_{\{i,j\}}(y_{(i,j)}, y_{(j,i)})$. Additionally, if $j = i+1$, output vk_i; if $i = j+1$, output vk_j. Here, addition is mod n.

(i). Execution of Dec

Apply $\Pi_g.\mathsf{Dec}(\{z_{\{i,j\}}\})$ to obtain σ. If $\sigma = \perp$, output $\perp$ and terminate. Otherwise, compute $(\mathsf{out}, t_1, \ldots, t_n) = \mathrm{MPRE.Dec}(\sigma)$ where MPRE.Dec is the decoder for g (see Definition 4.2). If $\mathrm{Ver}(\mathsf{out}, t_i, \mathsf{vk}_i) = 1$ for all $i \in [n]$, output out; else abort.

Fig. A.5. Π_f^{UA} computes f with computational security with abort.

to $\mathcal{A}$'s signal $(\texttt{oracle}, \{i,j\}, \mathrm{Inputs})$ with the output of $\mathcal{O}_{\{i,j\}}$ (if $\mathcal{O}_{\{i,j\}}$ has not be queried so far). Here, Inputs contains the inputs of all corrupt parties (if any) among $\{P_i, P_j\}$ to $\mathcal{O}_{\{i,j\}}$. Simulator can send a signal $(\texttt{evaluate}, \mathrm{Inputs})$ to the functionality $\mathcal{F}(\{x_i\}_{i\in H}, \cdot)$ (once during the execution). Here, Inputs contains the input $\hat{x}_i$ chosen by Sim for each party P_i it has corrupted. In response, the functionality outputs $f(\{x_i\}_{i\in H}, \{\hat{x}_i\}_{i\in H})$. Simulator can send either $\texttt{abort}$ or $\texttt{no\_abort}$ to $\mathcal{F}$ (once during the execution). In the former case, $\mathcal{F}$ sets the output of all uncorrupted parties to $\perp$; in the latter, it sets it to the previously computed output.

The description of the simulator Sim is given in Figure A.6.

Proof of Indistinguishability. We show that w.r.t. any adversary $\mathcal{A}$, the real world and the ideal world executions are computationally indistinguishable via a hybrid argument.

- Hyb_0 : This corresponds to the real world execution of the protocol.
- Hyb_1 : This hybrid is derived from the ideal world execution using Sim by replacing the emulation of $\mathcal{G}^{\mathrm{PwKO}}$ in step (i).3 of Figure A.6 by a direct use of $\mathcal{G}^{\mathrm{PwKO}}(\{x_i, \mathsf{sk}_i, r_i\} i \in H, \cdot)$. For this, sample $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathrm{Gen}(1^\lambda, \cdot)$, and r_i uniformly for each $i \in H$. Initialize $\mathcal{G}^{\mathrm{PwKO}}(\{x_i, \mathsf{sk}_i, r_i\} i \in H, \cdot)$ and make the following changes in the description of Sim:
  1. In step (i).3, when $\Pi_g.\mathsf{Sim}$ sends $(\texttt{evaluate}, \{\hat{x}_i, \mathsf{sk}_i, r_i\}_{i\in C})$, forward this to $\mathcal{G}^{\mathrm{PwKO}}(\{x_i, \mathsf{sk}_i, r_i\}_{i\in H}, \cdot)$ and reply with the response.

Sim

Let $\{P_i\}_{i\in C}$ for some $C \subset [n]$ be the set of parties corrupted by $\mathcal{A}$; C is known to Sim. Let $H = [n] \setminus C$.

(i). Responding to $(\texttt{oracle}, \{i,j\}, \text{Inputs})$

Let $\Pi_g.\mathsf{Sim}$ be the simulator for Π_g as described in Figure A.4. The simulator responds to $\mathcal{A}$'s demand for the output of $\mathcal{O}_{\{i,j\}}$ as follows:

1. Send $(\texttt{oracle}, \{i,j\}, \text{Inputs})$ to $\Pi_g.\mathsf{Sim}$, where Inputs contains $y_{(i,j)}$ if $i \in C$, and $y_{(j,i)}$ if $j \in C$. Forward the response to $\mathcal{A}$.
2. If $j = i + 1 \bmod n$, send vk_i, where vk_i is provided by $\mathcal{A}$ if $i \in C$, and sampled according to $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathrm{Gen}(1^\lambda, \cdot)$ otherwise. Similarly, output vk_j if $i = j + 1 \bmod n$.
3. When $\Pi_g.\mathsf{Sim}$ sends $(\texttt{evaluate}, \{\hat{x}_i, \mathsf{sk}_i, r_i\}_{i\in C})$ to $\mathcal{G}^{\mathrm{PwKO}}$, emulate the response of $\mathcal{G}^{\mathrm{PwKO}}$ as follows: Send $(\texttt{evaluate}, \{\hat{x}_i\}_{i\in C})$ to $\mathcal{F}(\{x_i\}_{i\in H}, \cdot)$, and receive $z = f(\{x_i\}_{i\in H}, \{\hat{x}_i\}_{i\in C})$ as output. Recall, we sampled (sk_i, vk_i) for each $i \in H$, and vk_i is provided by $\mathcal{A}$ for each $i \in C$. Compute $f_{\mathrm{Sig}}(\{\hat{x}_i, \mathsf{sk}_i\}) = (z, t_1, \ldots, t_n)$, where $t_i = \mathrm{Sig}(z, \mathsf{sk}_i)$ for each $i \in [n]$. Finally, compute $\mathrm{MPRE}.\mathsf{Sim}(1^\lambda, (z, t_1, \ldots, t_n), \{\hat{x}_i, \mathsf{sk}_i, r_i\}_{i\in C})$, where MPRE.Sim is the simulator for g (see Definition 4.2), and send it to $\Pi_g.\mathsf{Sim}$ as response from functionality $\mathcal{G}^{\mathrm{PwKO}}$.

(ii). Deciding to abort or not.

Receive $(\texttt{output}, \sigma)$ as signal from $\Pi_g.\mathsf{Sim}$ to $\mathcal{G}^{\mathrm{PwKO}}(\{x_i\}_{i\in H}, \cdot)$. If $\sigma = \bot$, send $\texttt{abort}$ to $\mathcal{F}(\{x_i\}_{i\in H}, \cdot)$. Otherwise, let $\sigma = (z', t'_1, \ldots, t'_n)$. Check if there exists $i \in [n]$ such that $\mathrm{Ver}(z', t'_i, \mathsf{vk}_i) = 0$; if so, send $\texttt{abort}$ to $\mathcal{F}(\{x_i\}_{i\in H}, \cdot)$, otherwise send $\texttt{no\_abort}$.

Fig. A.6. Simulator for Π_f^{UA}.

2. In step (ii), if $\sigma = \bot$ or $\sigma = (z', t'_1, \ldots, t'_n)$ such that there exists $i \in [n]$ such that $\mathrm{Ver}(z', t'_i, \mathsf{vk}_i) = 0$, set the outputs to all honest parties to $\bot$, otherwise set them to z'.

- Hyb_2 : This hybrid is obtained from Hyb_1 by reverting the instructions in step (i).3 to that in the description of Sim
- Hyb_3 : This hybrid is obtained from Hyb_2 by reverting the instructions in step (ii) to that in the description of Sim

In Lemma A.9 shows that Hyb_0 and Hyb_1 are statistically indistinguishable.
In Lemma A.10 shows that Hyb_1 and Hyb_2 are computationally indistinguishable.
In Lemma A.11 shows that Hyb_2 and Hyb_3 are computationally indistinguishable.
Note, Hyb_3 corresponds to the ideal world execution using the simulator.

