On Secure Computation of Solitary Output Functionalities With and Without Broadcast

Bar Alon*
alonbar08@gmail.com

Eran Omri†
omrier@ariel.ac.il

July 18, 2022

Abstract

Secure multiparty computation (MPC) models scenarios, where a set of mutually distrusting parties wish to compute some task over their private inputs. Assuming that the majority of the parties are honest and that the parties have access to a broadcast channel, every function can be computed with *full security*. Conversely, if either an honest majority or a broadcast channel cannot be assumed (as is the case in various real-world settings), then there are functionalities that cannot be computed with full security. Understanding the exact power of each of these assumptions is a valuable goal.

In this paper, we study full security for solitary output functionalities (where only a single party receives an output). We focus on three-party functionalities in the point-to-point model (without broadcast), assuming an honest majority. We develop new techniques for analyzing the security of MPC protocols in the point-to-point model. Using these techniques, we are able to give a characterization for several interesting classes of solitary output three-party functionalities (including Boolean and ternary-output functionalities over a polynomial-size domain) that are computable with full security in the setting of an honest majority without a broadcast channel.

Furthermore, using our techniques, we make progress in understanding the set of solitary output three-party functionalities that can be computed with full security, assuming broadcast but no honest majority. Specifically, we extend the set of such functionalities that are known to be computable, due to Halevi et al. [TCC '19].

Keywords: broadcast; point-to-point communication; secure multiparty computation; solitary output; impossibility result.

*Department of Computer Science, Ariel University. Ariel Cyber Innovation Center (ACIC). Research supported in part by grants from the Israel Science Foundation (no.152/17), and by the Ariel Cyber Innovation Center in conjunction with the Israel National Cyber directorate in the Prime Minister's Office.

†Department of Computer Science, Ariel University. Ariel Cyber Innovation Center (ACIC). Research supported in part by grants from the Israel Science Foundation (no.152/17), by the Ariel Cyber Innovation Center in conjunction with the Israel National Cyber directorate in the Prime Minister's Office, and by the Robert L. McDevitt, K.S.G., K.C.H.S. and Catherine H. McDevitt L.C.H.S. endowment at Georgetown University. Part of this work was done when E.O. was hosted by Georgetown University.

Contents

1 Introduction

In today's digital world, mutually distrustful parties communicate over large networks, such as the Internet, and perform common tasks together, where each party holds some private information. Secure multiparty computation (MPC) addresses this scenario and offers protocols for performing tasks in untrusted environments while guaranteeing security. The two most basic security properties are correctness and privacy. However, in most scenarios participating parties may also desire other properties, such as, fairness (namely, either all parties receive an output or none do), and guaranteed output delivery (honest parties always receive an output).

In this work we focus on the notion of *full security*, which captures all of the above security properties (and several others).[1] For general functionalities, there are two main ingredients that are essential for achieving fully secure protocols for any (efficiently computable) functionality. These ingredients are the guarantee that a strict majority of the parties are honest and the availability of a broadcast channel (allowing any party to reliably send the same message to all parties). Indeed, assuming an honest majority and a broadcast channel (on top of a complete point-to-point network) there is an MPC protocol that computes any functionality with full security [7, 25].

Cleve [9] showed that without an honest majority full security cannot be achieved even for the simple coin-tossing functionality (even with a broadcast channel). On the other hand, even if two-thirds of the parties are honest, there is no fully secure protocol for computing the broadcast functionality in the plain model (i.e., without setup/proof-of-work assumptions) [24, 22, 15]. This raises the questions of identifying the set of functions that can be computed with full security assuming one of these two ingredients, but not the other.

For the setting with broadcast but without an honest majority, a characterization was given for the set of two-party, Boolean, symmetric (i.e., where all parties receive the same output) functions over a constant size domain [19, 3, 23, 4]. The cases of asymmetric functions and of multiparty functions were also investigated [18, 4, 13, 20, 12], but both characterizations are open. For the setting of no broadcast and an honest majority, [11] characterized the symmetric functions that can be computed with full security without broadcast. The investigation of this setting was extended to deal with asymmetric functions in [2], who provided a variety of necessary and sufficient conditions for full security in this setting.

In this paper, we investigate the above two questions for the special case of *solitary output* functionalities, i.e., where only a single (predetermined) party receives an output from the computation. Besides being an interesting special case of asymmetric functionalities, solitary output functionalities capture many real-world scenarios of MPC. One motivating example is that of a service provider, wishing to perform some private data analysis over the private inputs of its users, where no one but the service provider should learn any information about the inputs or the output. Solitary output functionalities are also important for the setting of non-interactive MPC such as the Private Simultaneous Messages (PSM) model [14].

This leads us to the main question studied in this paper:

Characterize the set of solitary output functionalities
that can be computed with full security.

[1] Formally, full security is defined via the so-called real vs. ideal paradigm, where a (real-world) protocol is required to emulate an ideal setting, in which the adversary is limited to selecting inputs for the corrupted parties and receiving their outputs.

For these functionalities, fairness in not an issue. However, even for the setting with a broadcast channel, Halevi et al. [20] showed that without an honest majority some solitary output functionalities cannot be computed with guaranteed output delivery. On the positive end, they present fully-secure protocols for several natural and useful families of solitary output functionalities, including some variants of the Private Set Intersection (PSI) problem. For the setting without a broadcast channel, only a handful of solitary output functionalities were known to be impossible to compute, even when an honest majority is present [2, 16]. On the positive side, [2] identified a class of solitary output functionalities that can be computed with full security.[2] However, to the best of our knowledge, no characterization was given for an interesting sub-class of functionalities (either with or without a broadcast channel).

1.1 Our Results

In this paper, we focus on solitary output three-party functionalities where party A with input x, party B with input y, and party C with input z, compute a functionality f with only A receiving the output. Furthermore, we mainly focus on functionalities with polynomial-sized domain.

We develop new techniques for analyzing the security of MPC protocols in the point-to-point model. Using these techniques, we are able to give a clean characterization for several interesting classes of solitary output three-party functionalities (including, Boolean and even ternary-output functionalities over a domain of polynomial size) that are computable with full security in the setting of an honest majority without a broadcast channel. We believe that the new techniques can prove useful in analyzing the security of protocols for a broader class of MPC settings. Indeed, using these techniques, we show that any solitary output three-party Boolean functionality that can be securely computed without broadcast assuming an honest majority, can be securely computed with broadcast and no honest majority (extending the set of functionalities that are known to be computable, due to Halevi et al. [20]).

For the sake of simplicity of the presentation, in the rest of this introduction we only consider perfect security and functionalities with finite domain and range. A formal statement of the results for functionalities with polynomial-sized domain and computational security is given in Section 3. We will also limit the following discussion to two families of functionalities, for which our results admit a characterization. The first family we consider is that of *deterministic no-input output-receiving party* (NIORP) functionalities, where the output-receiving party A has no input. The second family we consider is the set of (possibly randomized) ternary-output functionalities, where the output of A is one of three values (with A possibly holding an input). In particular, this yields a characterization for Boolean functionalities. Below are informal statements of the characterizations for *deterministic* functionalities. We handle randomized functionalities by a reduction to the deterministic case (see Section 1.2.3 below).

Functionalities with no input for the output-receiving party (NIORP). Before stating the theorem, we define a special partitioning of the inputs of B and C. The partition is derived from an equivalence relation, which we call ***c****ommon* ***o****utput* ***re****lation* (CORE), hence, we call the partition the *CORE partition*.

[2] The main positive result of [2] holds for functionalities where two parties receive an output, hence solitary output is a special case of their results. Additionally, they identified several other solitary output functionalities that fall outside of their main results, that can also be securely computed.

Definition 1.1 (CORE partition)**.** *Let $f : \emptyset \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ be a deterministic solitary output three-party NIORP functionality. For inputs $y, y' \in \mathcal{Y}$, we say that $y \sim y'$ if and only if there exists $z, z' \in \mathcal{Z}$ such that $f(y, z) = f(y', z')$. We define the equivalence relation $\equiv$ to be the* transitive closure *of $\sim$. That is, $y \equiv y'$ if and only if either $y \sim y'$ or there exists a sequence of inputs $y_1, \ldots, y_k \in \mathcal{Y}$ such that*

$$y \sim y_1 \sim \ldots \sim y_k \sim y'.$$

We partition the set of inputs $\mathcal{Y}$ according to the equivalence classes of $\equiv$, and we write the partition as $\mathscr{Y} = \{\mathcal{Y}_i : i \in [n]\}$. We partition $\mathcal{Z}$ into disjoint sets $\mathscr{Z} = \{\mathcal{Z}_j : j \in [m]\}$ similarly. We refer to these partitions as the CORE partitions *of f.*

As an example, consider the following solitary output three-party functionality given by the matrix

$$\begin{pmatrix} 0 & 1 & 2 \\ 1 & 3 & 4 \\ 3 & 4 & 5 \end{pmatrix}$$

Here B chooses a row, C chooses a column, and the output of A is the value written in the chosen entry. Then the CORE partition of both the rows and the columns result in the trivial partition, i.e., all rows are equivalent and all columns are equivalent. To see this, note that both the first and second row contain the output 1. Therefore they satisfy the relation $\sim$. Similarly, the second and last row satisfy $\sim$ since 3 (and 4) are a common output. Thus, the first and last row are equivalent (though they do not satisfy the relation $\sim$). Similarly, all columns are equivalent.

We are now ready to state our characterization for NIORP functionalities.

Theorem 1.2 (Characterization of NIORP functionalities, informal)**.** *Let $f : \emptyset \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ be a deterministic solitary output three-party NIORP functionality, and let $\mathscr{Y} = \{\mathcal{Y}_i : i \in [n]\}$ and $\mathscr{Z} = \{\mathcal{Z}_j : j \in [m]\}$ be the CORE partitions of $\mathcal{Y}$ and $\mathcal{Z}$, respectively. Then f can be securely computed in the point-to-point model, if and only if there exists two families of efficiently samplable distributions $\{Q_i\}_{i\in[n]}$ and $\{R_j\}_{j\in[m]}$, such that the following holds. For all $i \in [n]$, $j \in [m]$, $y \in \mathcal{Y}_i$, and $z \in \mathcal{Z}_j$, it holds that*

$$f(y^*, z) \equiv f(y, z^*),$$

where $y^* \leftarrow Q_i$ and $z^* \leftarrow R_j$.

Stated differently, consider the partition of $\mathcal{Y} \times \mathcal{Z}$ into combinatorial rectangles defined by $\mathscr{R} = \{\mathcal{Y}_i \times \mathcal{Z}_j : i \in [n], j \in [m]\}$. Then f can be securely computed if and only if both B and C can each associate a distribution to each set in the partition of their respective set of inputs, such that the output distribution in each combinatorial rectangle in $\mathscr{R}$ is fixed.

In Table 1, we illustrate the usefulness of Theorem 1.2 by considering various functionalities (which were also considered by [20]) related to private-set intersection (PSI), and mark whether each variant can be computed with full security. Define the NIORP functionality $\mathsf{PSI}^{\ell_1,\ell_2}_{k_1,k_2,m}$ to output to A the intersection of $\mathcal{S}_1$ and $\mathcal{S}_2$, held by B and C, respectively. Here, $\mathcal{S}_i \subseteq \{1, \ldots, m\}$ and $k_i \leq |\mathcal{S}_i| \leq \ell_i$ for every $i \in \{1, 2\}$. The variants we consider are those that apply some function g over the output of A, i.e., the functionality the parties compute is $g(\mathsf{PSI}^{\ell_1,\ell_2}_{k_1,k_2,m}(\mathcal{S}_1, \mathcal{S}_2))$. The proofs for which parameters allow each function to be computed are presented in Section 7. It is important to note that the domains of the functionalities are *constant* as otherwise some of the claims are provably false (e.g., [2] showed that $\mathsf{PSI}^{1,1}_{1,1,\kappa}$, where κ is the security parameter, can be securely computed).

Input restriction\Function g	$g(\mathcal{S}) = \mathcal{S}$	$g(\mathcal{S}) = \lvert\mathcal{S}\rvert$	$g(\mathcal{S}) = \begin{cases} 1 & \text{if } \mathcal{S} = \emptyset \\ 0 & \text{otherwise} \end{cases}$
$k_1 = k_2 = 0$, or $\ell_1 = 0$, or $\ell_2 = 0$, or $k_1 = m$, or $k_2 = m$	✓	✓	✓
$k_1 = \ell_1 \notin \{0, m\}$ and $k_2 = \ell_2 \notin \{0, m\}$	✗	✓	✓
$0 < k_1 < \ell_1$, $0 < k_2 < \ell_2$, and $\ell_1 + k_2,\ k_1 + \ell_2 > m$	✗	✗	✓
Any other choice	✗	✗	✗

Table 1: Summary of our results stated for various versions of the PSI functionality. Each row in the table above corresponds to a different choice of the parameters. Each column corresponds to a different function g applied to the output of A.

Ternary-output functionalities. We next give our characterization for ternary-output functionalities. In this setting, party A also has an input, and its output is a value in $\{0, 1, 2\}$. Similarly to the NIORP case, we consider partitions over the inputs of B and C. Here, however, each input $x \in \mathcal{X}$ is associated with a different CORE partition. For the characterization, we consider the *meet* of all such partitions, which, intuitively, is the partition of the set using all partitions together. Formally, for partitions $\mathcal{P}_1, \ldots, \mathcal{P}_n$ over a set $\mathcal{S}$, their meet is defined as the collection of all non-empty intersections, i.e.,

$$\bigwedge_{i=1}^{n} \mathcal{P}_i := \left\{ \mathcal{T} \subseteq \mathcal{S} : \mathcal{T} \neq \emptyset, \exists \mathcal{T}_1 \in \mathcal{P}_1, \ldots, \mathcal{T}_n \in \mathcal{P}_n \text{ s.t. } \mathcal{T} = \bigcap_{i=1}^{n} \mathcal{T}_i \right\}.$$

Before stating the theorem, we formalize the meet of the CORE partitions, which we call $\text{CORE}_\wedge$-partition, for a given solitary output functionality.

Definition 1.3 ($\text{CORE}_\wedge$-partition)**.** *Let $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \{0, 1, 2\}$ be a deterministic solitary output three-party ternary-output functionality. For every $x \in \mathcal{X}$, we can view $f(x, \cdot, \cdot)$ as a NIORP functionality, and consider the same CORE partition as in Definition 1.1. We denote these partitions by $\mathscr{Y}_x = \{\mathcal{Y}_i^x : i \in [n(x)]\}$ and $\mathscr{Z}_x = \{\mathcal{Z}_j^x : j \in [m(x)]\}$. We define the* $\text{CORE}_\wedge$-partitions *of f as the meet of its CORE partitions, that is, we let $\mathscr{Y}_\wedge = \bigwedge_{x \in \mathcal{X}} \mathscr{Y}_x$ and $\mathscr{Z}_\wedge = \bigwedge_{x \in \mathcal{X}} \mathscr{Z}_x$. We denote their sizes as $n_\wedge = |\mathscr{Y}_\wedge|$ and $m_\wedge = |\mathscr{Z}_\wedge|$, and we write them as $\mathscr{Y}_\wedge = \{\mathcal{Y}_i^\wedge : i \in [n_\wedge]\}$ and $\mathscr{Z}_\wedge = \{\mathcal{Z}_j^\wedge : j \in [m_\wedge]\}$.*

As an example, consider the deterministic variant of the convergecast functionality [16], CC :

$(\{0,1\})^3 \mapsto \{0,1\}$ defined as[3]

$$\mathsf{CC}(x,y,z) = \begin{cases} y & \text{if } x = 0 \\ z & \text{otherwise} \end{cases}$$

Equivalently, CC can be defined by the two matrices

$$M_0 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} \quad \text{and} \quad M_1 = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$$

Here, A chooses a matrix, B chooses a row, and C chooses a column. The output of A is the value written in the chosen entry. Observe that in M_0, the rows are not equivalent while the columns are. In M_1, however, the converse holds, namely, the row are equivalent while the columns are not. Thus, in the $\mathrm{CORE}_\wedge$-partitions of CC any two inputs are in different sets.

We are now ready to state our characterization for ternary-output functionalities.