Lemma A.9. $\mathsf{Hyb}_0 \approx_{\epsilon(\lambda)} \mathsf{Hyb}_1$, *where* $\epsilon(\lambda)$ *is a neglibible in* λ.

Proof. Let $\mathsf{View}^g_{\mathcal{A}}$ be the view of the adversary in Π_g that is executed in Π_f^{UA}. When σ is as defined in Π_f^{UA}, Hyb1 is obtained from Hyb_0 by replacing the ensem-

ble $(\mathsf{View}^g_{\mathcal{A}}, \sigma)$ with $\mathsf{Ideal}(1^\lambda, \mathsf{Sim}^{\mathcal{F}(\{x_i, \mathsf{sk}_i, r_i\}_{i\in H}, \cdot)}, \{x_i\}_{i\in C})$, where r_i is sampled uniformly, and $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathrm{Gen}(1^\lambda, \cdot)$ for each $i \in H$. Hence, the indistinguishability of Hyb_0 and Hyb_1 can be reduced to the statitical security of Π_g with PwKO. Thus, the lemma follows from Theorem 6.5 which states that there exists a neglibible function $\epsilon(\lambda)$ such that

$$(\mathsf{View}^g_{\mathcal{A}}, \sigma) \approx_{\epsilon(\lambda)} \mathsf{Ideal}(1^\lambda, \mathsf{Sim}^{\mathcal{F}(\{x_i\}_{i\in H}, \cdot)}, \{x_i\}_{i\in H}).$$

This concludes the proof. □

Lemma A.10. $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$.

Proof. The following is the only difference between Hyb_1 and Hyb_2: When $\Pi_g.\mathsf{Sim}$ sends $(\mathtt{evaluate}, \{\hat{x}_i, \mathsf{sk}_i, r_i\}_{i\in C})$, the response from Sim in Hyb_1 is

$$g(1^\lambda, \{x_i, \mathsf{sk}_i; r_i\}_{i\in H}, \{\hat{x}_i, \mathsf{sk}_i; r_i\}_{i\in C}), \tag{16}$$

where $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathrm{Gen}(1^\lambda, \cdot)$ and r_i is sampled uniformly for each $i \in H$. Whereas, in Hyb_1, the response is

$$\mathrm{MPRE.Sim}\left(1^\lambda, (z, t_1, \ldots, t_n), \{\hat{x}_i, \mathsf{sk}_i, r_i\}_{i\in C}\right), \tag{17}$$

where $z = f(\{x_i\}_{i\in H}, \{\hat{x}_i\}_{i\in C})$ and, for each $i \in [n]$, $t_i = \mathrm{Sig}(z, \mathsf{sk}_i)$ when $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathrm{Gen}(1^\lambda, \cdot), \forall i \in H$. Thus, $(z, t_1, \ldots, t_n) = f_{\mathrm{Sig}}(\{x_i\}_{i\in H}, \{\hat{x}_i\}_{i\in C})$. But, g is a computationally secure MPRE of f_{Sig} that is secure against semi-malicious corruption of C. Hence, the distributions in eq. (16) and eq. (16) are computationally indistinguishable for any $\{\hat{x}_i, \mathsf{sk}_i, r_i\}_{i\in C}$. We conclude that Hyb_0 and Hyb_1 are computationally indistinguishable. □

Lemma A.11. $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$.

Proof. Consider the event E in which, $\sigma = (z', t'_1, \ldots, t'_n)$ such that $\mathrm{Ver}(z', t'_i, vk_i) = 1$ for all $i \in [n]$. The only difference between Hyb_2 and hyb_3 is that, conditioned on E, in Hyb_2 all honest parties output z', whereas, in Hyb_3, they output $f(\{x_i\}_{i\in H}, \{\hat{x}_i\}_{i\in C})$, where $\{\hat{x}_i\}_{i\in C}$ is the inputs that Sim chooses for the corrupt parties when it queries $\mathcal{F}(\{x_i\}_{i\in H}, \cdot)$. Suppose there exists $\{x_i\}_{i\in[n]}$ for which Hyb_2 and Hyb_3 are computationally distinguishable. We stress the view of the adversary is computationally indistinguishable in both hybrids. Hence, this is possible only if, for some polynomial p, in Hyb_2,

$$\Pr\left[z' \neq f(\{x_i\}_{i\in H}, \{\hat{x}_i\}_{i\in C})\right] \geq 1/p(\lambda).$$

We will show that this contradicts the unforgeability guarantee of the one-time signature scheme $(\mathrm{Gen}, \mathrm{Sig}, \mathrm{Ver})$.

Let $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathrm{Gen}(1^\lambda, \cdot)$. Consider an adversary $\mathcal{A}'$ instantiated with vk that behaves as follows: Adversary chooses $i^* \leftarrow H$, and sets $\mathsf{vk}_{i^*} = \mathsf{vk}$. For each $i \in H \setminus \{i^*\}$, it samples $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathrm{Gen}(1^\lambda, \cdot)$. It then executes Hyb_2 with these random variables and $\{x_i\}_{i\in[n]}$, with one significant difference: In step (i).3, when

$\Pi_g.\mathsf{Sim}$ sends $(\texttt{evaluate}, \{\hat{x}_i, \mathsf{sk}_i, r_i\})$, it responds with $(z, \{t_i\}_{i\in[n]\setminus\{i^*\}}, t_{i^*})$, where z is computed as $f(\{x_i\}_{i\in H}, \{\hat{x}_i\}_{i\in C})$; t_i is computed as $\mathrm{Sig}(z, \mathsf{sk}_i)$ using sk_i it has sampled for each $i \in [n] \setminus \{i^*\}$; and t_{i^*} is obtained by querying $\mathrm{Sig}(\cdot, \mathsf{sk})$ to which it has one-time access. Finally, on receiving $(z', t'_1, \ldots, t'_n)$ from $\Pi_g.\mathsf{Sim}$ in step (ii), $\mathcal{A}'$ outputs (z', t'_{i^*}) in the event E, and outputs (z, t_{i^*}) otherwise.

Since $(\mathsf{sk}, \mathsf{vk})$ is correctly sampled, the above interaction is identical to Hyb_2. But then,

$$\Pr\left[z' \neq z, \mathrm{Ver}(z', t_{i^*}, \mathsf{sk}) = 1 \;\middle|\; \begin{array}{l}(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathrm{Gen}(1^\lambda, \cdot)\\ (z, \mathrm{state}) \leftarrow \mathcal{A}'(1^\lambda, \mathsf{vk})\\ t = \mathrm{Sig}(z, \mathsf{sk})\\ (z', t') \leftarrow \mathcal{A}'(1^\lambda, \mathsf{vk}, z, t, \mathrm{state})\end{array}\right] \geq 1/p(\lambda).$$

This contradicts the unforgeability condition in definition 4.3. This proves the lemma. □

B Secure Computation using 3-Oracles

We observed that secure computation using 2-oracles is closely related to the notion of degree-2 MPRE. However, assuming only the existence of one-way functions, degree-2 MPRE's are known only to be secure when there is an honest majority. In this section, we will show that any function can be computed with computational security and unanimous abort using 3-oracles even against a dishonest majority.