Theorem 1.4 (Characterization of ternary-output functionalities, informal)**.** *Let $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \{0,1,2\}$ be a deterministic solitary output three-party ternary-output functionality, and let $\mathfrak{Y}_\wedge = \{\mathcal{Y}_i^\wedge : i \in [n_\wedge]\}$ and $\mathfrak{Z}_\wedge = \{\mathcal{Z}_j^\wedge : j \in [m_\wedge]\}$ be its $\mathrm{CORE}_\wedge$-partition. Then f can be securely computed in the point-to-point model, if and only if the following hold.*

1. *Either $\mathfrak{Y}_x = \{\mathcal{Y}\}$ for all $x \in \mathcal{X}$, or $\mathfrak{Z}_x = \{\mathcal{Z}\}$ for all $x \in \mathcal{X}$.*
2. *There exists an algorithm S, and there exists three families of efficiently samplable distributions $\{P_x\}_{x\in\mathcal{X}}$, $\{Q_i\}_{i\in[n_\wedge]}$, and $\{R_j\}_{j\in[m_\wedge]}$, such that the following holds. For all $i \in [n_\wedge]$, $j \in [m_\wedge]$, $y \in \mathcal{Y}_i^\wedge$, $z \in \mathcal{Z}_j^\wedge$, and $x \in \mathcal{X}$, it holds that*

$$\mathsf{S}(x, x^*, f(x^*, y, z)) \equiv f(x, y^*, z) \equiv f(x, y, z^*),$$

where $x^* \leftarrow P_x$, where $y^* \leftarrow Q_i$, and where $z^* \leftarrow R_j$.

Observe that the deterministic convergecast functionality CC does *not* satisfy Item 1 since $\mathfrak{Y}_0 \neq \{\mathcal{Y}\}$ and $\mathfrak{Z}_1 \neq \{\mathcal{Z}\}$. Therefore it cannot be securely computed. We stress that the positive direction of Theorem 1.4 holds even for functionalities that are not ternary-output. At first sight, it might seems that the algorithm S is the same as a simulator for a corrupt A. However, we stress that we only require S to output what would become the output of an honest A, and not the entire view of the adversary. Arguably, determining whether such an algorithm exists is much simpler than determining when a simulator for some protocol exists.

Randomized functionalities. So far, we have only dealt with deterministic functionalities. To handle the randomized case, we show how to reduce it to the deterministic case. That is, we show that for any randomized solitary output three-party functionality f, there exists a deterministic solitary output three-party functionality f', such f can be securely computed if and only if f' can be securely computed.

[3] Fitzi et al. [16] defined the convergecast functionality as the NIORP randomized solitary output functionality, where A outputs y with probability $1/2$, and outputs z with probability $1/2$.

Proposition 1.5 (Reducing randomized functionalities to deterministic functionalities, informal)**.** *Let $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ be a (randomized) solitary output three-party functionality, and let $\mathcal{R}$ denote the domain of its randomness. Define the deterministic solitary output three-party functionality $f' : (\mathcal{X} \times \mathcal{R}) \times (\mathcal{Y} \times \mathcal{R}) \times (\mathcal{Z} \times \mathcal{R}) \mapsto \mathcal{W}$ as*

$$f'((x, r_1), (y, r_2), (z, r_3)) = f(x, y, z; r_1 + r_2 + r_3),$$

where addition is done over $\mathcal{R}$ when viewed as an additive group. That is, the parties receive a share of the randomness in a 3-out-of-3 secret sharing scheme. Then f can be securely computed if and only if f' can be securely computed.

Assuming a broadcast channel. Surprisingly, we are also able to show that any (randomized) solitary output three-party functionality that can be securely computed, as captured by Theorems 1.2 and 1.4, can also be securely computed assuming the availability of a broadcast channel with security holding against two corrupted parties. In particular, any solitary output three-party Boolean functionality that can be securely computed without broadcast, assuming an honest majority, can be securely computed with broadcast and no honest majority. Moreover, the set of functions captured by Theorems 1.2 and 1.4 extends the set of functions previously known to be computable with broadcast, due to Halevi et al. [20] (see [20, Theorem 4.4]).

On the other hand, we claim that the converse is false. Indeed, consider the following solitary output three-party variant of the soGHKL functionality, defined by the matrix

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$$

where B chooses a row, C chooses a column, and the output of A is the value written in the chosen entry.

Since soGHKL is a NIORP functionality, by Theorem 1.2 it cannot be securely computed in the point-to-point model. On the other hand, Halevi et al. [20] showed that soGHKL can be computed assuming a broadcast channel.[4] As a result, for NIORP and ternary-output solitary output three-party functionalities, assuming a broadcast channel is *strictly stronger* than assuming an honest majority. This is summarized below. We stress that the construction requires to assume the existence of oblivious transfer, hence the resulting protocol admits computational security.

Theorem 1.6 (Informal)**.** *Let $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ be a deterministic solitary output three-party functionality. Assume that f is either a NIORP or a ternary-output functionality. Suppose that f can be securely computed assuming an honest majority in the point-to-point model. Then, assuming the existence of oblivious transfer and the availability of a broadcast channel, f can be computed with computational security tolerating two corruptions.*

Moreover, the converse is false. That is, there exists a NIORP Boolean functionality that can be securely computed assuming a broadcast channel and no honest majority, but cannot be securely computed in the point-to-point model assuming an honest majority.

In fact, Theorem 1.6 can be improved by slightly relaxing some of the conditions the function has to satisfy (see Section 1.2.4 below for more details). Furthermore, Theorem 1.6 captures

[4]In fact, [20] gave three different protocols for computing soGHKL securely.

NIORP functionalities whose status was previously unknown, e.g., the NIORP functionality $f_{\mathsf{special}} : \emptyset \times (\{0,1,2,3\})^2 \mapsto \{0,\ldots,7\}$ defined by the matrix

$$\begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 2 \\ 4 & 5 & 6 & 7 \\ 5 & 4 & 7 & 6 \end{pmatrix}$$

can be securely computed assuming a broadcast channel, tolerating two corruptions.

In Table 2 below, we present several examples of three-party functionalities, and compare their status assuming no broadcast channel and one corruption, to the case where such a channel is available with two possible corruptions.

Function\Model	Without broadcast (honest majority)	With broadcast (no honest majority)
Millionaires' Problem: $\mathsf{GT}(x,y,z) = \begin{cases} 0 & \text{if } x > y,z \\ 1 & \text{if } y > z, y \geq x \\ 2 & \text{otherwise} \end{cases}$	✗ Thm. 1.4	✓ [20]
NIORP Millionaires' Problem: $\mathsf{cGT}(y,z) = \begin{cases} 0 & \text{if } y > z \\ 1 & \text{otherwise} \end{cases}$	✓ Thm. 1.4	✓ [20]
$\mathsf{CC}(x,y,z)$	✗ Thm. 1.4	✓ [20]
$\mathsf{soGHKL}(y,z)$	✗ Thm. 1.4	✓ [20]
$\mathsf{Max}(x,y,z)$	✓ Thm. 1.4	✓ [20]
$\mathsf{EQ}(y,z) = \begin{cases} 1 & \text{if } y = z \\ 0 & \text{otherwise} \end{cases}$	✓ Thm. 1.4	✓ [20]
$\ell\mathsf{EQ}(y,z) = \begin{cases} y & \text{if } y = z \\ 0 & \text{otherwise} \end{cases}$	✗ Thm. 1.4	✗ [20]
$f_{\mathsf{special}} : \emptyset \times (\{0,1,2,3\})^2 \mapsto \{0,\ldots,7\}$	✓ Thm. 1.4	✓ Thm. 1.4

Table 2: Comparing the landscape of functionalities that can be computed without broadcast but with an honest majority, to functionalities that can be computed with broadcast but no honest majority. All functions above have a constant domain. It is important that the domain of $\ell\mathsf{EQ}$ does not include 0.

1.2 Our Techniques

We now turn to describe our techniques. In Section 1.2.1 we handle NIORP functionalities. Then, in Section 1.2.2 we handle ternary-output functionalities. Then, in Section 1.2.3 we show how to reduce the randomized case to the deterministic case. Finally, in Section 1.2.4 we prove Theorem 1.6, showing that for the families of functions considered, the broadcast assumption is strictly stronger than the honest majority assumption.

1.2.1 Characterizing NIORP Functionalities

We start with the negative direction of Theorem 1.2. Our argument is split into two parts. In the first part, we adapt the hexagon argument due to Fischer et al. [15] to the MPC setting. This results in 6 distributions, all of which are identically distributed. The second part of the proof is dedicated to the analysis of these 6 distributions.

The hexagon argument for NIORP functionalities. In the following, let f be a solitary output three-party NIORP functionality (no input for the output receiving party), and let π be a protocol computing f securely over point-to-point channels, tolerating a single corrupted party. At a high level, the hexagon argument is as follows. Given the three-party protocol π, we construct a new six-party protocol π'. Then, we consider six different *semi-honest* adversaries for π', and observe that each of them can be emulated by a *malicious* adversary in the original three-party protocol π. By the security of π, each of the malicious adversaries can simulated in the ideal world of f. Finally, since all adversaries for six-party protocol π' we consider are semi-honest, the view of each party is the identically distributed across all six scenarios. We then conclude that the output of all six simulators must be identically distributed. We stress that π' is *not* secure, but rather any attacker for it can be emulated by an attacker for the three-party protocol π.

We next provide a more formal argument. Consider the following six-party protocol. We have two copies of each party, all of which are acting honestly, i.e., each copy of party P is acting the same as an honest P does in π. Furthermore, the parties are connected via a cycle graph as depicted in Figure 1. Finally, we let B, B', C, and C' hold inputs y, y', z, and z', respectively.

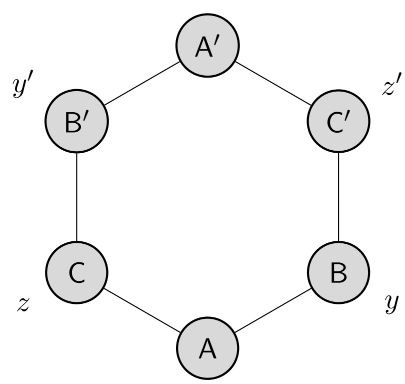

Figure 1: The six-party protocol

Now, consider the following 6 attack-scenarios for the six-party protocol, where in each scenario a *semi-honest* adversary corrupts four adjacent parties, as depicted in Figure 2. Observe that each attacker can be emulated in the original three-party protocol π, by a *malicious* adversary emulating the corresponding four parties in its head. For example, in Scenario 2a, an adversary in π can emulate the attack by corrupting C, and emulating in its head two virtual copies of C, a copy of A and a copy B.

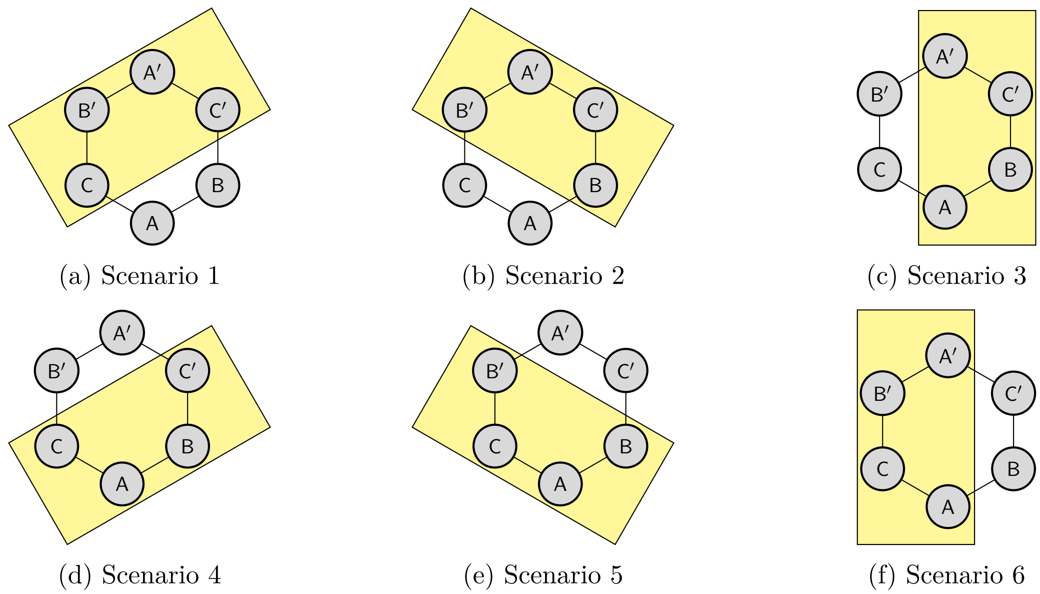

Figure 2: The six adversaries in the hexagon argument. The shaded yellow areas in each scenario correspond to the (virtual) parties the adversary controls.

We now focus on the output of A in an honest execution of the six-party protocol. Since in all six attack-scenarios all parties are acting honestly, it follows that the view of party A is identically distributed in all cases. In particular, its output is identically distributed, whether it is the output of an honest A (i.e., in scenarios 1 and 2), or it is part of the view of the adversary. By the assumed perfect security of π, each of the adversaries can be simulated in the corresponding ideal world of the three-party functionality f. Thus, we obtain six different expressions for the output of A, described as follows.

Scenarios 1 and 2: For each of the two scenarios, the simulator (of the three-party protocol) defines a distribution over the input it sends to the trusted party. Therefore, the output of A in scenario 1 must be distributed like $f(y, z^*)$, where z^* is sampled according to an efficiently samplable distribution $R_{y',z,z'}$ that depends only on y', z, and z'. Similarly, the output of A in scenario 2 must be distributed like $f(y^*, z)$, where y^* is sampled according to an efficiently samplable distribution $Q_{y,y',z'}$ that depends only on y, y', and z'.

Scenarios 4 and 5: In these two scenarios, A is being corrupted, and thus its output can be generated by a simulator corrupting C in scenario 4, and a simulator corrupting B in scenario 5. Since B and C have no output, it follows that there exists two efficient algorithms S_B and S_C, such that $\mathsf{S}_\mathsf{C}(y, z, z')$ and $\mathsf{S}_\mathsf{B}(y, y', z)$ are both identical to the output of the adversary in the real world.

Scenarios 3 and 6: Similarly to the previous case, the output of A in the real world can be generated by a simulator corrupting A in the ideal world of f. Since A receives an output from the trusted party, it follows that there exists two efficient algorithms S_3 and S_6 such that $\mathsf{S}_3(y, z', f(y', z))$ and $\mathsf{S}_6(y', z, f(y, z'))$ are both identical to the output of A in the real world.

To summarize, for all $y, y' \in \mathcal{Y}$ and $z, z' \in \mathcal{Z}$, we have that there exist two efficiently samplable distributions $Q_{y,y',z'}$ and $R_{y',z,z'}$ over $\mathcal{Y}$ and $\mathcal{Z}$, respectively, and four efficient algorithms S_B, S_C, S_3, and S_6, such that the following holds.

$$f\left(y^*, z\right) \equiv f\left(y, z^*\right) \equiv \mathsf{S}_\mathsf{B}\left(y, y', z\right) \equiv \mathsf{S}_\mathsf{C}\left(y, z, z'\right) \equiv \mathsf{S}_3\left(y, z', f\left(y', z\right)\right) \equiv \mathsf{S}_6\left(y', z, f\left(y, z'\right)\right), \quad (1)$$

where $y^* \leftarrow Q_{y,y',z'}$ and where $z^* \leftarrow R_{y',z,z'}$.

Analyzing the six distributions over the output of A. We now turn to the analysis of Equation (1), which results in the necessary conditions stated in Theorem 1.2. First, observe that since S_B is independent of z', it follows that all other distributions are also independent of it. Stated differently, changing z' to another value does not change the distributions. Similarly, since S_C is independent of y' it follows that all other distributions are also independent of it as well. Let y_0 and z_0 be the lexicographically smallest elements of $\mathcal{Y}$ an $\mathcal{Z}$, respectively, and define $Q'_y := Q_{y,y_0,z_0}$ and $R'_z := R_{y_0,z,z_0}$. Therefore,

$$f\left(y^*, z\right) \equiv f\left(y, z^*\right) \equiv \mathsf{S}_3\left(y, z', f\left(y', z\right)\right) \equiv \mathsf{S}_6\left(y', z, f\left(y, z'\right)\right), \quad (2)$$

for all $y' \in \mathcal{Y}$ and $z' \in \mathcal{Z}$, where $y^* \leftarrow Q'_y$ and $z^* \leftarrow R'_z$.