Our approach is very similar to that in Section 6 and we provide an informal overview below.

B.1 Overview

Securely Computing Degree-3 Polynomials with PwKO. We now explain our approach to construct a protocol for computing arbitrary degree-3 polynomials by making parallel calls to a 3-ary function and achieves PwKO against a malicious adversary that corrupts an arbitrary subset of the parties. For simplicity, let's assume that each party P_i gets a finite field element x_i as its private input. Let $p(\cdot)$ be a degree-3 polynomial and the parties want to compute $p(x_1, \ldots, x_n) = \sum_{i,j\in[n]} c_{i,j,k} \cdot x_i \cdot x_j \cdot x_k$. In the rest of the overview, we will use $\mathcal{O}_{\{i,j,k\}}$ to denote the function that is computed using P_i, P_j and P_k's inputs. Our construction closely follows the one sketched above for realizing degree-2 polynomials with security with PwKO.

Consider a semi-honest secure protocol for computing p with parallel calls to 3-ary functions that works as follows: $\mathcal{O}_{\{i,j,k\}}$ takes $(x_\ell, s_\ell[\{i,j,k\}])$ from P_ℓ for each $\ell \in \{i,j,k\}$, and outputs $c_{i,j,k} \cdot x_i \cdot x_j \cdot x_k + \sum_{\ell\in\{i,j,k\}} s_\ell[\{i,j,k\}]$ to every party. For every $i \in [n]$, if P_i chooses $\{s_i[\{i,j,k\}]\}_{\{j,k\}\in\binom{[n]\setminus\{i\}}{2}}$ as a random additive secret sharing of 0, then the parties can add the outputs of

all the 3-ary functions to obtain $p(x_1, \ldots, x_n)$. The security once again follows since $\{s_i[\{i,j,k\}]\}$ are all random subject to their sum being 0 and thus, only $p(x_1, \ldots, x_n)$ is revealed.

As we observed in the previous section, it suffices to look at two kinds of attacks: a corrupt party P_i could generate $\{s_i[\{i,j,k\}]\}$ as secret shares of a value other than 0; they could send $(x_i, \cdot)$ to $\mathcal{O}_{\{i,j,k\}}$ and $(x'_i, \cdot)$ (where $x_i \neq x'_i$) to $\mathcal{O}_{\{i,j',k'\}}$. As we previously observed, if corrupt parties disallowed from mounting the latter attack strategy, the above protocol already satisfies PwKO. However, we cannot guarantee security with PwKO when corrupt parties can mount attacks of the former kind. We get around this issue by constructing a protocol that satisfies the following two properties.

1. If every corrupted party P_i sends the same input x_i to every $\mathcal{O}_{\{i,j,k\}}$ where at least one of P_j and P_k is honest, then we want all the honest parties to compute the output of the polynomial subject to the adversary adding some offset.
2. If a corrupted party sends different x_i and $x_{i'}$ to $\mathcal{O}_{\{i,j,k\}}$ and $\mathcal{O}_{\{i,j',k'\}}$ where at least one of P_j and P_k is honest, and a least one of $P_{j'}$ and $P_{k'}$ is honest, then we want the adversary not to learn any information about the private inputs of the honest parties. Specifically, we want the outputs of all the 3-ary functions involving an honest party to be random or abort.

As we established in the previous section, the above two properties are sufficient to show PwKO. At this point, we crucially observe that, if a corrupt P_i reports different inputs to two 3-ary functions–both involving at least one honest party–then, there is an honest party P_j such that P_i has necessarily reported different inputs to two 3-ary functions involving P_j. This can be seen as follows: Suppose corrupt P_i sends distinct x_i and x'_i to $\mathcal{O}_{\{i,j,k\}}$ and $\mathcal{O}_{\{i,j',k'\}}$, where P_j and $P_{j'}$ are honest. If $j = j'$ then we are done. If not, when the input x''_i that P_i sends to $\mathcal{O}_{\{i,j,j'\}}$ is different from x_i, the above statement holds with respect to P_j, whereas, it holds with respect to $P_{j'}$ otherwise.

Using the above observation, we proceed to build the protocol along the lines of our previous construction. We modify our protocol using CDS to switch the output of each $\mathcal{O}_{\{i,j,k\}}$ to random if a corrupt party P_i sends inconsistent inputs to $\mathcal{O}_{\{i,j,k\}}$ and $\mathcal{O}_{\{i,j,k'\}}$, whenever P_j is honest. By arranging this, $s_j[\{i,j,k\}]$ and $s_{j'}[\{i,j,k'\}]$ are erased from the adversary's view. This results in the same domino effect we observed in the previous construction that renders the output of every 3-ary function involving at least one honest party completely random in the eyes of the adversary. The modified protocol involve CDS protocols in which a party P_j checks the consistency of the input provide by another party P_i to 3-ary functions $\mathcal{O}_{\{i,j,k\}}$ and $\mathcal{O}_{\{i,j,k'\}}$ for distinct k, k'. In fact such a CDS protocol is run for every (i,j,k,k') where i,j,k,k' are distinct. Interestingly, the 'verifiers' in such protocols are two copies of the same party participating in two different 3-ary function calls. Realizing this CDS protocol is straightforward: The CDS protocol can be thought of as effectively carried out with a pair of virtual parties P_{i_k} and $P_{i_{k'}}$–both simulated by P_j–as verifiers and P_i as the party whose inputs to $\mathcal{O}_{\{i,j,k\}}$ and $\mathcal{O}_{\{i,j,k'\}}$ are being verified. To realize this in the protocol using

3-ary functions, we compute the 2-ary function between P_k and P_i using $\mathcal{O}_{\{i,j,k\}}$ by having P_j play the role of the virtual P_k. Similarly, 2-ary function between $P_{k'}$ and P_i is computed using $\mathcal{O}_{\{i,j,k'\}}$. The 2-ary function between P_k and $P_{k'}$ can indeed be realized using any 3-ary function involving P_j who is simulating both these virtual parties.

Relaxing to Computational Security. We bootstrap the protocol for computing degree-3 polynomials with PwKO to computing arbitrary functions with stronger security guarantees against computationally bounded adversaries under the assumption of one-way functions.

Our approach is the same as in section 2.3: given a multiparty function f, we build an augmented function g that takes (x_i, sk_i) as input from each P_i, where x_i is P_i's input to f and sk_i is a secret key for a digital signature scheme. g outputs $(y, \sigma_1, \sigma_2, \ldots, \sigma_n)$, where $y = f(x_1, \ldots, x_n)$, and σ_i is the signature of y using P_i's secret key sk_i.

The parties use a degree-3 MPRE with *semi-malicious* security against arbitrary corruptions for computing the function g. Existence of such an MPRE assuming one-way functions is implied by BMR garbling of a gate [BMR90]; we elaborate on this in appendix B.3. As in our contruction in section 2.3, the parties use the previous protocol with PwKO to compute the pre-processing phase, the encoding function for the MPRE, and broadcast the verification key vk_i corresponding to sk_i. After the protocol, each party locally uses the MPRE decoder to learn the output of g, and checks if each σ_i is a valid signature on y under the verification key vk_i. If any of the checks do not pass, the party aborts. The security of the PwKO protocol and the digital signature scheme together imply security of this protocol computing f with abort.