Let us focus on S_3. Recall that the relation $\sim$ is defined as $z \sim \tilde{z}$ if and only if there exists $\tilde{y}, \tilde{y}' \in \mathcal{Y}$ such that $f(\tilde{y}, z) = f(\tilde{y}', \tilde{z})$. Since S_3 is independent of y', it follows that

$$\mathsf{S}_3\left(y, z', f\left(y', z\right)\right) \equiv \mathsf{S}_3\left(y, z', f\left(\tilde{y}, z\right)\right) \equiv \mathsf{S}_3\left(y, z', f\left(\tilde{y}', \tilde{z}\right)\right) \equiv \mathsf{S}_3\left(y, z', f\left(y', \tilde{z}\right)\right),$$

where the first and last transition follow from the observation that the output distribution of S_3 is independent of the value of y', and the second transition follows from the fact that S_3 receives the same inputs in both cases. Therefore, changing z to $\tilde{z}$ where $z \sim \tilde{z}$ does not change the distribution. By transitivity of $\equiv$, it follows that changing z to any $\tilde{z}'$ in the same set $\mathcal{Z}_j \in \mathbb{Z}$, does not change the distribution. Thus, all distributions in Equation (2) are not affected by such change.

As a result, for every $j \in [m]$, every $y \in \mathcal{Y}$, and every $z, \tilde{z}' \in \mathcal{Z}_j$, it holds that

$$f(y^*, z) \equiv f(y^*, \tilde{z}'),$$

where $y^* \leftarrow Q'_y$. Similarly, by focusing on S_6, it follows that $i \in [n]$, every $y, \tilde{y}' \in \mathcal{Y}_i$, and every $z \in \mathcal{Z}$, it holds that

$$f(y, z^*) \equiv f(\tilde{y}', z^*),$$

where $z^* \leftarrow R'_z$.

This implies that we can define the distribution Q'_y and R'_z using only the sets from the partitions containing y and z, respectively. Indeed, for any $i \in [n]$ let $Q''_i := Q'_{y_i}$, where y_i is the lexicographically smallest element in $\mathcal{Y}_i$. Similarly, for any $j \in [m]$ let $R''_j := R'_{z_j}$, where z_j is the lexicographically smallest element in $\mathcal{Z}_j$. Therefore, by Equation (2) it follows that for every $i \in [n]$, $j \in [m]$, $y \in \mathcal{Y}_i$, and $z \in \mathcal{Z}_j$ it holds that

$$f(y^*, z) \equiv f(y, z^*),$$

where $y^* \leftarrow Q''_i$ and $z^* \leftarrow R''_j$, as claimed.

The positive direction for NIORP functionalities. We now present a protocol for any solitary output three-party NIORP functionality f, satisfying the conditions stated in Theorem 1.2. Our starting point is the same as that of [11, 2], namely, computing f *fairly* (i.e., either all parties obtain the output or non do). This follows from the fact that, by the honest-majority assumption, the protocol of Rabin and Ben-Or [25] computes f assuming a *broadcast channel*; hence by [10] it follows that f can be computed with fairness over a point-to-point network.

We now describe the protocol. The parties start by computing f with fairness. If they receive outputs, then they can terminate, and output what they received.[5] If the protocol aborts, then B finds the unique $i \in [n]$ such that $y \in \mathcal{Y}_i$ and sends i to A. Similarly, C finds the unique $j \in [m]$ such that $z \in \mathcal{Z}_j$ and sends j to A. Observe that this can be done efficiently since the domain of f is of constant size. Party A then samples $y^* \leftarrow Q_i$ and outputs $f(y^*, z_j)$, where z_j is the lexicographically smallest element in $\mathcal{Z}_j$.

Observe that correctness holds since when all parties are honest, the fair protocol will never abort (note that without the fair computation of f the above protocol is not correct since A would always output $f(y^*, z_j)$ instead of $f(y, z)$). Now, consider a corrupt B (the case of a corrupt C is similar). First, note that the adversary does not obtain any information from the fair computation of f. Next, if the adversary sends some i' to A, then the simulator sends $y^* \leftarrow Q_{i'}$ to the trusted party. Then the output of A in the ideal world is $f(y^*, z)$. By our assumption on f this is identical to $f(y^*, z_j)$, which is the output of A in the real world.

Next, consider a corrupt A. Since it does not obtain any information from the (failed) fair computation of f, it suffices to show how a simulator that is given $f(y, z)$ can compute the corresponding i and j. Observe that by our definition for the partition of the inputs, any two distinct combinatorial rectangles $\mathcal{Y}_i \times \mathcal{Z}_j$ and $\mathcal{Y}_{i'} \times \mathcal{Z}_{j'}$, where $(i, j) \neq (i', j')$, have no common output. Indeed, if $f(y, z) = f(y', z')$, where $(y, z) \in \mathcal{Y}_i \times \mathcal{Z}_j$ and $(y', z') \in \mathcal{Y}_{i'} \times \mathcal{Z}_{j'}$, then $y \sim y'$ and $z \sim z'$, hence they belong to the same sets. Therefore, the simulator for the corrupt A can compute the corresponding i and j given the output by simply looking them up (which can be done efficiently since the domain is of constant size).

1.2.2 Characterizing Ternary-Output Functionalities

We now explain our techniques for proving Theorem 1.4. We begin with the negative direction. Similarly to the proof of Theorem 1.2, the argument is comprised of the hexagon argument and the analysis of the six distributions that are obtained. However, since A has an input, the argument is much more involved.

A generalized hexagon argument. Unlike in the previous proof, here the hexagon argument (as used there) does not suffice. To show where the argument falls short, let us first describe the six distributions obtained from the hexagon argument. In this setting, where A now has an input, the six-party protocol described earlier will now have A and A' hold inputs x and x', respectively. The two inputs are then given to the correct adversaries from the six scenarios. Furthermore, observe that S_3 and S_6, which corrupt A in the ideal world, can also send to the trusted party an input x_3^* and x_6^*, respectively, each sampled according to a distribution that depends on the simulator's

[5] Although B and C are suppose to receive no output from f, in a fair computation they either receive the empty string indicating that A received its output, or a special symbol $\perp$ indicating abort.

inputs. Thus, to adjust the hexagon argument to this scenario, Equation (1) should now be replaced with

$$\begin{aligned} f\left(x, y^*, z\right) &\equiv f\left(x, y, z^*\right) \\ &\equiv \mathsf{S}_{\mathsf{B}}\left(x, y, y', z\right) \\ &\equiv \mathsf{S}_{\mathsf{C}}\left(x, y, z, z'\right) \\ &\equiv \mathsf{S}_3\left(x, x', y, z', x_3^*, f\left(x_3^*, y', z\right)\right) \\ &\equiv \mathsf{S}_6\left(x, x', y', z, x_6^*, f\left(x_6^*, y, z'\right)\right), \end{aligned} \tag{3}$$

where $y^* \leftarrow Q_{x',y,y',z'}$, where $z^* \leftarrow R_{x',y',z,z'}$, where $x_3^* \leftarrow P^3_{x,x',y,z'}$, and where $x_6^* \leftarrow P^6_{x,x',y',z}$.

Now, recall that we defined the deterministic variant of the convergecast functionality [16], $\mathsf{CC} : (\{0,1\})^3 \mapsto \{0,1\}$ as

$$\mathsf{CC}(x, y, z) = \begin{cases} y & \text{if } x = 0 \\ z & \text{otherwise} \end{cases}$$

We observe that there exist distributions and algorithms satisfying Equation (3). Indeed, take $Q_{x',y,y',z'}$ to always output y, take $R_{x',y',z,z'}$ to always output z, and define S_{B} and S_{C} compute $\mathsf{CC}(x, y, z)$. Then the first four distributions always output $\mathsf{CC}(x, y, z)$. Observe that for $P^3_{x,x',y,z'}$ and $P^6_{x,x',y',z}$ that always output 1 and 0, respectively, it holds that S_3 and S_6 receive z and y from the trusted party, respectively. Therefore, the two algorithms can also compute $\mathsf{CC}(x, y, z)$.

However, as we will see below, the functionality CC cannot be computed securely in our setting. Intuitively, this is because the adversary corrupting A as in Scenario 2c, learns both y' and z. However, in the ideal world, a simulator can only learn one of them. To generalize this intuition, we consider the joint distribution of the outputs of A and A' in the six-party protocol, rather than only the distribution of the output of A. Doing a similar analysis results in the existence of six efficiently samplable distributions $P^3_{x,x',y,z'}$, $P^6_{x,x',y',z}$, $Q_{x,y,y',z}$, $Q'_{x',y,y',z'}$, $R_{x,y,z,z'}$, and $R'_{x',y',z,z'}$, and the existence of six efficient algorithms S_3, S_6, S_{B}, S'_{B}, S_{C}, and S'_{C}, where S_3 and S_6 output two values (corresponding to the outputs of A and A'), such that the following six distributions are identically distributed:

1. $\mathsf{S}_3(x, x', y, z', x_3^*, f(x_3^*, y', z))$, where $x_3^* \leftarrow P^3_{x,x',y,z'}$.
2. $\mathsf{S}_6(x, x', y', z, x_6^*, f(x_6^*, y, z'))$, where $x_6^* \leftarrow P^6_{x,x',y',z}$.
3. $(\mathsf{S}_{\mathsf{B}}(x, y, y', z, y_1^*), f(x', y_1^*, z'))$, where $y_1^* \leftarrow Q_{x,y,y',z}$.
4. $(f(x, y_2^*, z), \mathsf{S}'_{\mathsf{B}}\left(x', y, y', z', y_2^*\right))$, where $y_2^* \leftarrow Q'_{x',y,y',z'}$.
5. $(\mathsf{S}_{\mathsf{C}}(x, y, z, z', z_1^*), f(x', y', z_1^*))$, where $z_1^* \leftarrow R_{x,y,z,z'}$.
6. $(f(x, y, z_2^*), \mathsf{S}'_{\mathsf{C}}\left(x', y', z, z', z_2^*\right))$, where $z_2^* \leftarrow R'_{x',y',z,z'}$.

Observe that for CC the above distributions and algorithms do not exist. Indeed, for $x = 1$ and $x' = 0$, it holds that $\mathsf{CC}(x', y', z_1^*) = y'$ and $\mathsf{CC}(x, y_2^*, z) = z$. However, S_3 is given only one of y' or z, depending on the value of x_3^*, hence it cannot always output both of them correctly.

Analyzing the six joint distributions over the outputs of A and A'. We now analyze the new six distribution described earlier. First, similarly to the case of NIORP functionalities, the marginal distribution of the first entry is independent of x', y', and z', and the marginal distribution of the second entry is independent of x, y, and z. Let us focus on S_3 and the distribution $P^3_{x,x',y,z'}$.

Recall that by the distribution in Item 4 above, it holds that S_3 must be able to output the value $f(x, y^*_2, z)$. Further recall that we partitioned $\mathcal{Z}$ with respect to any $x \in \mathcal{X}$, and we denote the partition as $\mathfrak{Z}_x = \{\mathcal{Z}^x_j : j \in [m(x)]\}$. Now, observe that from $f(x, y^*_2, z)$ it is possible to infer the (unique) $j \in [m(x)]$ satisfying $z \in \mathcal{Z}^x_j$. However, the only information that S_3 can obtain on j is from the output $f(x^*_3, y', z)$ it receives from the trusted party, from which S_3 can infer the set containing z with respect to the partition of x^*_3. Thus, x^*_3 must be such that $\mathcal{Z}^{x^*_3}_{j^*_3} \subseteq \mathcal{Z}^x_j$, where $j^*_3 \in [m(x^*_3)]$ is such that $z \in \mathcal{Z}^{x^*_3}_{j^*_3}$. Since the distribution $P^3_{x,x',y,z'}$, from which x^*_3 is drawn from, is independent of z, this must hold for all z. In other words, the partition $\mathfrak{Z}_{x^*_3}$ must be a *refinement* of $\mathfrak{Z}_x$. Similarly, since S_3 must also output $f(x', y', z^*_1)$, it follows that $\mathfrak{Y}_{x^*_3}$ is a refinement of $\mathfrak{Y}_{x'}$. As a result, for any x and x' such x^*_3 exists.

Now, since we assume f to be ternary-output, for every x it holds that either $\mathfrak{Y}_x = \{\mathcal{Y}\}$ or $\mathfrak{Z}_x = \{\mathcal{Z}\}$. However, to prove Item 1 of Theorem 1.4, we need to show a stronger statement, reversing the order of quantifiers. That is, we need to show that $\mathfrak{Y}_x = \{\mathcal{Y}\}$ for all x, or $\mathfrak{Z}_x = \{\mathcal{Z}\}$ for all x. Assuming otherwise, there exists x and x' such that $\mathfrak{Y}_x \neq \{\mathcal{Y}\}$ and $\mathfrak{Z}_{x'} \neq \{\mathcal{Z}\}$. Then, as argued above, there exists x^* such that $\mathfrak{Y}_{x^*}$ refines $\mathfrak{Y}_x$ and $\mathfrak{Z}_{x^*}$ refines $\mathfrak{Z}_{x'}$. However, this implies that $\mathfrak{Y}_{x^*} \neq \{\mathcal{Y}\}$ and $\mathfrak{Z}_{x^*} \neq \{\mathcal{Z}\}$, which is impossible for ternary-output functions.

We now prove Item 2 of Theorem 1.4. From here on, we will only focus on the first (i.e., left) entry in each of the above 6 distributions. The proof follows similar ideas to that of the NIORP case. That is, we show that changing z to any $\tilde{z}$ that belongs to the same set $\mathcal{Z}^\wedge_j \in \mathfrak{Z}_\wedge$, does not change the distribution. To see this, first observe that if $z, \tilde{z} \in \mathcal{Z}^\wedge_j$ for some $j \in [m_\wedge]$, then for any x there exists $j_x \in [m(x)]$ such that $z, \tilde{z} \in \mathcal{Z}^x_{j_x}$. Then (focusing on the first entry in the output of S_3), a similar analysis to the NIORP case shows that

$$\mathsf{S}_3(x, x', y, z', x^*_3, f(x^*_3, y', z))[0] \equiv \mathsf{S}_3(x, x', y, z', x^*_3, f(x^*_3, y', \tilde{z}))[0],$$

for any *fixed* $x^*_3 \in \mathcal{X}$ such that $\mathfrak{Z}_{x^*_3}$ refines $\mathfrak{Z}_x$. As the support of $P^3_{x,x',y,z'}$ is exactly those x^*_3 where $\mathfrak{Z}_{x^*_3}$ refines $\mathfrak{Z}_x$, it follows that

$$\mathsf{S}_3(x, x', y, z', x^*_3, f(x^*_3, y', z))[0] \equiv \mathsf{S}_3(x, x', y, z', x^*_3, f(x^*_3, y', \tilde{z}))[0],$$

where $x^*_3 \leftarrow P^3_{x,x',y,z'}$.

In the following we let x_0, y_0, and z_0 be the lexicographically smallest elements of $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$, respectively. For $i \in [n_\wedge]$ let $Q''_i := Q_{x_0,y_i,y_0,z_0}$, where y_i is the lexicographically smallest elements of $\mathcal{Y}^\wedge_i$. Similarly, for $j \in [m_\wedge]$ we let $R''_j := R'_{x_0,y_0,z_j,z_0}$, where z_j is the lexicographically smallest element of $\mathcal{Z}^\wedge_j$.

The algorithm S is defined to be either the first entry in the output of S_3, or the first entry in the output of S_6, depending on whether $\mathfrak{Y}_x = \{\mathcal{Y}\}$ for all $x \in \mathcal{X}$, or $\mathfrak{Z}_x = \{\mathcal{Z}\}$ for all $x \in \mathcal{X}$. Assume without loss of generality that the former holds. In this case we let $\mathsf{S}(x, x^*, w) = \mathsf{S}_3(x, x_0, y_0, z_0, x^*, w)[0]$. Then, for $P_x := P^3_{x,x_0,y_0,z_0}$ it holds that

$$\mathsf{S}\left(x, x^*, f(x^*, y, z)\right) \equiv \mathsf{S}_3\left(x, x_0, y_0, z_0, x^*, f(x^*, y, z)\right)[0] \equiv f(x, y^*, z) \equiv f(x, y, z^*),$$

where $x^* \leftarrow P_x$, $y^* \leftarrow Q''_1$ (recall we assume that $\mathfrak{Y}_x = \{\mathcal{Y}\}$ for all x which implies that $n_\wedge = 1$), and $z^* \leftarrow R''_j$.

The positive direction for ternary-output functionalities. We turn to the positive direction. Here we show that the protocol suggested by [2] securely computes f. Roughly speaking, in their protocol, in case an attack is detected (without the identity of the attacker being revealed) party A interacts either B or C while ignoring the other party, where the decision is based only on the function being computed (this is done even if the ignored party is honest).[6]

However, in [2], determining which party should interact with A (given the function f) is rather difficult. In contrast, as we show below, in our setting this is only determined by Item 1. Specifically, if $\mathcal{Y}_x = \{\mathcal{Y}\}$ for all $x \in \mathcal{X}$ then A interacts with C, and if $\mathcal{Z}_x = \{\mathcal{Z}\}$ for all $x \in \mathcal{X}$ then A interacts with B. In fact, if both $\mathcal{Y}_x = \{\mathcal{Y}\}$ and $\mathcal{Z}_x = \{\mathcal{Z}\}$ hold for all $x \in \mathcal{X}$, then A does not interact with any party in case of an attack. Additionally, in this case the assumption of the existence of the algorithm S and the distributions $\{P_x\}_{x\in\mathcal{X}}$ is made redundant.

We next present the protocol. We assume without loss of generality that $\mathcal{Z}_x = \{\mathcal{Z}\}$ for all $x \in \mathcal{X}$. First, similarly to the NIORP case, by the honest-majority assumption the parties can compute f fairly. If the parties receive an output, they can terminate; otherwise, similarly to [2] we let A and B compute the two-party functionality $f(x, y, z^*)$, where $z^* \leftarrow R_1$, ignoring C in the process (recall that since $\mathcal{Z}_x = \{\mathcal{Z}\}$ for all $x \in \mathcal{X}$ there is only one distribution given by Item 2 of Theorem 1.4).

Similarly to the NIORP case, correctness holds due to the correctness of the fair protocol. Furthermore, it is clear that a corrupt C cannot attack the protocol. Indeed, it does not gain any information in the fair computation of f; hence, if it aborts in this phase then the output of A is $g(x, y) = f(x, y, z^*)$, where $z^* \leftarrow R_1$. Similarly, a corrupt B cannot attack the protocol since its simulator can send $y^* \leftarrow Q_i$, where $i \in [n_\wedge]$ is such that $y \in \mathcal{Y}_i^\wedge$. By Item 2 of Theorem 1.4 the output of A in the ideal world is

$$f(x, y^*, z) \equiv f(x, y, z^*) \equiv g(x, y),$$

where $y^* \leftarrow Q_i$ and $z^* \leftarrow R_1$.

Next, consider a corrupt A. Similarly to the previous two cases, we only need to consider the case where A aborts during the fair computation of f. Observe that the only information it receives is $g(x, y) = f(x, y, z^*)$, where $z^* \leftarrow R_1$. The simulator will simply send $x^* \leftarrow P_x$ to the trusted party, and receive back w as the output. Then, the corrupt A will output whatever $\mathsf{S}(x, x^*, w)$ outputs. By Item 2 of Theorem 1.4, the simulator output is identically distributed as $g(x, y)$.

1.2.3 Reducing the Randomized Case to the Deterministic Case

We next explain how to reduce the randomized case to the deterministic case. The reduction works in both the positive and the negative directions. Thus, we obtain characterizations for randomized NIORP functionalities, and randomized ternary-output functionalities as well.

Recall that for a randomized solitary output three-party $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$, we define the deterministic solitary output three-party functionality

$$f'((x, r_1), (y, r_2), (z, r_3)) = f(x, y, z; r_1 + r_2 + r_3).$$

[6] For the general case, where the domain of f is not constant, the protocol we use is a slight generalization of the one suggested by [2]. Specifically, the decision of whether A interacts with B or C in case of an attack depends on the security parameter κ. Assuming the domain of f is of polynomial size in κ, the decision can be computed efficiently and locally by every party.

Namely, the parties hold a share of the randomness of f in a 3-out-of-3 secret sharing scheme. We next show that f can be securely computed in the point-to-point model if and only if f' can.

Let us first assume that f' can be securely computed. Then in order to compute f, the parties will compute f' with their original inputs, and where r_1, r_2, and r_3 are sampled uniformly at random. Security follows from the fact that at least one party is honest, hence either r_1, r_2, or r_3 are sampled uniformly at random.

Let us now assume that f can be securely computed. First, similarly to the previous protocols, by the honest-majority assumption the parties can compute f' with fairness. If the parties receive an output, they can terminate; otherwise they compute f on their respective inputs. Correctness is given by the fact that the parties first compute f' fairly. Security is guaranteed since the adversary obtains no information from the fair computation, and since the simulator can send a uniform random r as part of the input, in case the fair computation aborted.

1.2.4 When a Broadcast Channel is Available

In this section we show that any NIORP or ternary-output functionality that can be securely computed in the point-to-point model against a single corrupted party, can also be securely computed assuming a broadcast channel against two corruptions. Similarly to the point-to-point model, we will only handle deterministic functionalities, as the randomized case can be handled using the same reduction from Section 1.2.3.

The NIORP case. Let us start with describing a protocol for the NIORP functionalities captured by Theorem 1.2. Recall that for these functionalities there exist two families of efficiently samplable distributions $\{Q_i\}_{i\in[n]}$ and $\{R_j\}_{j\in[m]}$ such that the following holds. For all $i \in [n]$, $j \in [m]$, $y \in \mathcal{Y}_i$, and $z \in \mathcal{Z}_j$, it holds that

$$f(y^*, z) \equiv f(y, z^*),$$

where $y^* \leftarrow Q_i$ and $z^* \leftarrow R_j$.

In the following, we show that a larger class of functionalities than those described above, can be securely computed against two corruptions. Specifically, it suffices to assume the existence of only a single efficiently samplable distribution, one for B *or* one for C. By symmetry, we only consider the latter case. That is, we assume there exists $j^* \in [m]$ and there exists an efficiently samplable distribution R_{j^*} over $\mathcal{Z}_{j^*}$ such that the following holds. For every $i \in [n]$ and every $y, y' \in \mathcal{Y}_i$ it holds that

$$f(y, z^*) \equiv f(y', z^*), \tag{4}$$

where $z^* \leftarrow R_{j^*}$.

The protocol is an extension of one of the protocols suggested by [20], and it proceeds as follows. First, the parties compute a 3-out-of-3 secret sharing of the output $f(x, y, z)$ using a secure-with-identifiable-abort protocol (i.e., the adversary can force an abort after obtaining the output, but at the expense of revealing the identity of a corrupted party).[7] In case a single party aborts, the remaining two parties compute the function on their inputs and with the input of the aborting

[7] Additionally, the shares are also signed using a MAC to ensure that B and C won't change their values. For simplicity we assume that a malicious adversary does not modify these values, but can abort the execution.

party set to a default value. Observe that since f is solitary output, this can be done securely using the protocol of Kilian [21]. If both B and C abort, then A outputs $f(x, y_0, z_0)$, where $y_0 \in \mathcal{Y}$ and $z_0 \in \mathcal{Z}$ are default inputs.

If no abort occurs, then first B sends its share to A, and additionally, it sends the (unique) index $i \in [n]$ such that $y \in \mathcal{Y}_i$. If B aborts, then A and C compute $f(x, y_0, z)$. Otherwise, C sends its share to A. If C aborts, then A outputs $f(x, y_i, z^*)$, where y_i is the lexicographically smallest element in $\mathcal{Y}_i$, and where $z^* \leftarrow R_{j^*}$.

Similarly to the point-to-point case, corrupting A will not provide the adversary with any information since the index i can be inferred from the output given by the trusted party. Additionally, B and C obtain no information from the execution, since their views contain only secret shares of the output. Furthermore, if a corrupt B aborts (at any point during the computation), then it can be simulated by sending y_0 to the trusted party. Finally, if a corrupt C aborts after B sent its share, then this attack can be simulated by sending $z^* \leftarrow R_{j^*}$ to the trusted party. Then the output of A in the ideal world is $f(x, y, z^*)$, while its output in the real world is $f(x, y_i, z^*)$. By Equation (4), the two distributions are identical.

The ternary-output case. We now turn to ternary-output functionalities. In fact, we show that a much larger class of functionalities than those captured by Theorem 1.4, can be securely computed. Similarly to the point-to-point case, the protocol we present securely computes non-ternary-output functionalities. First, recall that by Item 1 of Theorem 1.4, it holds that either $\mathscr{Y}_x = \{\mathcal{Y}\}$ for all $x \in \mathcal{X}$, or $\mathscr{Z}_x = \{\mathcal{Z}\}$ for all $x \in \mathcal{X}$. We assume the latter without loss of generality. We next present a relaxation of Item 2 of Theorem 1.4 that suffices for f to be securely computable against two corruptions. Specifically, we assume that there exists two distributions Q and R over $\mathcal{Y}_i^{\wedge}$ and $\mathcal{Z}$, respectively, for some $i \in [n_{\wedge}]$, such that the following holds. There exists $y_i \in \mathcal{Y}_i^{\wedge}$, such that for all $z \in \mathcal{Z}$, and $x \in \mathcal{X}$, it holds that

$$f(x, y^*, z) \equiv f(x, y_i, z^*), \tag{5}$$

where $y^* \leftarrow Q$ and $z^* \leftarrow R$. Observe that this is indeed a relaxation, since we do not require the assumption of the existence of S, and the distributions $\{P_x\}_{x \in \mathcal{X}}$ and $\{Q_{i'}\}_{i' \in [n_{\wedge}] \setminus \{i\}}$, and since the quantifier over y_i is replaced with an existential quantifier.

The protocol proceeds as follows. The parties first compute a secret sharing of the output of $f(x, y, z)$ using a secure-with-identifiable-abort protocol. The sharing scheme is a 2-out-of-2 scheme, with the shares given only to A and B. Assuming the computation followed through, B sends its share to A, which reconstructs the output. If B aborts at any point in the computation, then A outputs $f(x, y_i, z^*)$ where $z^* \leftarrow R$. If C aborts during the secure-with-identifiable-abort computation, then its input is replaced with a default value and protocol restarts.

Clearly, corrupting C will not provide the adversary with any advantage. Additionally, corrupting A and (possibly) B will provide the adversary with only the output. The only case left is when B is corrupted and A is honest. In this case, the adversary gains no information from the secure-with-identifiable-abort computation, since it obtains only one share of the output. Now, if B aborts then we let its simulator send to the trusted party the input $y^* \leftarrow Q$. Then A outputs $f(x, y^*, z)$ in the ideal world. On the other hand, in the real world the output of A is $f(x, y_i, z^*)$. By Equation (5), the two distributions are identical.

1.3 Related Work

The hexagon argument has been used in the context of Byzantine agreement to rule out three-party protocols tolerating one corruption [22, 24, 15]. Cohen et al. [11] considered *symmetric* (possibly randomized) functionalities in the point-to-point model, and showed that a symmetric n-party functionality f can be computed against t corruptions, if and only if f is $(n-2t)$-dominated, i.e., there exists y^* such that any $n-2t$ of the inputs can fix the output of f to be y^*.

Recently, Alon et al. [2] extended the discussion to consider asymmetric functionalities in the point-to-point model. They provided various necessary and sufficient conditions for a functionality to be securely computable. They considered some interesting examples for the special case of solitary-output functionalities, however, provided no characterization for any class of functions.

Halevi et al. [20] investigated the round complexity required to compute solitary output functionalities assuming the availability of a broadcast channel, but no honest majority. They provided various negative and positive results. Solitary output computation was already considered in non-interactive setting of MPC, such as PSM [14] and its robust variants [6, 1]. Badrinarayanan et al. [5] investigated the round complexity required to compute solitary output functionalities, assuming the availability of a broadcast channel and no PKI, and vice versa.

1.4 Organization

The preliminaries and definition of the model of computation appear in Section 2. In Section 3 we state our results in the point-to-point model. Then, in Sections 4 and 5 we prove the negative and positive results, respectively. Finally, in Section 6 we state and prove our results assuming a broadcast channel.

2 Preliminaries

2.1 Notations

We use calligraphic letters to denote sets, uppercase for random variables and distributions, lowercase for values, and we use bold characters to denote vectors. For $n \in \mathbb{N}$, let $[n] = \{1, 2 \ldots n\}$. For a set $\mathcal{S}$ we write $s \leftarrow \mathcal{S}$ to indicate that s is selected uniformly at random from $\mathcal{S}$. Given a random variable (or a distribution) X, we write $x \leftarrow X$ to indicate that x is selected according to X. A PPT algorithm is probabilistic polynomial time, and a PPTM is a polynomial time (interactive) Turing machine.

A function $\mu \colon \mathbb{N} \to [0,1]$ is called negligible, if for every positive polynomial $p(\cdot)$ and all sufficiently large n, it holds that $\mu(n) < 1/p(n)$. We write neg for an unspecified negligible function, and write poly for an unspecified polynomial. For a randomized function (or an algorithm) f we write $f(x)$ to denote the random variable induced by the function on input x, and write $f(x;r)$ to denote the value when the randomness of f is fixed to r.

A *distribution ensemble* $X = \{X_{a,n}\}_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ is an infinite sequence of random variables indexed by $a \in \mathcal{D}_n$ and $n \in \mathbb{N}$, where $\mathcal{D}_n$ is a domain that might depend on n. When the domains are clear, we will sometimes write $\{X_{a,n}\}_{a,n}$ in order to alleviate notations.

The statistical distance between two finite distributions is defined as follows.

Definition 2.1. *The statistical distance between two finite random variables X and Y is*

$$\mathrm{SD}\left(X, Y\right) = \max_{\mathcal{S}} \left\{ \Pr\left[X \in \mathcal{S}\right] - \Pr\left[Y \in \mathcal{S}\right] \right\}.$$

For a function $\varepsilon : \mathbb{N} \mapsto [0,1]$, the two ensembles $X = \{X_{a,n}\}_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ and $Y = \{Y_{a,n}\}_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ are said to be ε-close, if for all sufficiently large n and $a \in \mathcal{D}_n$, it holds that

$$\mathrm{SD}\left(X_{a,n}, Y_{a,n}\right) \leq \varepsilon(n),$$

and are said to be ε-far otherwise. X and Y are said to be statistically close*, denoted $X \stackrel{\mathrm{S}}{\equiv} Y$, if they are ε-close for some negligible function ε. If X and Y are 0-close then they are said to be equivalent, denoted $X \equiv Y$.*

Computational indistinguishability is defined as follows.

Definition 2.2. *Let $X = \{X_{a,n}\}_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ and $Y = \{Y_{a,n}\}_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ be two ensembles. We say that X and Y are computationally indistinguishable, denoted $X \stackrel{\mathrm{C}}{\equiv} Y$, if for every non-uniform* PPT *distinguisher D, there exists a negligible function $\mu(\cdot)$, such that for all n and $a \in \mathcal{D}_n$, it holds that*

$$\left|\Pr\left[\mathsf{D}(X_{a,n}) = 1\right] - \Pr\left[\mathsf{D}(Y_{a,n}) = 1\right]\right| \leq \mu(n).$$

The following is simple fact states that whenever two ensembles with polynomial-size supports are computationally indistinguishable, they are also statistically close.

Fact 2.3. *Let $X = \{X_{a,n}\}_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ and $Y = \{Y_{a,n}\}_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ be two computationally indistinguishable ensembles over a set-family $\{\mathcal{S}_n\}_{n \in \mathbb{N}}$, of size $|\mathcal{S}_n| \leq \mathrm{poly}(n)$. Then $X \stackrel{\mathrm{S}}{\equiv} Y$.*

Proof sketch. Assume for the sake of contradiction that the claim is false. It follows that there exists a set-family $\{\mathcal{T}_n\}_{n \in \mathbb{N}}$ where $\Pr\left[X_{a,n} \in \mathcal{T}_n\right] - \Pr\left[Y_{a,n} \in \mathcal{T}_n\right] \geq 1/p(n)$, for some polynomial p. Since $\mathcal{T}_n \subseteq \mathcal{S}_n$, it follows that $\mathcal{T}_n$ is of polynomial size, hence it can be given as auxiliary input to a bounded distinguisher D. Then, D can distinguish X from Y by outputting 1 if its input belongs to $\mathcal{T}_n$, and outputting 0 otherwise, thus contradicting the assumption that $X \stackrel{\mathrm{C}}{\equiv} Y$. $\square$

The following fact states an equivalent definition for statistical distance.

Fact 2.4. *Let $X = \{X_{a,n}\}_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ and $Y = \{Y_{a,n}\}_{a \in \mathcal{D}_n, n \in \mathbb{N}}$ be two ensembles. Then $X \stackrel{\mathrm{S}}{\equiv} Y$ if and only if for every unbounded distinguisher D, there exists a negligible function $\mu(\cdot)$, such that for all n and $a \in \mathcal{D}_n$, it holds that*

$$\left|\Pr\left[\mathsf{D}(X_{a,n}) = 1\right] - \Pr\left[\mathsf{D}(Y_{a,n}) = 1\right]\right| \leq \mu(n).$$

Definition 2.5 (Minimal and minimum elements)**.** *Let $\mathcal{S}$ be a set and let $\preceq$ be a partial order over $\mathcal{S}$. An element $s \in \mathcal{S}$ is called minimal, if no other element is smaller than s, that is, for any $s' \in \mathcal{S}$, if $s' \preceq s$ then $s' = s$.*

An element $s \in \mathcal{S}$ is called minimum if it is smaller then any other element, that is, for any $s' \in \mathcal{S}$ it holds that $s \preceq s'$.