B.2 Computing Degree-3 Functions with PwKO

A deterministic n-party function f that takes input $x_i \in \{0,1\}^m$ from party $P_i, i \in [n]$ has effective degree 3 if it can be decomposed into functions $\{h_{\{i,j,k\}}\}$, where, for each $\{i,j,k\} \in \binom{[n]}{3}$, $h_{\{i,j,k\}} : \{0,1\}^m \times \{0,1\}^m \times \{0,1\}^m \to \mathbb{G}$ for an additive finite group $\mathbb{G}$, such that, for all $x_1, \ldots, x_n$,

$$f(x_1, \ldots, x_n) = \sum_{\{i,j,k\}} h_{\{i,j,k\}}(x_i, x_j, x_k). \tag{18}$$

In this section, we construct a protocol that securely computes any function with effective degree 3. The construction follows the same blueprint as our construction using 2-ary functions for security with PwKO. Notably, this construction uses a variant of conditional disclosure protocol that allows P_i to verify that the inputs of P_j to $\mathcal{O}_{\{i,j,k\}}$ and $\mathcal{O}_{\{i,j,k'\}}$ are consistent in the outer protocol for

distinct k, k'. We denote this variant of conditional disclosure by

$$\begin{aligned}&\mathrm{CD}_{\{i_k,i_{k'}\},j}\\&= \left(\left\{\mathrm{CD}_{\{i_k,i_{k'}\},j}.\mathcal{O}_{\{\ell,\ell'\}}\right\}_{\{\ell,\ell'\}\in\binom{\{i_k,i_{k'},j\}}{3}}, \mathrm{CD}_{\{i_k,i_{k'}\},j}.\mathsf{Enc}, \mathrm{CD}_{\{i_k,i_{k'}\},j}.\mathsf{Dec}\right)\end{aligned} \tag{19}$$

since the 'parties' carrying out the verification is simply two copies of P_i. Just like the previous variant, this is an n-party non-interactive protocol with that effectively delivers a secret provided by P_i to all parties if P_k's input to $\mathcal{O}_{\{i,j,k\}}$ and $\mathcal{O}_{\{i,j,k'\}}$ are consistent. If P_k uses inconsistent inputs, then all parties detect malpractice and abort; furthermore, the secret of P_i is kept hidden from the adversary. On the other hand, when P_i is corrupt, the protocol ensures perfect privacy of P_j's input. This protocol works exactly like its previous variant after creating two copies of P_i–namely, P_{i_k} and $P_{i_{k'}}$ both simulated by P_i and executing $\mathrm{CD}_{\{i_k,i_{k'}\},j}$. In the 3-oracle setting, the 2-oracle between P_{i_k} and P_j ($\mathcal{O}_{\{i_k,j\}}$) is realized using $\mathcal{O}_{\{i,j,k\}}$, that between $P_{i_{k'}}$ and P_j is realized using $\mathcal{O}_{\{i,j,k'\}}$, and, finally, that between P_{i_k} and $P_{i_{k'}}$ is simulated by P_i and broadcasted using one of the oracles, say $\mathcal{O}_{\{i,j,k\}}$.

Description of the Protocol. Let f be a deterministic n-party function of effective degree 3, and let $\{h_{\{i,j,k\}}\}$ be as described in eq. (18). Let $\lambda \in \mathbb{N}$ be a security parameter. The Figure B.1 provides a formal description of a protocol that securely computes f with PwKO when the input of each party P_i is x_i.

The construction uses the following resources and notations: We denote the set of all ordered d-tuples of distinct elements of a set S by $\binom{S}{d}_{\mathrm{ord}}$. Let $\mathrm{CD}_{\{i_k,i_{k'}\},j}$ be a conditional disclosure protocol with $\mathrm{negl}(\lambda)$ soundness as defined in (19). For conciseness, we will drop 1^λ in the argument of $\mathrm{CD}_{\{i_k,i_{k'}\},j}.\mathsf{Enc}$.

For legibility, we suppress the private randomness used in the encoder of Π_f as well as the conditional disclosure protocols invoked as sub-protocols. But, we stress that, an honest party uses the same private randomness while invoking the encoder to obtain their input to different 3-ary oracles.

Correctness of the protocol. When all parties behave honestly, for distinct i, j and $\{k, k'\} \in \binom{[n]\setminus\{i,j\}}{2}$, the the output of $\mathrm{CD}_{\{i_k,i_{k'}\},j}$ is $s_{\{i_k,i_{k'}\},j} = s_{i_k}[i_{k'}, j] + s_{i_{k'}}[i_k, j]$. This follows from the correctness of conditional disclosure protocol. Furthermore, $\{\gamma_i[\{i,\ell,\ell'\}]\}_{\{\ell,\ell'\}\in\binom{[n]\setminus\{i\}}{2}}$ forms an additive secret sharing of 0.

Π_f

(i). Execution of Enc

For any $\{i,j,k\} \in \binom{[n]}{3}$,

$$\left(1^\lambda, x_i, \gamma_i[\{i,j,k\}], \{s_{i_\ell}[i_{\ell'}, \ell']\}_{(\ell,\ell',\ell'') \in \binom{[n]\setminus\{i\}}{3}_{\text{ord}}}, R_i\right) \leftarrow \mathsf{Enc}(1^\lambda, (i, \{i,j,k\}), x_i, \cdot). \tag{20}$$

Here, $\gamma_i[\{i,j,k\}]$ is such that $\{\gamma_i[\{i,\ell,\ell'\}]\}_{\{\ell,\ell'\} \in \binom{[n]\setminus\{i\}}{2}}$ form an additive secret sharing of $0 \in \mathbb{G}$; for $(\ell,\ell',\ell'') \in \binom{[n]\setminus\{i\}}{3}_{\text{ord}}$, $s_{i_\ell}[i_{\ell'}, \ell'']$ is uniform in $\mathbb{G}$; R_i is a uniformly random string used to derive private randomness for the encoders of various parallel invocations of conditional disclosure protocols; see (ii).2.(a-c). Recall, an honest P_i uses the same randomness (and input x_i) while invoking $\mathsf{Enc}(1^\lambda, (i, \{i,j,k\}), \cdot, \cdot)$ for all admissible j,k. This is crucial in ensuring the required correlation among P_i's inputs to all oracles involving P_i.

Fig. B.1. Description of the encoder of Π_f that computes f of effective degree 3 with privacy with knowledge of output.

Hence, by (21), the decoder outputs

$$\begin{aligned}
&\sum_{\{i,j,k\} \in \binom{[n]}{3}} \hat{z}_{\{i,j,k\}} - \sum_{\{i,j,k,k'\} \in \binom{[n]}{4}} \left(s_{\{i_k, i_{k'}\}, j} + s_{\{j_k, j_{k'}\}, i}\right) \\
&= \sum_{\{i,j,k\} \in \binom{[n]}{3}} \left(\hat{z}_{\{i,j,k\}} - \sum_{(a,b,c) \in \binom{\{i,j,k\}}{3}_{\text{ord}}} \sum_{\ell \notin \{i,j,k\}} s_{a_b}[a_\ell, c] \right) \\
&= \sum_{\{i,j,k\} \in \binom{[n]}{3}} h_{\{i,j,k\}}(1^\lambda, x_i, x_j, x_k).
\end{aligned}$$

This proves the correctness of the protocol.

The proof of security with PwKO proceeds as outlined in the technical overview. A formal proof is provided in Appendix B.4. Along with correctness, this implies the following theorem.

Theorem B.1. *The protocol Π_f in Figure B.1 computes any n-party function f of effective degree 3 with statistical security while guaranteeing privacy with knowledge output against a malicious adversary that corrupts any set of parties.*

B.3 Achieving General Secure Computation with Abort

We use the protocol developed in the previous section to realize general secure computation with abort against computationally bounded adversaries assuming

Π_f Contd.

(ii). Execution of $\mathcal{O}_{\{i,j,k\}}$

Let P_i's input be $(1^{\lambda_i}, x_i, \gamma_i[\{i,j,k\}], \{s_{i_\ell}[i_{\ell'}, \ell'']\}, R_i)$;
let P_j's input be $(1^{\lambda_j}, x_j, \gamma_j[\{i,j,k\}], \{s_{j_\ell}[j_{\ell'}, \ell'']\}, R_j)$;
let P_k's input be $(1^{\lambda_k}, x_k, \gamma_k[\{i,j,k\}], \{s_{k_\ell}[k_{\ell'}, \ell'']\}, R_k)$.
$\mathcal{O}_{\{i,j,k\}}$ behaves as follows:

1. If $\lambda_j \neq \lambda_i$, output $\perp$ and terminate; otherwise, let $\lambda = \lambda_i$.
   Output $\hat{z}_{\{i,j,k\}}$ where

$$\hat{z}_{\{i,j,k\}} = h_{\{i,j,k\}}(1^\lambda; x_i, x_j, x_k) + \sum_{(a,b,c)\in\binom{\{i,j,k\}}{3}_{\text{ord}}} \sum_{\ell\notin\{i,j,k\}} s_{a_b}[a_\ell, c] + \sum_{\ell\in\{i,j,k\}} \gamma_\ell[\{i,j,k\}]. \quad (21)$$

2. For each $\ell \notin \{i,j,k\}$:
   (a) Output $\text{CD}_{\{i_k,i_\ell\},j}.\mathcal{O}_{\{i_k,j\}}(y_{(i_k,j)}, y_{(j,i_k)})$, where

$$y_{(i_k,j)} \leftarrow \text{CD}_{\{i_k,i_\ell\},j}.\mathsf{Enc}_{(i_k,j)}(s_{i_k}[i_\ell, j], \cdot),$$
$$y_{(j,i_k)} \leftarrow \text{CD}_{\{i_k,i_\ell\},j}.\mathsf{Enc}_{(j,i_k)}(x_j, \cdot).$$

   Note, fresh randomness $r_{i_k,(\{i_k,i_\ell\},k)}$ from R_i is used to compute P_i's encodings $\text{CD}_{\{i_k,i_\ell\},j}.\mathsf{Enc}_{(i_k,j)}$ and $\text{CD}_{\{i_k,i_\ell\},j}.\mathsf{Enc}_{(i_k,i_\ell)}$; similarly for P_j.
   (b) Output $\text{CD}_{\{i_j,i_\ell\},k}.\mathcal{O}_{\{i_j,k\}}(y_{(i_j,k)}, y_{(k,i_j)})$ with analogous $y_{(i_j,k)}, y_{(k,i_j)}$.
   (c) Output $\text{CD}_{\{j_k,j_\ell\},i}.\mathcal{O}_{\{j_k,i\}}(y_{(j_k,i)}, y_{(i,j_k)})$ with analogous $y_{(j_k,i)}, y_{(i,j_k)}$.
   (d) Output $\text{CD}_{\{j_i,j_\ell\},k}.\mathcal{O}_{\{j_i,k\}}(y_{(j_i,k)}, y_{(k,j_i)})$ with analogous $y_{(j_i,k)}, y_{(k,j_i)}$.
   (e) Output $\text{CD}_{\{k_j,k_\ell\},i}.\mathcal{O}_{\{k_j,i\}}(y_{(k_j,i)}, y_{(i,k_j)})$ with analogous $y_{(k_j,i)}, y_{(i,k_j)}$.
   (f) Output $\text{CD}_{\{k_i,k_\ell\},j}.\mathcal{O}_{\{k_i,j\}}(y_{(k_i,j)}, y_{(j,k_i)})$ with analogous $y_{(k_i,j)}, y_{(j,k_i)}$.
3. For $(\ell,\ell',\ell'') \in \binom{[n]\setminus\{i\}}{3}_{\text{ord}}$: output $\text{CD}_{\{i_\ell,i_{\ell'}\},\ell''}.\mathcal{O}_{\{i_\ell,i_{\ell'}\}}(y_{(i_\ell,i_{\ell'})}, y_{(i_{\ell'},i_\ell)})$, where

$$y_{(i_\ell,i_{\ell'})} \leftarrow \text{CD}_{\{i_\ell,i_{\ell'}\},\ell''}.\mathsf{Enc}_{(i_\ell,i_{\ell'})}(s_{i_\ell}[i_{\ell'}, \ell''], \cdot),$$
$$y_{(i_{\ell'},i_\ell)} \leftarrow \text{CD}_{\{i_\ell,i_{\ell'}\},\ell''}.\mathsf{Enc}_{(i_{\ell'},i_\ell)}(s_{i_{\ell'}}[i_\ell, \ell''], \cdot).$$

4. For $(\ell,\ell',\ell'') \in \binom{[n]\setminus\{i\}}{3}_{\text{ord}}$: output $\text{CD}_{\{j_\ell,j_{\ell'}\},\ell''}.\mathcal{O}_{\{j_\ell,j_{\ell'}\}}(y_{(j_\ell,j_{\ell'})}, y_{(j_{\ell'},j_\ell)})$.
5. For $(\ell,\ell',\ell'') \in \binom{[n]\setminus\{i\}}{3}_{\text{ord}}$: output $\text{CD}_{\{k_\ell,k_{\ell'}\},\ell''}.\mathcal{O}_{\{k_\ell,k_{\ell'}\}}(y_{(k_\ell,k_{\ell'})}, y_{(k_{\ell'},k_\ell)})$.

Fig. B.2. Description of the 3-ary oracles in Π_f that computes f of effective degree 3 with privacy with knowledge of output.

only one-way functions. For this, we once again rely on a computational semi-malicious degree-3 MPRE. As outlined in appendix B.1, this construction is the same as that in section 6.3 with the secure computation of degree-2 MPRE with

Π_f Contd.

(iii). Execution of Dec.

For distinct i, j and $\{k, k'\} \in \binom{[n]\setminus\{i,j\}}{2}$, let $s_{\{i_k,i_{k'}\},j}$ be the output of $\mathsf{CD}_{\{i_k,i_{k'}\},j}$. If $\hat{z}_{\{i,j,k\}}$ is $\perp$ for some i, j, k or $s_{\{i_k,i_{k'}\},j}$ is $\perp$ for some i, j, k, k', output $\perp$ and terminate. Otherwise, output

$$\sum_{\{i,j,k\}\in\binom{[n]}{3}} \hat{z}_{\{i,j,k\}} - \sum_{\{i,j,k,k'\}\in\binom{[n]}{4}} (s_{\{i_k,i_{k'}\},j} + s_{\{j_k,j_{k'}\},i}).$$

Fig. B.3. Π_f computes f of effective degree 3 with privacy with knowledge of output.

PwKO in the latter replaced with secure computation of degree-3 MPRE with PwKO.