We next define a refinement of a partition of some set.

Definition 2.6 (Refinement of partitions)**.** *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two partitions of some set $\mathcal{S}$. We say that $\mathcal{P}_1$* refines *$\mathcal{P}_2$, if for every $\mathcal{S}_1 \in \mathcal{P}_1$ there exists $\mathcal{S}_2 \in \mathcal{P}_2$ such that $\mathcal{S}_1 \subseteq \mathcal{S}_2$.*

The meet of two partitions is the partition formed by taking all non-empty intersections. Formally, it is defined as follows.

Definition 2.7 (Meet of partitions)**.** *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two partitions of some set $\mathcal{S}$. The* meet *of $\mathcal{P}_1$ and $\mathcal{P}_2$, denoted $\mathcal{P}_1 \wedge \mathcal{P}_2$, is defined as*

$$\mathcal{P}_1 \wedge \mathcal{P}_2 := \{\mathcal{S}_1 \cap \mathcal{S}_2 \mid \forall i \in \{1,2\} : \mathcal{S}_i \in \mathcal{P}_i \text{ and } \mathcal{S}_1 \cap \mathcal{S}_2 \neq \emptyset\}.$$

Observe that $\wedge$ is associative, thus we can naturally extend the definition for several partitions.

Definition 2.8 (Equivalence class and quotient sets)**.** *For an equivalence relation $\equiv$ over some set $\mathcal{S}$, and an element $s \in \mathcal{S}$ we denote by $[s]_\equiv$ the equivalence class of s, i.e.,*

$$[s]_\equiv := \{s' \in \mathcal{S} : s \equiv s'\}.$$

We let $\mathcal{S}/\equiv$ denote the quotient set with respect to $\equiv$ defined as the set of all equivalence classes. Stated differently, it is the partition of $\mathcal{S}$ induced by the equivalence relation $\equiv$.

2.2 The Model of Computation

We provide the basic definitions for secure multiparty computation according to the real/ideal paradigm, for further details see [17]. Intuitively, a protocol is considered secure if whatever an adversary can do in the real execution of protocol, can be done also in an ideal computation, in which an uncorrupted trusted party assists the computation.

In this paper we focus mostly on solitary output three-party functionalities. A functionality is a sequence of function $f = \{f_\kappa\}_{\kappa \in \mathbb{N}}$, where $f_\kappa \colon \mathcal{X}_\kappa \times \mathcal{Y}_\kappa \times \mathcal{Z}_\kappa \mapsto \mathcal{W}_\kappa$ for every $\kappa \in \mathbb{N}$.[8] The functionality is called solitary output if only one party obtains an output. We denote the parties by A, B and C, holding inputs x, y, and z, respectively, and let A receive the output, denoted w. To alleviate notations, we will remove κ from f and its domain and range, and simply write it as $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$.

Although we mostly focus one solitary output functionalities and deal with adversaries that corrupt a single party, we present the definition for the general case, as it will be useful later.

The Real Model

A three-party protocol π is defined by a set of three PPT interactive Turing machines A, B, and C. Each Turing machine (party) holds at the beginning of the execution the common security parameter 1^κ, a private input, and random coins. The *adversary* $\mathcal{A}$ is another PPT interactive Turing machine describing the behavior of the corrupted parties. It starts the execution with input that contains the identities of the corrupted parties, their inputs, and an additional auxiliary input aux.

The parties execute the protocol over a synchronous network. That is, the execution proceeds in rounds: each round consists of a *send phase* (where parties send their messages for this round) followed by a *receive phase* (where they receive messages from other parties).

We consider a fully connected point-to-point network, where every pair of parties is connected by a communication line. We will consider the *secure-channels* model, where the communication lines are assumed to be ideally private (and thus the adversary cannot read or modify messages sent between two honest parties). Depending on the context, we may assume the parties have

[8] The typical convention in secure computation is to let $f : (\{0,1\}^*)^3 \mapsto \{0,1\}^*$. However, we will mostly be dealing with functionalities whose domain is of polynomial size in κ, which is why we introduce this notation.

access to a broadcast channel. We note that our upper bounds (protocols) can also be stated in the *authenticated-channels* model, where the communication lines are assumed to be ideally authenticated but not private (and thus the adversary cannot modify messages sent between two honest parties but can read them) via standard techniques, assuming public-key encryption. On the other hand, stating our lower bounds assuming secure channels will provide stronger results.

We consider a fully connected synchronous point-to-point network, where every pair of parties is connected by a communication line. We will consider the *secure-channels* model, where the communication lines are assumed to be ideally private (and thus the adversary cannot read or modify messages sent between two honest parties). Depending on the context, we may assume the parties have access to a broadcast channel.

Throughout the execution of the protocol, all the honest parties follow the instructions of the prescribed protocol, whereas the corrupted parties receive their instructions from the adversary. The adversary is considered to be *malicious*, meaning that it can instruct the corrupted parties to deviate from the protocol in any arbitrary way. Additionally, the adversary has full-access to the view of the corrupted parties, which consists of their inputs, their random coins, and the messages they see throughout this execution. At the conclusion of the execution, the honest parties output their prescribed output from the protocol, the corrupted parties output nothing, and the adversary outputs a function of its view. In some of our proofs we consider *semi-honest* adversaries that always instruct the corrupted parties to honestly execute the protocol, but may try to learn more information than they should.

We consider malicious adversaries, meaning that it can instruct the corrupted parties to deviate from the protocol in any arbitrary way. The adversary has full-access to the view of the corrupted parties, which consists of their inputs, their random coins, and the messages they see throughout this execution. At the conclusion of the execution, the honest parties output their prescribed output from the protocol, the corrupted parties output nothing, and the adversary outputs a function of its view. In some of our proofs we consider *semi-honest* adversaries that always instruct the corrupted parties to honestly execute the protocol.

We next define the real-world global view for security parameter $\kappa \in \mathbb{N}$, inputs $x, y, z \in \{0,1\}^*$, and an auxiliary input $\mathsf{aux} \in \{0,1\}^*$ with respect to some adversary $\mathcal{A}$ controlling a subset $\mathcal{I} \subseteq \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ of the parties. Let $\text{OUT}^{\mathsf{real}}_{\pi,\mathcal{A}(\mathsf{aux})}(\kappa,(x,y,z))$ denote the outputs of the honest parties in a random execution of π on inputs (x, y, z) and security parameter κ interacting with $\mathcal{A}$ with auxiliary input aux corrupting the parties in $\mathcal{I}$. Further let $\text{VIEW}^{\mathsf{real}}_{\pi,\mathcal{A}(\mathsf{aux})}(\kappa,(x,y,z))$ be the adversary's output, being a function of its view (i.e., its auxiliary input, its random coins, the input of the corrupted party, and the messages it sees during the execution of the protocol) during an execution of π. We denote the global view in the real model by

$$\text{REAL}_{\pi,\mathcal{A}(\mathsf{aux})}(\kappa,(x,y,z)) = \left(\text{VIEW}^{\mathsf{real}}_{\pi,\mathcal{A}(\mathsf{aux})}(\kappa,(x,y,z))\,,\ \text{OUT}^{\mathsf{real}}_{\pi,\mathcal{A}(\mathsf{aux})}(\kappa,(x,y,z))\right).$$

The Ideal Model

We consider an ideal computation with *guaranteed output delivery* (also referred to as *full security*), where a trusted party performs the computation on behalf of the parties, and the ideal-model adversary *cannot* abort the computation. An ideal computation of a three-party functionality $f = (f_1, f_2, f_3)$, with $f_1, f_2, f_3 : (\{0,1\}^*)^3 \to \{0,1\}^*$, on inputs $x, y, z \in \{0,1\}^*$ and security parameter κ, with an ideal-world adversary $\mathcal{A}$ running with an auxiliary input aux and corrupting a subset $\mathcal{I} \subseteq \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ of the parties, proceeds as follows:

Parties send inputs to the trusted party: Each honest party sends its input to the trusted party. The adversary $\mathcal{A}$ sends a value v from its domain as the input for the corrupted party. Let (x', y', z') denote the inputs received by the trusted party.

The trusted party performs computation: The trusted party selects a random string r, computes $(w_1, w_2, w_3) = f(x', y', z'; r)$, and sends w_1 to A, sends w_2 to B, and sends w_3 to C.

Outputs: Each honest party outputs whatever output it received from the trusted party and the corrupted party outputs nothing. The adversary $\mathcal{A}$ outputs some function of its view (i.e., the auxiliary input, its randomness, and the input and output of the corrupted party).

We next define the ideal-world global view for security parameter $\kappa \in \mathbb{N}$, inputs $x, y, z \in \{0,1\}^*$, and an auxiliary input $\mathsf{aux} \in \{0,1\}^*$ with respect to some adversary $\mathcal{A}$ controlling a subset $\mathcal{I}$ of the parties. Let $\text{OUT}^{\mathsf{ideal}}_{f,\mathcal{A}(\mathsf{aux})}(\kappa, (x, y, z))$ denote the output of honest parties in a random execution of the above ideal-world process, interacting with $\mathcal{A}$. Further let $\text{VIEW}^{\mathsf{ideal}}_{f,\mathcal{A}(\mathsf{aux})}(\kappa, (x, y, z))$ be the *output* (a simulated view) of $\mathcal{A}$ in such a process. We denote the global view in the ideal model by

$$\text{IDEAL}_{f,\mathcal{A}(\mathsf{aux})}(\kappa, (x, y, z)) = \left(\text{VIEW}^{\mathsf{ideal}}_{f,\mathcal{A}(\mathsf{aux})}(\kappa, (x, y, z)),\ \text{OUT}^{\mathsf{ideal}}_{f,\mathcal{A}(\mathsf{aux})}(\kappa, (x, y, z))\right).$$

The Security Definition

Having defined the real and ideal models, we can now define security of protocols according to the real/ideal paradigm.

Definition 2.9 (Malicious security)**.** *Let f be a three-party functionality and let π be a three-party protocol. For $t \in \{1, 2\}$, we say that π computes f with t-*security*, if for every non-uniform* PPT *adversary $\mathcal{A}$, controlling a subset $\mathcal{I} \subseteq \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ of size at most t in the real-world, there exists a non-uniform* PPT *adversary Sim, controlling the same subset $\mathcal{I}$ in the ideal-world such that*

$$\left\{\text{IDEAL}_{f,\mathsf{Sim}(\mathsf{aux})}(\kappa, (x, y, z))\right\}_{\kappa\in\mathbb{N},x,y,z,\mathsf{aux}\in\{0,1\}^*} \stackrel{\mathrm{C}}{\equiv} \left\{\text{REAL}_{\pi,\mathcal{A}(\mathsf{aux})}(\kappa, (x, y, z))\right\}_{\kappa\in\mathbb{N},x,y,z,\mathsf{aux}\in\{0,1\}^*}.$$

We define statistical t-security similarly, by replacing computational indistinguishability with statistical distance.

When $t = 2$ we will sometimes say that π computes f will full security*.*

Ideal computation with fairness. Although all our results are stated with respect to guaranteed output delivery, in our proofs we will consider a weaker security variant, where the adversary may cause the computation to prematurely abort, but only before it learns any new information from the protocol. Formally, security with fairness is defined by only modifying the ideal computation. Specifically, the difference is that during the Parties send inputs to the trusted party step, the adversary can send a special abort symbol. In this case, the trusted party send $\perp$ to all parties instead of computing the function.

Ideal computation with security-with-identifiable-abort. We also use a security notion called security-with-identifiable-abort where, similarly to fairness,the adversary can cause the computation to prematurely abort. However, it can do so after learning the output, at the expense of revealing the identity of a corrupted party (see Appendix A for a formal definition).

The Hybrid Model

The *hybrid model* is a model that extends the real model with a trusted party that provides ideal computation for specific functionalities. The parties communicate with this trusted party in exactly the same way as in the ideal models described above.

Let f be a functionality. Then, an execution of a protocol π computing a functionality g in the f-hybrid model involves the parties sending normal messages to each other (as in the real model) and in addition, having access to a trusted party computing f. It is essential that the invocations of f are done sequentially, meaning that before an invocation of f begins, the preceding invocation of f must finish. In particular, there is at most a single call to f per round, and no other messages are sent during any round in which f is called.

Let $\mathsf{type} \in \{\mathsf{g.o.d.}, \mathsf{fair}, \mathsf{s.w.i.a}\}$, and let $\mathcal{A}$ be a non-uniform PPT machine with auxiliary input aux controlling a subset of the parties. We denote by $\textsc{hybrid}^{f,\mathsf{type}}_{\pi,\mathcal{A}(\mathsf{aux})}(\kappa,(x,y,z))$ the random variable consisting of the view of the adversary and the output of the honest parties, following an execution of a protocol π with ideal calls to a trusted party computing f according to the ideal model "type," on input vector (x,y,z), auxiliary input aux to $\mathcal{A}$, and security parameter κ. We call this the (f,type)-hybrid model.

The sequential composition theorem of Canetti [8] states the following. Let ρ be a protocol that securely computes f in the ideal model "type." Then, if a protocol π computes g in the (f,type)-hybrid model, then the protocol π^ρ, that is obtained from π by replacing all ideal calls to the trusted party computing f with the protocol ρ, securely computes g in the real model.

Theorem 2.10 ([8])**.** *Let $t \in \{1,2\}$, let f be a three-party functionality, let $\mathsf{type}_1, \mathsf{type}_2 \in \{\mathsf{g.o.d.}, \mathsf{fair}, \mathsf{s.w.i.a}\}$, let ρ be a protocol that t-securely computes f with type_1, and let π be a protocol that t-securely computes g with type_2 in the (f,type_1)-hybrid model. Then, protocol π^ρ t-securely computes g with type_2 in the real model.*

We make use of a known fact stating that any functionality can be computed fairly assuming an honest majority.

Fact 2.11. *Let f be a three-party functionality. Then (f,fair) can be computed with statistical 1-security.*

This follows from the results of [25] and [10]. Specifically, Rabin and Ben-Or [25] showed how to compute any functionality with full security assuming an honest majority and a *broadcast* channel. On the other hand, Cohen and Lindell [10] showed that any protocol computing some functionality f with full security assuming a broadcast channel, can be transformed into a protocol computing f fairly over a point-to-point network without the use of broadcast.

3 Our Main Results in the Point-to-Point Model

In this section we present the statement of our main results in the point-to-point model. We present a necessary condition and two sufficient conditions for solitary output three-party functionalities with polynomial-sized domains, that can be computed with 1-security without broadcast. In Section 3.2.1, we present several corollaries of our results. In particular, we show that various interesting families of functionalities, such as deterministic NIORP and (possibly randomized) ternary-output functionalities, our necessary and sufficient conditions are equivalent, thus we obtain a characterization.

3.1 Useful Definitions

Before stating the result, we first present several important definitions. Throughout the entire subsection, we let $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ be a deterministic solitary output three-party functionality.

The first definition introduces an equivalence relation over the domains $\mathcal{Y}$ and $\mathcal{Z}$ with respect to any fixed input $x \in \mathcal{X}$. We call this relation the ***co****mmon* ***o****utput* ***re****lation* (CORE). Note that the relation depends on the security parameter κ as well. We will not write κ as part of the notations in order to alleviate them.

Definition 3.1 (CORE and CORE partition)**.** *For an input $x \in \mathcal{X}$ we define the relation $\sim_x$ over $\mathcal{Y}$ as follows.*

$$y \sim_x y' \text{ if there exist } z, z' \in \mathcal{Z} \text{ such that } f(x, y, z) = f(x, y', z').$$

We define relation $\equiv_x$, called CORE, *to be the* transitive closure *of $\sim_x$, i.e., $y \equiv_x y'$ if either $y \sim_x y'$ or if there exists $y_1, \ldots, y_k \in \mathcal{Y}$ such that*

$$y \sim_x y_1 \sim_x \ldots \sim_x y_k \sim_x y'.$$

Observe that $\equiv_x$ is an equivalence relation. We let $\mathfrak{Y}_x$ denote the set of equivalence classes of $\mathcal{Y}$ formed by $\equiv_x$. We also abuse notations, and define the relations $z \sim_x z'$ and $z \equiv_x z'$ over $\mathcal{Z}$ similarly, and let $\mathfrak{Z}_x$ denote the set of equivalence classes over $\mathcal{Z}$ formed by $\equiv_x$.