In the following discussion, we will establish the existence of computational semi-malicious degree-3 MPRE. The existence of such an MPRE is implied by the multiparty garbling function used for garbling NAND gates in the BMR protocol [BMR90]. In [GIS18], the authors observed that the n-party multiparty garbling of a NAND gate g with input wires a and b and output wire c in [BMR90] is a set of values $\{G^i_{r_1,r_2}\}_{i\in[n],r_0,r_1\in\{0,1\}}$, where $G^i_{r_1,r_2}$ is computed as

$$\left(\bigoplus_{i=1}^{n} F_{k^i_{a,r_1}}(g,j,r_1,r_2) \oplus F_{k^i_{b,r_2}}(g,j,r_1,r_2)\right) \oplus k^j_{c,0} \oplus (\chi_{r_1,r_2} \wedge (k^j_{c,0} \oplus k^j_{c,1})), \tag{22}$$

where $\chi_{r_1,r_2} = ((\oplus_{i=1}^n \lambda_{i,a} \oplus r_1) \cdot (\oplus_{i=1}^n \lambda_{i,b} \oplus r_2) \oplus 1) \oplus (\oplus_{i=1}^n \lambda_{i,c})$. Here, F is a PRF, $k^i_{x,r}$ for $x \in \{a, b, c\}$ and $r \in \{0, 1\}$ is a PRF key, and $\lambda_{i,x}$ is a bit for $x \in \{a, b, c\}$.

By inspecting $G^i_{r_1,r_2}$, and noting that keys $k^i_{x,r}$ and $\lambda_{i,x}$ are chosen by each party P_i, it is easy to see that $G^i_{r_1,r_2}$ has effective degree 3, when the computation of $F_{k^i_{a,r_1}}(g, j, r_1, r_2)$ and $F_{k^i_{b,r_2}}(g, j, r_1, r_2)$ are carried out in a preprocessing phase, as the inputs to PRF is provided by a single party.

As in our construction using 2-MPRE in section 6.3, the protocol computing functions of effective degree-3 using 3-ary oracles is used to first apply a local function on the parties private inputs–this is used to compute the value of PRF on the party's key–and then compute a degree-3 polynomial on the outputs of these local functions. This allows us to compute the pre-processing phase inside the 3-ary oracles and thus, allowing us to rely on an MPRE that is secure against semi-malicious adversaries. Note, this necessarily requires a PRF to be computed inside the 3-ary oracles; thus, the protocol makes non-black box use of PRF.

The construction and security analysis of the protocol sketched above follow as a straightforward generalization of our construction for theorem 6.6; hence, we leave out these details. Using such a construction, we get the result in theorem 6.6 as PRF is implied by one-way functions.

B.4 Proof of Privacy with Knowledge of Output of the Protocol in Figures B.1, B.2 and B.3

We prove security against a computationally unbounded non-adaptive adversary in the standalone setting via a simulator Sim. The proof extends to the UC setting, and against an adaptive computationally unbounded adversary in a straightforward manner since our simulator is straightline.
Let $\{P_i\}_{i\in C}$ for some $C \subset [n]$ be the set of parties corrupted by $\mathcal{A}$; C is known to Sim. Let $H = [n] \setminus C$. At any point during the interaction, the simulator Sim needs to respond to $\mathcal{A}$'s signal $(\texttt{oracle}, \{i,j,k\}, \text{Inputs})$ with the output of $\mathcal{O}_{\{i,j,k\}}$ (if $\mathcal{O}_{\{i,j,k\}}$ has not be queried so far). Here, Inputs contains the inputs of all corrupt parties, (if any) among $\{P_i, P_j, P_k\}$, to $\mathcal{O}_{\{i,j,k\}}$. Simulator can send a signal $(\texttt{evaluate}, \text{Inputs})$ to the functionality $\mathcal{F}^{\text{PwKO}}(\{x_i\}_{i\in H}, \cdot)$ (once during the execution). Here, Inputs contains the input $\hat{x}_i$ chosen by Sim for each party P_i it has corrupted. In response, the functionality outputs $f(\{x_i\}_{i\in H}, \{\hat{x}_i\}_{i\in H})$. Simulator can send a signal $(\texttt{output}, \text{Output})$ to $\mathcal{F}^{\text{PwKO}}$ (once during the execution), where Output is $\perp$ or belongs to the co-domain of f. In response, $\mathcal{F}^{\text{PwKO}}$ sets the output of all uncorrupted parties to Output.

We will use the following notations in the description of Sim: Let the set of 3-ary functions whose outputs have not yet been queried by $\mathcal{A}$ at any stage of the execution be denoted by $\{\mathcal{O}_{\{i,j,k\}}\}_{\{i,j,k\}\in\mathcal{T}}$. Initially, $\mathcal{T} = \binom{[n]}{3}$. The description of the simulator Sim is given in Figure B.4.

Proof of Indistinguishability. We show that w.r.t. any adversary $\mathcal{A}$, the real world and the ideal world executions are computationally indistinguishable via a hybrid argument.

- $\underline{\mathsf{Hyb}_0}$: This corresponds to the real world execution of the protocol.
- $\underline{\mathsf{Hyb}_1}$: This hybrid is obtained by making the following changes in the description of Sim in Figure A.4. All the variables used below are already set in the description.
  - Replace instructions in step 3.(b) as follows:
    When $a \in H$ and $c \in C$, compute the output of $\text{CD}_{\{a_b,a_\ell\},c}.\mathcal{O}_{\{a_b,c\}}$ is computed as described in Π_f using the input of P_a as sampled by Sim and the input of P_c as provided by the adversary.
  - Replace instructions in step 3.(c) as follows when $c \in H$:
    The output of $\text{CD}_{\{a_b,a_\ell\},c}.\mathcal{O}_{\{a_b,c\}}$ is computed as described in Sim after changing the input of P_c from 0 as sampled by Sim to the true value provided by the environment.
  - In step 4, use $z_{\{i,j,k\}} = h_{\{i,j,k\}}(x_i, x_j, x_k)$ while computing $\hat{z}_{\{i,j,k\}}$. Here x_i (resp. x_j and x_k) are as provided by $\mathcal{A}$ if $i \in C$ (resp. $j \in C$ and $k \in C$), and as provided by the environment if $i \in H$ (resp. $j \in H$ and $k \in H$).
- $\underline{\mathsf{Hyb}_2}$: In this hybrid, revert changes made in step 3(c), when $c \in H$, back to the instructions in the simulator. That is, the output of $\text{CD}_{\{a_b,a_\ell\},c}.\mathcal{O}_{\{a_b,c\}}$ is computed as described in Sim using 0 as the input of P_c as used in Sim.

Sim

(i). Initialization of Sim

For each $i \in H$, set $x_i = 0$, and sample R_i uniformly at random. Sample $\{\gamma_i[\{i,\ell,\ell'\}]\}_{\{\ell,\ell'\}\in\binom{[n]\setminus\{i\}}{2}}$ as an additive secret sharing of $0 \in \mathbb{G}$. Further, for all distinct $j,k,k' \neq i$, sample $s_{i_k}[i_{k'},j]$ uniformly from $\mathbb{G}$.

(ii). Responding to $(\texttt{oracle}, \{i,j,k\}, \text{Inputs})$

1. If $i \in C$, receive the input of P_i to $\mathcal{O}_{\{i,j\}}$ from $\mathcal{A}$. Let this be
$$\left(1^\lambda, x_i, \gamma_i[\{i,j,k\}], \{s_{i_\ell}[i_{\ell'},\ell']\}_{(\ell,\ell',\ell'')\in\binom{[n]\setminus\{i\}}{3}_{\text{ord}}}, R_i\right)$$
Similarly, if $j \in C$ (resp. $k \in C$), receive the input of P_j (resp. P_k) to $\mathcal{O}_{\{i,j\}}$ from $\mathcal{A}$.
2. If $\{i,j,k\} \subseteq C$, compute and output $\mathcal{O}_{\{i,j,k\}}$ as described in Π_f using these inputs.
3. For any $(a,b,c) \in \binom{\{i,j,k\}}{3}_{\text{ord}}$ and $\ell \notin \{i,j,k\}$:
   (a) The output of $\text{CD}_{\{a_b,a_\ell\},c}.\mathcal{O}_{\{a_b,a_\ell\}}$ is computed as described in Π_f using the input of P_a as provided by the $\mathcal{A}$ when $a \in C$ and as sampled by Sim when $a \in H$.
   (b) When $a \in H$ and $c \in C$, compute the output of $\text{CD}_{\{a_b,a_\ell\},c}.\mathcal{O}_{\{a_b,c\}}$ as follows: initialize $\text{CD}_{\{a_b,a_\ell\},c}.\mathsf{Sim}^{\mathcal{F}_{\text{CDS}}(a_b[a_\ell,c],a_\ell[a_b,c],\cdot)}$ if uninitialized. Query this simulator for the output of $\text{CD}_{\{a_b,a_\ell\},c}.\mathcal{O}_{\{a_b,c\}}$ with P_i's input (as provided by $\mathcal{A}$), and forward the response.
   (c) Otherwise, the output of $\text{CD}_{\{a_b,a_\ell\},c}.\mathcal{O}_{\{a_b,c\}}$ is computed as described in Π_f using the input of P_a (resp. P_b) as provided by the $\mathcal{A}$ if $a \in C$ (resp. $c \in C$) and as sampled by Sim when $a \in H$ (resp. $c \in H$).

Fig. B.4. Simulator for Π_f.

- Hyb_3 : In this hybrid, revert changes made in steps 3(b) back to the instructions in the simulator. That is, when $a \in H$ and $c \in C$, compute the output of $\text{CD}_{\{a_b,a_\ell\},c}.\mathcal{O}_{\{a_b,c\}}$ as follows: initialize $\text{CD}_{\{a_b,a_\ell\},c}.\mathsf{Sim}^{\mathcal{F}_{\text{CDS}}(a_b[a_\ell,c],a_\ell[a_b,c],\cdot)}$ if uninitialized. Query this simulator for the output of $\text{CD}_{\{a_b,a_\ell\},c}.\mathcal{O}_{\{a_b,c\}}$ with P_i's input (as provided by $\mathcal{A}$), and forward the response.
- Hyb_4 : In this hybrid, revert changes made in steps 4 back to the instructions in the simulator. That is, $z_{\{i,j,k\}}$ is chosen as follows: if there exists $\{i',j',k'\} \in \mathcal{T}$ such that $\{i',j',k'\} \not\subseteq C$, then sample $z_{\{i,j,k\}}$ uniformly at random. Otherwise, check if, for every $\ell \in C$, there exists $\hat{x}_\ell$ such that, for all $\{\ell,\ell',\ell''\} \not\subseteq C$, the adversary used $\hat{x}_\ell$ while invoking $\mathcal{O}_{\{\ell,\ell',\ell''\}}$.
  1. If false, sample $z_{\{i,j,k\}}$ uniformly at random.
  2. If true, send the signal $(\texttt{evaluate}, \text{Inputs})$ to $\mathcal{F}^{\text{PwKO}}(\{x_k\}_{k\in[n]\setminus C}, \cdot)$, where the input of each $P_\ell, \ell \in C$ is $\hat{x}_\ell$. Store the output as out. Choose $z_{\{i,j,k\}}$

Sim Contd.

4. Finally, output

$$\hat{z}_{\{i,j,k\}} = z_{\{i,j,k\}} + \sum_{(a,b,c)\in\binom{\{i,j,k\}}{3}_{\text{ord}}} \sum_{\ell\notin\{i,j,k\}} s_{a_b}[a_\ell, c] + \sum_{\ell\in\{i,j,k\}} \gamma_\ell[\{i,j,k\}].$$

where $z_{\{i,j,k\}}$ is sampled as follows: If there exists $\{i',j',k'\} \in \mathcal{T}$ such that $\{i',j',k'\} \not\subseteq C$, then sample $z_{\{i,j\}}$ uniformly at random. Otherwise, check if, for every $\ell \in C$, there exists $\hat{x}_\ell$ such that, for all $\{\ell,\ell',\ell''\} \not\subseteq C$, the adversary used $\hat{x}_\ell$ while invoking $\mathcal{O}_{\{\ell,\ell',\ell''\}}$.

(a) If false, sample $z_{\{i,j,k\}}$ uniformly at random.
(b) If true, send the signal $(\texttt{evaluate}, \text{Inputs})$ to $\mathcal{F}^{\text{PwKO}}(\{x_k\}_{k\in[n]\setminus C}, \cdot)$, where the input of each $P_\ell, \ell \in C$ is $\hat{x}_\ell$. Store the output as out. Choose $z_{\{i,j,k\}}$ such that

$$\sum_{\{\ell,\ell',\ell''\}\not\subseteq C} z_{\ell,\ell',\ell''} = \mathsf{out} - \sum_{\{\ell,\ell',\ell''\}\subseteq C} h_{\{\ell,\ell',\ell''\}}(\hat{x}_\ell, \hat{x}_{\ell'}, \hat{x}_{\ell''}).$$

(iii). Choosing the output of the honest parties.

When $\mathcal{T} = \emptyset$, use the outputs of all 3-ary functions to compute the output σ' according to Π_f. Send $(\texttt{output}, \sigma')$ to $\mathcal{F}^{\text{PwKO}}$.

Fig. B.5. Simulator for Π_f.

such that

$$\sum_{\{\ell,\ell',\ell''\}\not\subseteq C} z_{\ell,\ell',\ell''} = \mathsf{out} - \sum_{\{\ell,\ell',\ell''\}\subseteq C} h_{\{\ell,\ell',\ell''\}}(\hat{x}_\ell, \hat{x}_{\ell'}, \hat{x}_{\ell''}).$$

In Lemma B.2 we show that Hyb_0 and Hyb_1 are perfectly indistinguishable.
In Lemma B.3 we show that Hyb_1 and Hyb_2 are perfectly indistinguishable.
In Lemma B.4 we show that Hyb_2 and Hyb_3 are perfectly indistinguishable.
In Lemma B.5 we show that Hyb_3 and Hyb_4 are statistically indistinguishable.
We note that Hyb_4 corresponds to the ideal world execution using the simulator.

Lemma B.2. $\mathsf{Hyb}_0 \equiv \mathsf{Hyb}_1$.

Proof. This can be verified by inspection. Note that the view of the adversary is composed of the outputs of all oracle calls, the inputs of corrupt parties and adversary's own private randomness. The description of the simulator has been changed to involve the real inputs of honest parties in steps 3(c), and 4. The use of simulator has been reverted back to the actual computation of conditional disclosure protocols. All the variables sampled by honest parties have been sampled

according to the same distribution. Finally, the output is computed according to the instructions in the protocol.

Lemma B.3. $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2$.