Additionally, we denote $n(x) = |\mathfrak{Y}_x|$, $m(x) = |\mathfrak{Z}_x|$, and we write

$$\mathfrak{Y}_x = \{\mathcal{Y}_i^x : i \in [n(x)]\} \quad \text{and} \quad \mathfrak{Z}_x = \{\mathcal{Z}_j^x : j \in [m(x)]\}.$$

Finally, we let

$$\mathfrak{R}_x = \{\mathcal{Y}_i^x \times \mathcal{Z}_j^x : i \in [n(x)], j \in [m(x)]\}$$

be the partition of $\mathcal{Y} \times \mathcal{Z}$ into the combinatorial rectangles formed by $\mathfrak{Y}_x$ and $\mathfrak{Z}_x$. We call $\mathfrak{Y}_x$, $\mathfrak{Z}_x$, and $\mathfrak{R}_x$ the CORE partitions of f with respect to x.

We next introduce equivalence relations over $\mathcal{X}$ that corresponds to the CORE partitions formed by the inputs. In addition, we define partial orders over the the quotient sets associated with theses equivalence relations. Roughly, both the equivalence relations and the partial orders are defined by comparing the corresponding CORE partitions. Similarly to Definition 3.1, the following definition also depends κ, which is omitted from the notations to alleviate them.

Definition 3.2 (Equivalence relations and partial orders over $\mathcal{X}$)**.** *We define three equivalence relations $\equiv_{\mathsf{B}}$, $\equiv_{\mathsf{C}}$, and $\equiv$, over $\mathcal{X}$ as follows.*

- *We say that $x \equiv_{\mathsf{B}} x'$ if $\mathfrak{Y}_x = \mathfrak{Y}_{x'}$.*
- *We say that $x \equiv_{\mathsf{C}} x'$ if $\mathfrak{Z}_x = \mathfrak{Z}_{x'}$*
- *We say that $x \equiv x'$ if $\mathfrak{R}_x = \mathfrak{R}_{x'}$. Equivalently, $x \equiv x'$ if $x \equiv_{\mathsf{B}} x'$ and $x \equiv_{\mathsf{C}} x'$.*

We define partial orders $\preceq_{\mathsf{B}}$, $\preceq_{\mathsf{C}}$, and $\preceq$ over the quotient sets $\mathcal{X}/{\equiv_{\mathsf{B}}}$, $\mathcal{X}/{\equiv_{\mathsf{C}}}$, and $\mathcal{X}/{\equiv}$, respectively, as follows.

- *We say that $[x]_{\equiv_{\mathsf{B}}} \preceq_{\mathsf{B}} [x']_{\equiv_{\mathsf{B}}}$ if $\mathfrak{Y}_x$ refines $\mathfrak{Y}_{x'}$.*
- *We say that $[x]_{\equiv_{\mathsf{C}}} \preceq_{\mathsf{C}} [x']_{\equiv_{\mathsf{C}}}$ if $\mathfrak{Z}_x$ refines $\mathfrak{Z}_{x'}$.*

- *We say that* $[x]_\equiv \preceq [x']_\equiv$ *if* $\mathcal{R}_x$ *refines* $\mathcal{R}_{x'}$. *Equivalently,* $[x]_\equiv \preceq [x']_\equiv$ *if* $[x]_{\equiv_{\mathsf{B}}} \preceq_{\mathsf{B}} [x']_{\equiv_{\mathsf{B}}}$ *and* $[x]_{\equiv_{\mathsf{C}}} \preceq_{\mathsf{C}} [x']_{\equiv_{\mathsf{C}}}$.

For brevity, we write the partial orders as if they are over $\mathcal{X}$*, e.g., we write* $x \preceq_{\mathsf{B}} x'$ *instead of* $[x]_{\equiv_{\mathsf{B}}} \preceq_{\mathsf{B}} [x']_{\equiv_{\mathsf{B}}}$. *Finally,* $\chi \in \mathcal{X}$ *is called* B*-minimal if* $[\chi]_{\equiv_{\mathsf{B}}}$ *is minimal with respect to* $\preceq_{\mathsf{B}}$*,* χ *is called* C*-minimal if* $[\chi]_{\equiv_{\mathsf{C}}}$ *is minimal with respect to* $\preceq_{\mathsf{C}}$*, and* χ *is called* R*-minimal if* $[\chi]_\equiv$ *is minimal with respect to* $\preceq$*.*[9]

As mentioned in Section 1, we are interested in the meet of all CORE partitions. We call this new partition the CORE$_\wedge$-partition of f. Similarly to previous notations, CORE$_\wedge$-partition also depends on κ, and we will not write it to alleviate notations.

Definition 3.3 (CORE$_\wedge$-partition)**.** *We denote*

$$\mathcal{Y}_\wedge := \bigwedge_{x\in\mathcal{X}} \mathcal{Y}_x = \bigwedge_{\substack{\chi\in\mathcal{X}:\\ \chi \text{ is } \mathsf{R}\text{-minimal}}} \mathcal{Y}_\chi \quad \text{and} \quad \mathcal{Z}_\wedge := \bigwedge_{x\in\mathcal{X}} \mathcal{Z}_x = \bigwedge_{\substack{\chi\in\mathcal{X}:\\ \chi \text{ is } \mathsf{R}\text{-minimal}}} \mathcal{Z}_\chi,$$

and call these two partitions the CORE$_\wedge$-partitions *of* f. *We let* $n_\wedge = |\mathcal{Y}_\wedge|$ *and* $m_\wedge = |\mathcal{Z}_\wedge|$*, and we write the partitions as*

$$\mathcal{Y}_\wedge := \{\mathcal{Y}_i^\wedge : i \in [n_\wedge]\} \quad \text{and} \quad \mathcal{Z}_\wedge := \{\mathcal{Z}_j^\wedge : j \in [m_\wedge]\}.$$

Finally, we let

$$\mathcal{R}_\wedge = \{\mathcal{Y}_i^\wedge \times \mathcal{Z}_j^\wedge : i \in [n_\wedge], j \in [m_\wedge]\},$$

be the partition of $\mathcal{Y} \times \mathcal{Z}$ *into the combinatorial rectangles formed by* $\mathcal{Y}_\wedge$ *and* $\mathcal{Z}_\wedge$.

The partitions $\mathcal{Y}_\wedge$ *and* $\mathcal{Z}_\wedge$ *are naturally associated with an equivalence relation* $\equiv_\wedge$ *over* $\mathcal{Y}$ *and over* $\mathcal{Z}$*, respectively: We say that* $y \equiv_\wedge y'$ *if there exists* $\mathcal{Y}^\wedge \in \mathcal{Y}_\wedge$ *such that* $y, y' \in \mathcal{Y}^\wedge$. *Equivalently,* $y \equiv_\wedge y'$ *if* $y \equiv_\chi y'$ *for all* R*-minimal* $\chi \in \mathcal{X}$. *Similarly, we say that* $z \equiv_\wedge z'$ *if there exists* $\mathcal{Z}^\wedge \in \mathcal{Z}_\wedge$ *such that* $z, z' \in \mathcal{Z}^\wedge$.

We next define an important special property of a functionality f, which we call CORE$_\wedge$-*forced*. This property plays a central role in both our positive and negative results, and generalizes the forced property defined in [20], which states that any party can fix the distribution of the output, using an appropriate distribution over its input.

Roughly, f is called CORE$_\wedge$-forced if both B and C can each associate a distribution to each set in the CORE$_\wedge$-partition of their respective set of inputs, such that the output distribution of A in each combinatorial rectangle in $\mathcal{R}_\wedge$ is fixed for every input $x \in \mathcal{X}$.

Definition 3.4 (CORE$_\wedge$-forced)**.** *The function* f *is said to be* CORE$_\wedge$-forced *if there exists two ensembles of efficiently samplable distributions* $\mathcal{Q} = \{Q_{\kappa,i}\}_{\kappa\in\mathbb{N},i\in[n_\wedge]}$ *and* $\mathcal{R} = \{R_{\kappa,j}\}_{\kappa\in\mathbb{N},j\in[m_\wedge]}$ *over* $\mathcal{Y}$ *and* $\mathcal{Z}$*, respectively, such that the following holds.*

$$\begin{aligned}\{f(x, y^*, z_j)\}_{\kappa\in\mathbb{N},x\in\mathcal{X},i\in[n_\wedge],j\in[m_\wedge],y\in\mathcal{Y}_i^\wedge,z\in\mathcal{Z}_j^\wedge} &\stackrel{\text{S}}{\equiv} \{f(x, y^*, z)\}_{\kappa\in\mathbb{N},x\in\mathcal{X},i\in[n_\wedge],j\in[m_\wedge],y\in\mathcal{Y}_i^\wedge,z\in\mathcal{Z}_j^\wedge} \\ &\stackrel{\text{S}}{\equiv} \{f(x, y, z^*)\}_{\kappa\in\mathbb{N},x\in\mathcal{X},i\in[n_\wedge],j\in[m_\wedge],y\in\mathcal{Y}_i^\wedge,z\in\mathcal{Z}_j^\wedge} \\ &\stackrel{\text{S}}{\equiv} \{f(x, y_i, z^*)\}_{\kappa\in\mathbb{N},x\in\mathcal{X},i\in[n_\wedge],j\in[m_\wedge],y\in\mathcal{Y}_i^\wedge,z\in\mathcal{Z}_j^\wedge},\end{aligned}$$

[9] Note that if we had defined $\preceq_{\mathsf{B}}$, $\preceq_{\mathsf{C}}$, and $\preceq$ directly over $\mathcal{X}$, then they would not correspond to partial orders. Indeed, for the relations to be partial orders, it required that they are antisymmetric, i.e., if $x \preceq x'$ and $x' \preceq x$ then $x = x'$. Observe that this is not generally the case, as the only guarantee we have is that $x \equiv x'$.

where $y^* \leftarrow Q_{\kappa,i}$, $z^* \leftarrow R_{\kappa,j}$, and where y_i and z_j are the lexicographically smallest elements in $\mathcal{Y}_i^\wedge$ and $\mathcal{Z}_j^\wedge$, respectively.

Remark 3.5. *Though our lowerbound shows that any securely computable solitary output functionality must be* $\mathrm{CORE}_\wedge$*-partition, this can be strengthen as follows. Instead of requiring that every rectangle in $\mathcal{R}_\wedge$ is fixed for every x, it suffices to consider the meet of partitions formed by the CORE partitions with respect to all* R*-minimal elements that are smaller than x, i.e., $\bigwedge_{\chi \preceq x : \chi \text{ is } \mathsf{R}\text{-minimal}} \mathcal{R}_\chi$. Then our lowerbound shows that for any x, the output distributions in the above collections of rectangles are fixed.*

3.2 Our Main Results

We are now ready to state our results, providing both sufficient and necessary conditions for a deterministic solitary output three-party functionalities with polynomial sized domain, to be computable with 1-security over point-to-point channels. The result for randomized functionalities, where the domain of the randomness is polynomial as well, is handled below in Proposition 3.10 by reducing it to the deterministic case. We start with stating our negative results.

Theorem 3.6. *Let $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ be a deterministic solitary output three-party functionality. Assume that oblivious transfer exists and $|\mathcal{X}|, |\mathcal{Y}|, |\mathcal{Z}| = \mathrm{poly}(\kappa)$. If f can be computed with 1-security, then the following hold.*

1. *For all sufficiently large $\kappa \in \mathbb{N}$, and all χ_{B} and χ_{C} that are* B*-minimal and* C*-minimal, respectively, there exists an* R*-minimal $\chi \in \mathcal{X}$ such that $\chi_{\mathsf{B}} \equiv_{\mathsf{B}} \chi \equiv_{\mathsf{C}} \chi_{\mathsf{C}}$.*

2. *f is* $\mathrm{CORE}_\wedge$*-forced.*

Moreover, suppose that f has the property that for all sufficiently large κ, it holds that either $y \equiv_x y'$ for all $x \in \mathcal{X}$ and $y, y' \in \mathcal{Y}$, or $z \equiv_x z'$ for all $x \in \mathcal{X}$ and $z, z' \in \mathcal{Z}$. Then there exists an ensemble of efficiently samplable distributions $\mathcal{P} = \{P_{\kappa,x}\}_{\kappa \in \mathbb{N}, x \in \mathcal{X}}$ and there exists a PPT *algorithm S such that*

$$\{\mathsf{S}(1^\kappa, x, x^*, f(x^*, y, z))\}_{\kappa \in \mathbb{N}, x \in \mathcal{X}, i \in [n_\wedge], j \in [m_\wedge], y \in \mathcal{Y}_i^\wedge, z \in \mathcal{Z}_j^\wedge} \stackrel{\mathrm{S}}{\equiv} \{f(x, y^*, z)\}_{\kappa \in \mathbb{N}, x \in \mathcal{X}, i \in [n_\wedge], j \in [m_\wedge], y \in \mathcal{Y}_i^\wedge, z \in \mathcal{Z}_j^\wedge},$$

where $x^* \leftarrow P_{\kappa,x}$ and $y^* \leftarrow Q_{\kappa,i}$, where $Q_{\kappa,i}$ is the distribution given the $\mathrm{CORE}_\wedge$*-forced property.*

The proof is given in Section 4.

We now state our two positive results. The first positive result considers functionalities that satisfy the property given in the "moreover" part of Theorem 3.6. Specifically, we get a characterization (see Corollary 3.8 below) for when such functionalities can be computed securely. Interestingly, the protocol used in the proof of the theorem below is a slight generalization of the protocol suggested by [2].

Theorem 3.7. *Let $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ be a deterministic solitary output three-party functionality. Assume that oblivious transfer exists, that $|\mathcal{X}|, |\mathcal{Y}|, |\mathcal{Z}| = \mathrm{poly}(\kappa)$, and that the following hold.*

1. *For all sufficiently large κ, either $y \equiv_x y'$ for all $x \in \mathcal{X}$ and $y, y' \in \mathcal{Y}$, or $z \equiv_x z'$ for all $x \in \mathcal{X}$ and $z, z' \in \mathcal{Z}$.*

2. *f is* $\mathrm{CORE}_\wedge$*-forced.*

3. *There exists an ensemble of efficiently samplable distributions* $\mathcal{P} = \{P_{\kappa,x}\}_{\kappa\in\mathbb{N},x\in\mathcal{X}}$ *and a* PPT *algorithm* S *such that*

$$\{\mathsf{S}(1^\kappa, x, x^*, f(x^*, y, z))\}_{\kappa\in\mathbb{N},x\in\mathcal{X},i\in[n_\wedge],j\in[m_\wedge],y\in\mathcal{Y}_i^\wedge,z\in\mathcal{Z}_j^\wedge} \stackrel{\text{S}}{\equiv} \{f(x, y^*, z)\}_{\kappa\in\mathbb{N},x\in\mathcal{X},i\in[n_\wedge],j\in[m_\wedge],y\in\mathcal{Y}_i^\wedge,z\in\mathcal{Z}_j^\wedge},$$

where $x^* \leftarrow P_{\kappa,x}$ *and* $y^* \leftarrow Q_{\kappa,i}$*, where* $Q_{\kappa,i}$ *is the distribution given the* $\text{CORE}_\wedge$*-forced property.*

Then f *can be computed with 1-security.*

We thus have the following corollary, stating a characterization for a special class of functionalities.

Corollary 3.8. *Let* $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ *be a deterministic solitary output three-party functionality. Assume that oblivious transfer exists and that* $|\mathcal{X}|, |\mathcal{Y}|, |\mathcal{Z}| = \text{poly}(\kappa)$*. Further assume that* f *has the property that for all sufficiently large* κ*, either* $y \equiv_x y'$ *for all* $x \in \mathcal{X}$ *and* $y, y' \in \mathcal{Y}$*, or* $z \equiv_x z'$ *for all* $x \in \mathcal{X}$ *and* $z, z' \in \mathcal{Z}$*. Then* f *can be computed with 1-security if and only if the following hold.*

1. f *is* $\text{CORE}_\wedge$*-forced.*

2. *There exists an ensemble of efficiently samplable distributions* $\mathcal{P} = \{P_{\kappa,x}\}_{\kappa\in\mathbb{N},x\in\mathcal{X}}$ *and a* PPT *algorithm* S *such that*

$$\{\mathsf{S}(1^\kappa, x, x^*, f(x^*, y, z))\}_{\kappa\in\mathbb{N},x_\kappa\in\mathcal{X},i\in[n_\wedge],j\in[m_\wedge],y\in\mathcal{Y}_i^\wedge,z\in\mathcal{Z}_j^\wedge} \stackrel{\text{S}}{\equiv} \{f(x, y^*, z)\}_{\kappa\in\mathbb{N},x_\kappa\in\mathcal{X},i\in[n_\wedge],j\in[m_\wedge],y\in\mathcal{Y}_i^\wedge,z\in\mathcal{Z}_j^\wedge},$$

where $x^* \leftarrow P_{\kappa,x}$ *and* $y^* \leftarrow Q_{\kappa,i}$*, where* $Q_{\kappa,i}$ *is the distribution given the* $\text{CORE}_\wedge$*-forced property.*

The proof of Theorem 3.7 is given in Section 5.1. The next result gives another sufficient condition. In fact, it characterizes a special class of functionalities, which includes (deterministic) NIORP functionalities, where the output-receiving party A has no input (see Corollary 3.15 below). Here, instead of assuming the functionality satisfies the property stated in the "moreover" part of Theorem 3.6, we assume that A has a *minimum input*, i.e., smaller than all other inputs with respect to $\preceq$.