Proof. Hyb_1 differs from Hyb_2 only in the input of P_c, where $c \in H$, in $\mathrm{CD}_{\{a_b,a_\ell\},c}$ for each ℓ. In the former, the actual input x_i provided by the environment is used, whereas, in the latter, a dummy input 0^m chosen by Sim is used. The perfect indistinguishibility between these two hybrids can be shown exactly as in lemma A.5.

Lemma B.4. $\mathsf{Hyb}_2 \equiv \mathsf{Hyb}_3$.

Proof. For $a \in H$ and $c \in C$, the real world execution of $\mathrm{CD}_{\{a_b,a_\ell\},c}$ in Hyb_2 has been replaced with an ideal world execution in Hyb_3. This is the only difference between Hyb_2 and Hyb_3. The perfect indistinguishibility between these two hybrids can be shown exactly as in lemma A.6.

Lemma B.5. $\mathsf{Hyb}_3 \approx_{\epsilon(\lambda)} \mathsf{Hyb}_4$, *where* $\epsilon(\lambda)$ *is a negligible function.*

Proof. Consider a hybrid $\mathsf{Hyb}_{3,1}$ obtained by modifying the instructions for sampling $z_{\{i,j,k\}}$ in step 4 of Hyb_4 as follows: If there exists $\{i',j',k'\} \in \mathcal{T}$ such that $\{i',j',k'\} \not\subseteq C$, then sample $z_{\{i,j,k\}}$ uniformly at random. Otherwise, Choose $z_{\{i,j,k\}}$ such that

$$\sum_{\{\ell,\ell',\ell''\} \not\subseteq C} z_{\{\ell,\ell',\ell''\}} = h_{\{\ell,\ell',\ell''\}}(x_\ell^{\ell,\ell',\ell''}, x_{\ell'}^{\ell,\ell',\ell''}, x_{\ell''}^{\ell,\ell',\ell''}), \tag{23}$$

where $x_\ell^{\ell,\ell',\ell''}$ is the provided by the adversary while invoking $\mathcal{O}_{\{\ell,\ell,\ell''\}}$ if $\ell \in C$, and it is the input provided by the environment when $\ell \in H$; similarly for the other two cases.

The Hyb_3 and $\mathsf{Hyb}_{3,1}$ differ only in the manner in which $z_{\{i,j,k\}}$ is chosen while computing $\hat{z}_{\{i,j,k\}}$ for each $\{i,j,k\} \not\subseteq C$ in step 4. In the former, for each $\{i,j,k\} \not\subseteq C$, $z_{\{i,j,k\}} = h_{\{i,j,k\}}(x_i^{i,j,k}, x_j^{i,j,k}, x_k^{i,j,k})$. In the latter, $\{z_{\{\ell,\ell',\ell''\}}\}$ have been chosen as an additive secret sharing satisfying (23).

Claim B.5.1 *When I denotes the indicator function, for each $\{i,j,k\}$, define*

$$u_{\{i,j,k\}} = \sum_{\ell \in \{i,j,k\}} I(\ell \in H) \cdot \gamma_\ell[\{i,j,k\}].$$

Then, $\{u_{\{i,j,k\}}\}_{\{i,j,k\} \not\subseteq C}$ form an additive secret sharing of 0.

The proof of the claim follows the same outline as claim A.7.1, hence we leave it to the reader.

It can be verified that, for each $\{i,j,k\}$, $\gamma_\ell[\{i,j,k\}]$ for $\ell \in \{i,j,k\}$, and hence $u_{\{i,j,k\}}$ is used exclusively in step 4. The above claim implies that $\{z_{\{i,j,k\}} + u_{\{i,j,k\}}\}_{\{i,j,k\} \not\subseteq C}$ is an additive secret sharing of $\sum_{\{i,j,k\} \not\subseteq C} z_{\{i,j,k\}}$. This directly implies that $\mathsf{Hyb}_3 \equiv \mathsf{Hyb}_{3,1}$.

Next, we obtain $\mathsf{Hyb}_{3,2}$ by modifying $\mathsf{Hyb}_{3,1}$. Let $\mathcal{O}_{\{i,j,k\}}$ be the last 3-ary function that $\mathcal{A}$ queries such that $\{i,j,k\} \not\subseteq C$; i.e., there exists no $\{i',j',k'\} \in \mathcal{T}$ such that $\{i',j',k\} \not\subseteq C$. We modify the choice of $z_{\{i,j,k\}}$ in step 4 as follows: If for all $\ell \in C$, there exists $\hat{x}_\ell$ such that, for all $\{\ell,\ell',\ell''\} \not\subseteq C$, the adversary used $\hat{x}_\ell$ as input of P_ℓ while invoking $\mathcal{O}_{\{\ell,\ell',\ell''\}}$, the choose $z_{i,j,k}$ such that

$$\sum_{\{\ell,\ell',\ell''\} \not\subseteq C} z_{\{\ell,\ell',\ell''\}} = f(\hat{x}_1,\ldots,\hat{x}_n) - \sum_{\{\ell,\ell',\ell''\} \subseteq C} h_{\{\ell,\ell',\ell''\}}(\hat{x}_\ell,\hat{x}_{\ell'},\hat{x}_{\ell''}).$$

If not, make no changes to $\mathsf{Hyb}_{3,1}$. $\mathsf{Hyb}_{3,1} \equiv \mathsf{Hyb}_{3,2}$ by the above claim, since $\sum_{\{i,j,k\} \not\subseteq C} z_{\{i,j,k\}} = h_{\{i,j,k\} \not\subseteq}(\hat{x}_i,\hat{x}_j,\hat{x}_k)$.

Finally, we show that $\mathsf{Hyb}_{3,2} \approx_{\epsilon(\lambda)} \mathsf{Hyb}_4$, where $\epsilon(\lambda)$ is a negligible function. Let $\mathcal{O}_{\{i,j,k\}}$ be the last 3-ary function that $\mathcal{A}$ queries such that $\{i,j,k\} \not\subseteq C$. For all $\{\ell,\ell',\ell''\} \neq \{i,j,k\}$, $\mathsf{Hyb}_{3,2}$ and Hyb_4 behave identically. The only difference between the two hybrids is in the choice of $z_{\{i,j,k\}}$, when there exists $\ell \in C$, $\{\ell,\ell_1,\ell_2\} \not\subseteq C$ and $\{\ell,\ell_1',\ell_2'\} \not\subseteq C$ such that, the input provided by $\mathcal{A}$ on behalf of P_ℓ in $\mathcal{O}_{\{\ell,\ell_1,\ell_2\}}$ is distinct from that provided in $\mathcal{O}_{\{\ell,\ell_1',\ell_2'\}}$. We crucially observe that, whenever there exists $\ell,\ell_1,\ell_2,\ell_1',\ell_2'$ satisfying these conditions, there exists $a \in H$ such that the input provided by $\mathcal{A}$ to $\mathrm{CD}_{\{a_b,a_c\},\ell}.\mathcal{O}_{\{a_b,\ell\}}$ and $\mathrm{CD}_{\{a_b,a_c\},\ell}.\mathcal{O}_{\{a_c,\ell\}}$ on behalf of P_ℓ are inconsistent. In conclusion, the two hybrids differ only in the sampling of $z_{\{i,j,k\}}$ (where $\{i,j,k\}$ is as descibed above) when there exist $a \in H$ and $\ell \in C$ satisfying the above condition. At this point, we may use the same line of argument used in the proof of lemma A.7 to prove this lemma.

This concludes the proof of the lemma. □