Theorem 3.9. *Let* $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ *be a deterministic solitary output three-party functionality. Assume that* $|\mathcal{X}|, |\mathcal{Y}|, |\mathcal{Z}| = \text{poly}(\kappa)$*, and that for all sufficiently large* κ*, there exists* $\chi \in \mathcal{X}$ *such that for all* $x \in \mathcal{X}$ *it holds that* $\chi \preceq x$.[10] *Then* f *can be computed with 1-security if and only if it is* $\text{CORE}_\wedge$*-forced.*

The negative direction directly follows from Theorem 3.6. We prove both Theorem 3.7 and the positive direction of Theorem 3.9 in Section 5.

The next proposition reduces the randomized case to the deterministic case. We stress that the reduction holds for general domain sizes, and functionalities where every party obtains an output (in fact, the reduction can be easily generalized to the multiparty setting assuming an honest majority).

[10]Note that there may be several minimum inputs, however, the assumption implies that they are all equivalent.

Proposition 3.10 (Reducing randomized functionalities to deterministic functionalities)**.** *Let $f : (\{0,1\}^*)^3 \mapsto \{0,1\}^*$ be a (randomized) three-party functionality. Define the deterministic functionality $f' : (\{0,1\}^*)^2 \times (\{0,1\}^*)^2 \times (\{0,1\}^*)^2 \mapsto \{0,1\}^*$ as follows.*

$$f'((x, r_1), (y, r_2), (z, r_3)) = f(x, y, z; r_1 \oplus r_2 \oplus r_3).$$

Then f can be computed with 1-security if and only if f' can be computed with 1-security.

Proof. Let us first assume that f' can be computed with 1-security. To compute f (in the $(f', \mathsf{g.o.d.})$-hybrid model), the parties will invoke $(f', \mathsf{g.o.d.})$ with their original inputs x, y, and z, and where $r_1, r_2, r_3 \leftarrow \{0,1\}^*$ are sampled uniformly at random. Security follows directly from the fact that either r_1, r_2, or r_3 are guaranteed to be a uniform random string. Indeed, a simulator for some corrupted party will send to the trusted party T the same input the corrupted party used in the protocol.

We next show that if f can be computed with 1-security, then so is f'. Using Fact 2.11 and the composition theorem, it suffices to present a protocol for f' in the $\{(f, \mathsf{g.o.d.}), (f', \mathsf{fair})\}$-hybrid model.

Protocol 3.11.

Private inputs: party A holds $(x, r_1) \in (\{0,1\}^*)^2$, party B holds $(y, r_2) \in (\{0,1\}^*)^2$, and party C holds $(z, r_3) \in (\{0,1\}^*)^2$.

Common input: the parties hold the security parameter 1^κ.

1. *The parties invoke (f', fair) with their inputs. Let w_1, w_2, and w_2 be the outputs of A, B, and C, respectively.*
2. *If $w_1, w_2, w_3 \neq \bot$ then A outputs w_1, B outputs w_2, and C outputs w_3.*
3. *Otherwise, the parties invoke $(f, \mathsf{g.o.d.})$ on their inputs x, y, and z, and output the result.*

We next show that the protocol is secure. Consider an adversary $\mathcal{A}$ corrupting A. The other cases are analogous. The simulator $\mathsf{Sim}_{\mathcal{A}}$ will first query $\mathcal{A}$ to receive its input (x', r_1') to (f', fair).

- If $(x', r_1') \neq \mathsf{abort}$, then $\mathsf{Sim}_{\mathcal{A}}$ sends (x', r_1') to the trusted party.
- Otherwise, the adversary $\mathcal{A}$ chooses an input $x'' \in \{0,1\}^*$ to send to $(f, \mathsf{g.o.d.})$.[11] The simulator samples $r_1^* \leftarrow \{0,1\}^*$ and sends (x'', r_1^*) to the trusted party.

In both cases, $\mathsf{Sim}_{\mathcal{A}}$ forwards the output w_1 received from the trusted party to $\mathcal{A}$, outputs whatever $\mathcal{A}$ outputs, and halts.

Clearly, if $\mathcal{A}$ does not abort during the invocation of (f', fair), then its joint view and the output of the honest parties is $f(x', y, z; r_1' \oplus r_2 \oplus r_3)$ in both worlds. Observe that if $\mathcal{A}$ does abort, however, then the output in both worlds is distributed as $f(x'', y, z)$. $\square$

[11] If $\mathcal{A}$ sends an invalid value or does not send any value, the simulator sets x'' to be the default value used by the ideal functionality of f.

3.2.1 Interesting Corollaries

Although our necessary and sufficient conditions do not coincide in general, for various interesting families of functionalities the results do form a characterization. In the following section, we consider several such interesting families, and present a characterization for them, as can be derived from Theorems 3.6, 3.7 and 3.9.

We first state the characterization for functionalities with at most three possible outputs. For this class of functionalities, we make the observation that for every $x \in \mathcal{X}$, either $y \equiv_x y'$ for all $y, y' \in \mathcal{Y}$, or $z \equiv_x z'$ for all $z, z' \in \mathcal{Z}$.

Corollary 3.12 (Characterization of ternary-output functionalities)**.** *Let $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \{0, 1, 2\}$ be a deterministic solitary output three-party functionality. Assume that oblivious transfer exists and that $|\mathcal{X}|, |\mathcal{Y}|, |\mathcal{Z}| = \mathrm{poly}(\kappa)$. Then f can be computed with 1-security if and only if the following hold.*

1. *For all sufficiently large $\kappa \in \mathbb{N}$, and all χ_{B} and χ_{C} that are* B*-minimal and* C*-minimal, respectively, there exists an* R*-minimal $\chi \in \mathcal{X}$ such that $\chi_{\mathsf{B}} \equiv_{\mathsf{B}} \chi \equiv_{\mathsf{C}} \chi_{\mathsf{C}}$.*

2. *f is* CORE$_\wedge$*-forced.*

3. *There exists an ensemble of efficiently samplable distributions $\mathcal{P} = \{P_{\kappa,x}\}_{\kappa \in \mathbb{N}, x \in \mathcal{X}}$ and a* PPT *algorithm* S *such that*

$$\{\mathsf{S}\left(1^\kappa, x, x^*, f(x^*, y, z)\right)\}_{\kappa \in \mathbb{N}, x_\kappa \in \mathcal{X}, i \in [n_\wedge], j \in [m_\wedge], y \in \mathcal{Y}_i^\wedge, z \in \mathcal{Z}_j^\wedge} \stackrel{\mathrm{S}}{\equiv} \{f(x, y^*, z)\}_{\kappa \in \mathbb{N}, x_\kappa \in \mathcal{X}, i \in [n_\wedge], j \in [m_\wedge], y \in \mathcal{Y}_i^\wedge, z \in \mathcal{Z}_j^\wedge},$$

where $x^* \leftarrow P_{\kappa,x}$ and $y^* \leftarrow Q_{\kappa,i}$, where $Q_{\kappa,i}$ is the distribution given the CORE$_\wedge$*-forced property.*

Proof. It suffices to show that Item 1 from the above statement implies Item 1 from Theorem 3.7. That is, we show that for all sufficiently large κ, either $y \equiv_x y'$ for all $x \in \mathcal{X}$ and $y, y' \in \mathcal{Y}$, or $z \equiv_x z'$ for all $x \in \mathcal{X}$ and $z, z' \in \mathcal{Z}$. Assume towards contradiction that for infinitely many κ's, there exists $x, x' \in \mathcal{X}$, $y, y' \in \mathcal{Y}$, and $z, z' \in \mathcal{Z}$ such that $y \not\equiv_x y'$ and $z \not\equiv_{x'} z'$. Now, observe that as f is a ternary-output functionality, it holds that x and x' are B-minimal and C-minimal, respectively. Moreover, it holds that $z \equiv_x z'$ and that $y \equiv_{x'} y'$. By (the assumed) Item 1 there exists an R-minimal $\chi \in \mathcal{X}$ satisfying $x \equiv_{\mathsf{B}} \chi \equiv_{\mathsf{C}} x'$. However, such χ cannot exists since it satisfies $y \equiv_\chi y'$ and $z \equiv_\chi z'$. ☐

We now state a characterization for functionalities that are symmetric with respect to the inputs of B and C, i.e., where $f(x, y, z) = f(x, z, y)$ for all x, y, and z. Here, the characterization follows from the observation all y's are equivalent and z's are equivalent with respect to all x's. In particular, the CORE$_\wedge$-forced property implies the simpler forced property (i.e., both B and C can fix the distribution of the output).

Corollary 3.13 (Characterization of (B, C)-symmetric functionalities)**.** *Let $f : \mathcal{X} \times \mathcal{D} \times \mathcal{D} \mapsto \mathcal{W}$ be a deterministic solitary output three-party functionality. Assume that oblivious transfer exists, that $|\mathcal{X}|, |\mathcal{D}| = \mathrm{poly}(\kappa)$, and that for all sufficiently large $\kappa \in \mathbb{N}$, for all $x \in \mathcal{X}$ and for all $y, z \in \mathcal{D}$ it holds that $f(x, y, z) = f(x, z, y)$. Then f can be computed with 1-security if and only if it is forced.*

We next state a characterization for the case where the input of party A is a single bit. The proof follows from the observation that for such functionalities there exist a minimum χ.

Corollary 3.14. *Let $f : \{0,1\} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ be a deterministic solitary output three-party functionality. Assume that $|\mathcal{Y}|, |\mathcal{Z}| = \operatorname{poly}(\kappa)$. Then f can be computed with 1-security if and only if the following hold.*

1. *For all sufficiently large $\kappa \in \mathbb{N}$, either $0 \preceq 1$ or $1 \preceq 0$.*

2. *f is $\mathrm{CORE}_\wedge$-forced.*

Proof. First observe that if $0 \preceq 1$ or $1 \preceq 0$ for all sufficiently large $\kappa \in \mathbb{N}$, then f can be computed due to Theorem 3.9. For the other direction, we consider two cases. First, if f is not $\mathrm{CORE}_\wedge$-forced then by Theorem 3.6 it cannot be computed with 1-security. Otherwise, if $0 \npreceq 1$ and $1 \npreceq 0$ infinitely often, then both are R-minimal inputs infinitely often. However, there is no R-minimal χ such that $0 \equiv_{\mathsf{B}} \chi \equiv_{\mathsf{C}} 1$. Therefore, f cannot be computed due to Theorem 3.6. $\square$

If A has no input, then the first property of Corollary 3.14 holds vacuously. Thus we have the following.

Corollary 3.15 (Characterization of NIORP functionalities)**.** *Let $f : \emptyset \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ be a deterministic solitary output three-party functionality. Assume that $|\mathcal{Y}|, |\mathcal{Z}| = \operatorname{poly}(\kappa)$. Then f can be computed with 1-security if and only if it is $\mathrm{CORE}_\wedge$-forced.*

4 Impossibility Results

In this section, we prove the necessary conditions stated in Theorem 3.6. Our proof is split into two parts. In the first part, presented in Section 4.1, we apply the hexagon argument over the secure protocol assumed to exist. This results in 6 ensembles of distributions, all of which are statistically close. The second part of the proof, presented in Section 4.2, is dedicated to the analysis of these 6 ensembles. Specifically, we show how the assumption that the ensembles are close implies the necessary conditions stated in Theorem 3.6.

4.1 The Hexagon Argument

In this section we present the hexagon argument, that is the first step in the proof of Theorem 3.6. For a fixed a three-party protocol $\pi = (\mathsf{A}, \mathsf{B}, \mathsf{C})$ that is defined over secure point-to-point channels in the plain model (without a broadcast channel or trusted setup assumptions), we can associate a six-party protocol denoted $\mathsf{Hex}(\pi) = (\mathsf{B}, \mathsf{A}, \mathsf{C}, \mathsf{B}', \mathsf{A}', \mathsf{C}')$ as illustrated in Figure 1. Formally, $\mathsf{Hex}(\pi)$ is defined as follows.

Definition 4.1 (The hexagon protocol)**.** *Given a three-party protocol $\pi = (\mathsf{A}, \mathsf{B}, \mathsf{C})$ we denote by $\mathsf{Hex}(\pi) = (\mathsf{B}, \mathsf{A}, \mathsf{C}, \mathsf{B}', \mathsf{A}', \mathsf{C}')$ the following six-party protocol. Parties A and A' are set with the code of A from π, parties B and B' with the code of B from π, and parties C and C' with the code of C from π.*

The communication network of $\mathsf{Hex}(\pi)$ is a cycle. Party A is connected to C, which is connected to B', which is connected to A', which is connected to C', which is connected to B, which is connected to A.

The following lemma states that any attacker corrupting any four adjacent parties in the six-party protocol $\mathsf{Hex}(\pi)$, can be perfectly emulated by an adversary corrupting a single party in three-party π.

Lemma 4.2 (Mapping attackers for $\mathsf{Hex}(\pi)$ to attackers for π)**.** *Let $\pi = (\mathsf{A}, \mathsf{B}, \mathsf{C})$ be a three-party protocol and let $\mathsf{Hex}(\pi) = (\mathsf{B}, \mathsf{A}, \mathsf{C}, \mathsf{B}', \mathsf{A}', \mathsf{C}')$ be as in Definition 4.1. In the following, for possible inputs (x, x', y, y', z, z') for protocol $\mathsf{Hex}(\pi)$ we let $\mathbf{h} = (x, x', y, y', z, z')$. Then the following hold.*

1. *For every non-uniform* PPT *adversary $\mathcal{A}_{\mathsf{H}}^{\mathsf{B},\mathsf{C}'}$ corrupting $\{\mathsf{A}, \mathsf{B}, \mathsf{C}', \mathsf{A}'\}$ in $\mathsf{Hex}(\pi)$, there exists a non-uniform* PPT *adversary $\mathcal{A}$ corrupting A in π, receiving the inputs y, z', and x' for B, C', and A', respectively, as auxiliary information, that perfectly emulates $\mathcal{A}_{\mathsf{H}}^{\mathsf{B},\mathsf{C}'}$, namely*

$$\left\{\mathrm{REAL}_{\pi,\mathcal{A}(y,z',x',\mathsf{aux})}\left(\kappa,(x,y',z)\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}} \equiv \left\{\mathrm{REAL}_{\mathsf{Hex}(\pi),\mathcal{A}_{\mathsf{H}}^{\mathsf{B},\mathsf{C}'}(\mathsf{aux})}\left(\kappa,\mathbf{h}\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}}.$$

2. *For every non-uniform* PPT *adversary $\mathcal{A}_{\mathsf{H}}^{\mathsf{B}',\mathsf{C}}$ corrupting $\{\mathsf{A}', \mathsf{B}', \mathsf{C}, \mathsf{A}\}$ in $\mathsf{Hex}(\pi)$, there exists a non-uniform* PPT *adversary $\mathcal{A}'$ corrupting A in π, receiving the inputs y', z, and x for B', C, and A, respectively, as auxiliary information, that perfectly emulates $\mathcal{A}_{\mathsf{H}}^{\mathsf{B}',\mathsf{C}}$, namely*

$$\left\{\mathrm{REAL}_{\pi,\mathcal{A}'(y',z,x,\mathsf{aux})}\left(\kappa,(x',y,z')\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}} \equiv \left\{\mathrm{REAL}_{\mathsf{Hex}(\pi),\mathcal{A}_{\mathsf{H}}^{\mathsf{B}',\mathsf{C}}(\mathsf{aux})}\left(\kappa,\mathbf{h}\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}}.$$

3. *For every non-uniform* PPT *adversary $\mathcal{B}_{\mathsf{H}}^{\mathsf{A},\mathsf{C}}$ corrupting $\{\mathsf{B}, \mathsf{A}, \mathsf{C}, \mathsf{B}'\}$ in $\mathsf{Hex}(\pi)$, there exists a non-uniform* PPT *adversary $\mathcal{B}$ corrupting B in π, receiving the inputs x, z, and y' for A, C, and B', respectively, as auxiliary information, that perfectly emulates $\mathcal{B}_{\mathsf{H}}^{\mathsf{A},\mathsf{C}}$, namely*

$$\left\{\mathrm{REAL}_{\pi,\mathcal{B}(x,z,y',\mathsf{aux})}\left(\kappa,(x',y,z')\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}} \equiv \left\{\mathrm{REAL}_{\mathsf{Hex}(\pi),\mathcal{B}_{\mathsf{H}}^{\mathsf{A},\mathsf{C}}(\mathsf{aux})}\left(\kappa,\mathbf{h}\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}}.$$

4. *For every non-uniform* PPT *adversary $\mathcal{B}_{\mathsf{H}}^{\mathsf{A}',\mathsf{C}'}$ corrupting $\{\mathsf{B}', \mathsf{A}', \mathsf{C}', \mathsf{B}\}$ in $\mathsf{Hex}(\pi)$, there exists a non-uniform* PPT *adversary $\mathcal{B}'$ corrupting B in π, receiving the inputs x', z', and y for A', C', and B, respectively, as auxiliary information, that perfectly emulates $\mathcal{B}_{\mathsf{H}}^{\mathsf{A}',\mathsf{C}'}$, namely*

$$\left\{\mathrm{REAL}_{\pi,\mathcal{B}'(x',z',y,\mathsf{aux})}\left(\kappa,(x,y',z)\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}} \equiv \left\{\mathrm{REAL}_{\mathsf{Hex}(\pi),\mathcal{B}_{\mathsf{H}}^{\mathsf{A}',\mathsf{C}'}(\mathsf{aux})}\left(\kappa,\mathbf{h}\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}}.$$

5. *For every non-uniform* PPT *adversary $\mathcal{C}_{\mathsf{H}}^{\mathsf{A},\mathsf{B}}$ corrupting $\{\mathsf{C}', \mathsf{B}, \mathsf{A}, \mathsf{C}\}$ in $\mathsf{Hex}(\pi)$, there exists a non-uniform* PPT *adversary $\mathcal{C}$ corrupting C in π, receiving the inputs y, x, and z for A, B, and C, respectively, as auxiliary information, that perfectly emulates $\mathcal{C}_{\mathsf{H}}^{\mathsf{A},\mathsf{B}}$, namely*

$$\left\{\mathrm{REAL}_{\pi,\mathcal{C}(y,x,z,\mathsf{aux})}\left(\kappa,(x',y',z')\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}} \equiv \left\{\mathrm{REAL}_{\mathsf{Hex}(\pi),\mathcal{C}_{\mathsf{H}}^{\mathsf{A},\mathsf{B}}(\mathsf{aux})}\left(\kappa,\mathbf{h}\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}}.$$

6. *For every non-uniform* PPT *adversary $\mathcal{C}_{\mathsf{H}}^{\mathsf{A}',\mathsf{B}'}$ corrupting $\{\mathsf{C}, \mathsf{B}', \mathsf{A}', \mathsf{C}'\}$ in $\mathsf{Hex}(\pi)$, there exists a non-uniform* PPT *adversary $\mathcal{C}'$ corrupting C in π, receiving the inputs y', x', and z' for A, B, and C, respectively, as auxiliary information, that perfectly emulates $\mathcal{C}_{\mathsf{H}}^{\mathsf{A}',\mathsf{B}'}$, namely*

$$\left\{\mathrm{REAL}_{\pi,\mathcal{C}'(y',x',z',\mathsf{aux})}\left(\kappa,(x,y,z)\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}} \equiv \left\{\mathrm{REAL}_{\mathsf{Hex}(\pi),\mathcal{C}_{\mathsf{H}}^{\mathsf{A}',\mathsf{B}'}(\mathsf{aux})}\left(\kappa,\mathbf{h}\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}}.$$

Proof. We will prove only Item 1 as the rest follows from a similar argument. Fix an adversary $\mathcal{A}_{\mathsf{H}}^{\mathsf{B},\mathsf{C}'}$ corrupting $\{\mathsf{A},\mathsf{B},\mathsf{C}',\mathsf{A}'\}$ in $\mathsf{Hex}(\pi)$. Define an adversary $\mathcal{A}$ corrupting A in π as follows. First, it initializes $\mathcal{A}_{\mathsf{H}}^{\mathsf{B},\mathsf{C}'}$ with input x for A, input y for B, input z' for C', input x' for A', and auxiliary information aux. Each round, it passes to $\mathcal{A}_{\mathsf{H}}^{\mathsf{B},\mathsf{C}'}$ the messages received from the honest parties B and C, and replies to them as $\mathcal{A}_{\mathsf{H}}^{\mathsf{B},\mathsf{C}'}$ does. Finally, $\mathcal{A}$ output whatever $\mathcal{A}_{\mathsf{H}}^{\mathsf{B},\mathsf{C}'}$ outputs.

By the definition of $\mathcal{A}$, in each round, the messages it receives from and sends to B and C in π, are identically distributed to the messages $\mathcal{A}_{\mathsf{H}}^{\mathsf{B},\mathsf{C}'}$ received from and sent to B' and C in $\mathsf{Hex}(\pi)$. Therefore the transcript in both executions are identically distributed. In particular, the joint distribution of the view of the adversary and the output of the honest parties are identical in both executions. ☐

An important use-case of the above lemma is for 1-secure protocols π computing some 3-party functionality f. Here, any attacker in π that emulates some attacker for $\mathsf{Hex}(\pi)$ as given by Lemma 4.2, can be simulated in the ideal world of f. Thus, we get the following corollary.

Corollary 4.3 (Mapping attackers for $\mathsf{Hex}(\pi)$ to simulators for f)**.** *Let $\pi = (\mathsf{A},\mathsf{B},\mathsf{C})$ be a three-party protocol computing some solitary output three-party functionality $f : (\{0,1\}^*)^3 \mapsto \{0,1\}^*$ with 1-security. Then the following hold.*

1. *For every non-uniform* PPT *adversary $\mathcal{A}_{\mathsf{H}}^{\mathsf{B},\mathsf{C}'}$ corrupting $\{\mathsf{A},\mathsf{B},\mathsf{C}',\mathsf{A}'\}$ in $\mathsf{Hex}(\pi)$, there exists a non-uniform* PPT *simulator $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{B},\mathsf{C}'}$ in the ideal world of f corrupting A, such that*
$$\left\{\text{IDEAL}_{f,\mathsf{Sim}_{\mathcal{A}}^{\mathsf{B},\mathsf{C}'}(y,z',x',\mathsf{aux})}\left(\kappa,(x,y',z)\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}} \stackrel{\text{C}}{\equiv} \left\{\text{REAL}_{\mathsf{Hex}(\pi),\mathcal{A}_{\mathsf{H}}^{\mathsf{B},\mathsf{C}'}(\mathsf{aux})}\left(\kappa,\mathbf{h}\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}}.$$

2. *For every non-uniform* PPT *adversary $\mathcal{A}_{\mathsf{H}}^{\mathsf{B}',\mathsf{C}}$ corrupting $\{\mathsf{A}',\mathsf{B}',\mathsf{C},\mathsf{A}\}$ in $\mathsf{Hex}(\pi)$, there exists a non-uniform* PPT *simulator $\mathsf{Sim}_{\mathcal{A}}^{\mathsf{B}',\mathsf{C}}$ in the ideal world of f corrupting A, such that*
$$\left\{\text{IDEAL}_{\pi,\mathsf{Sim}_{\mathcal{A}}^{\mathsf{B}',\mathsf{C}}(y',z,x,\mathsf{aux})}\left(\kappa,(x',y,z')\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}} \stackrel{\text{C}}{\equiv} \left\{\text{REAL}_{\mathsf{Hex}(\pi),\mathcal{A}_{\mathsf{H}}^{\mathsf{B}',\mathsf{C}}(\mathsf{aux})}\left(\kappa,\mathbf{h}\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}}.$$

3. *For every non-uniform* PPT *adversary $\mathcal{B}_{\mathsf{H}}^{\mathsf{A},\mathsf{C}}$ corrupting $\{\mathsf{B},\mathsf{A},\mathsf{C},\mathsf{B}'\}$ in $\mathsf{Hex}(\pi)$, there exists a non-uniform* PPT *simulator $\mathsf{Sim}_{\mathcal{B}}^{\mathsf{A},\mathsf{C}}$ in the ideal world of f corrupting B, such that*
$$\left\{\text{IDEAL}_{\pi,\mathsf{Sim}_{\mathcal{B}}^{\mathsf{A},\mathsf{C}}(x,z,y',\mathsf{aux})}\left(\kappa,(x',y,z')\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}} \stackrel{\text{C}}{\equiv} \left\{\text{REAL}_{\mathsf{Hex}(\pi),\mathcal{B}_{\mathsf{H}}^{\mathsf{A},\mathsf{C}}(\mathsf{aux})}\left(\kappa,\mathbf{h}\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}}.$$

4. *For every non-uniform* PPT *adversary $\mathcal{B}_{\mathsf{H}}^{\mathsf{A}',\mathsf{C}'}$ corrupting $\{\mathsf{B}',\mathsf{A}',\mathsf{C}',\mathsf{B}\}$ in $\mathsf{Hex}(\pi)$, there exists a non-uniform* PPT *simulator $\mathsf{Sim}_{\mathcal{B}}^{\mathsf{A}',\mathsf{C}'}$ in the ideal world of f corrupting B, such that*
$$\left\{\text{IDEAL}_{\pi,\mathsf{Sim}_{\mathcal{B}}^{\mathsf{A}',\mathsf{C}'}(x',z',y,\mathsf{aux})}\left(\kappa,(x,y',z)\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}} \stackrel{\text{C}}{\equiv} \left\{\text{REAL}_{\mathsf{Hex}(\pi),\mathcal{B}_{\mathsf{H}}^{\mathsf{A}',\mathsf{C}'}(\mathsf{aux})}\left(\kappa,\mathbf{h}\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}}.$$

5. *For every non-uniform* PPT *adversary $\mathcal{C}_{\mathsf{H}}^{\mathsf{A},\mathsf{B}}$ corrupting $\{\mathsf{C}',\mathsf{B},\mathsf{A},\mathsf{C}\}$ in $\mathsf{Hex}(\pi)$, there exists a non-uniform* PPT *simulator $\mathsf{Sim}_{\mathcal{C}}^{\mathsf{A},\mathsf{B}}$ in the ideal world of f corrupting C, such that*
$$\left\{\text{IDEAL}_{\pi,\mathsf{Sim}_{\mathcal{C}}^{\mathsf{A},\mathsf{B}}(y,x,z,\mathsf{aux})}\left(\kappa,(x',y',z')\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}} \stackrel{\text{C}}{\equiv} \left\{\text{REAL}_{\mathsf{Hex}(\pi),\mathcal{C}_{\mathsf{H}}^{\mathsf{A},\mathsf{B}}(\mathsf{aux})}\left(\kappa,\mathbf{h}\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}}.$$

6. *For every non-uniform* PPT *adversary* $\mathcal{C}_{\mathsf{H}}^{\mathsf{A}',\mathsf{B}'}$ *corrupting* $\{\mathsf{C}, \mathsf{B}', \mathsf{A}', \mathsf{C}'\}$ *in* $\mathsf{Hex}(\pi)$*, there exists a non-uniform* PPT *simulator* $\mathsf{Sim}_{\mathcal{C}}^{\mathsf{A}',\mathsf{B}'}$ *in the ideal world of* f *corrupting* C*, such that*

$$\left\{\mathrm{IDEAL}_{\pi,\mathsf{Sim}_{\mathcal{C}}^{\mathsf{A}',\mathsf{B}'}(y',x',z',\mathsf{aux})}\left(\kappa,(x,y,z)\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}} \stackrel{\mathrm{C}}{\equiv} \left\{\mathrm{REAL}_{\mathsf{Hex}(\pi),\mathcal{C}_{\mathsf{H}}^{\mathsf{A}',\mathsf{B}'}(\mathsf{aux})}\left(\kappa,\mathbf{h}\right)\right\}_{\kappa,\mathbf{h},\mathsf{aux}}.$$

One important use-case of Corollary 4.3 is when the six adversaries for $\mathsf{Hex}(\pi)$ are semi-honest. This is due to the fact that the views of the honest parties are identically distributed in all six cases, hence the same holds with respect to their outputs. Next, consider the joint distribution of the outputs of A and A' in $\mathsf{Hex}(\pi)$. Observe that for any adversary corrupting either of them, say A, its simulator given by Corollary 4.3 must be able to generate the output of A, as it is part of the view. Furthermore, if either A or A' is honest, then the simulator can force the output of A in the ideal world of f to be indistinguishable from the real world.

Now, recall that these simulators are for the *malicious* setting, hence they can send arbitrary inputs to the trusted party. Thus, the distributions over the outputs depend on the distribution over the input sent by each simulator to the trusted party. Notice that when considering semi-honest adversaries for $\mathsf{Hex}(\pi)$ that have no auxiliary input, these distributions depend only on the security parameter and the inputs given to the semi-honest adversary.

For example, in the case where $\{\mathsf{B}, \mathsf{A}, \mathsf{C}, \mathsf{B}'\}$ are corrupted, the simulator samples a random input y^* according to some distribution Q that depends only on the security parameter κ, and the inputs y, x, z, and y' given to the adversary. The input y^* must be such that the joint output of the simulator and the output of A in the ideal world of f, must be indistinguishable from the joint output of A and A' in $\mathsf{Hex}(\pi)$.

Lemma 4.4. *Let* $f : (\{0,1\}^*)^3 \mapsto \{0,1\}^*$ *be a solitary output three-party functionality that can be computed with 1-security. Then the there exists*

- *two ensembles of efficiently samplable distributions*

  $$\mathcal{P}^{\mathsf{B},\mathsf{C}'} = \{P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y,z',x'}\}_{\kappa\in\mathbb{N},x,x',y,z'\in\{0,1\}^*} \quad \textit{and} \quad \mathcal{P}^{\mathsf{B}',\mathsf{C}} = \{P^{\mathsf{B}',\mathsf{C}}_{\kappa,x',y',z,x}\}_{\kappa\in\mathbb{N},x,x',y,z'\in\{0,1\}^*}$$

  over $\mathcal{X}$*,*

- *two ensembles of efficiently samplable distributions*

  $$\mathcal{Q} = \{Q_{\kappa,y',z,x,y}\}_{\kappa\in\mathbb{N},x,y,y',z\in\{0,1\}^*} \quad \textit{and} \quad \mathcal{Q}' = \left\{Q'_{\kappa,y,z',x',y'}\right\}_{\kappa\in\mathbb{N},x,y,y',z\in\{0,1\}^*}$$

  over $\mathcal{Y}$*,*

- *two ensembles of efficiently samplable distributions*

  $$\mathcal{R} = \{R_{\kappa,z,x,y,z'}\}_{\kappa\in\mathbb{N},x,y,z,z'\in\{0,1\}^*} \quad \textit{and} \quad \mathcal{R}' = \left\{R'_{\kappa,z',x',y',z}\right\}_{\kappa\in\mathbb{N},x,y,z,z'\in\{0,1\}^*}$$

  over $\mathcal{Z}$*,*

- *and six* PPT *algorithms* $\mathsf{S}^{\mathsf{B},\mathsf{C}'}$*,* $\mathsf{S}^{\mathsf{B}',\mathsf{C}}$*,* S_{B}*,* S'_{B}*,* S_{C}*, and* S'_{C}*,*

such that the following six distribution ensembles are computationally indistinguishable

1. $\{\mathsf{S}^{\mathsf{B},\mathsf{C}'}(x, y, z', x', x_1^*, f(x_1^*, y', z))\}_{\kappa,x,x',y,y',z,z'}$, *where* $x_1^* \leftarrow P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y,z',x'}$.

2. $\{\mathsf{S}^{\mathsf{B}',\mathsf{C}}(x', y', z, x, x_2^*, f(x_2^*, y, z'))\}_{\kappa,x,x',y,y',z,z'}$, *where* $x_2^* \leftarrow P^{\mathsf{B}',\mathsf{C}}_{\kappa,x',y',z,x}$.

3. $\{(\mathsf{S}_{\mathsf{B}}(y', z, x, y, y_1^*), f(x', y_1^*, z'))\}_{\kappa,x,x',y,y',z,z'}$, *where* $y_1^* \leftarrow Q_{\kappa,y',z,x,y}$.

4. $\{(f(x, y_2^*, z), \mathsf{S}'_{\mathsf{B}}(y, z', x', y', y_2^*))\}_{\kappa,x,x',y,y',z,z'}$, *where* $y_2^* \leftarrow Q'_{\kappa,y,z',x',y'}$.

5. $\{(\mathsf{S}_{\mathsf{C}}(z, x, y, z', z_1^*), f(x', y', z_1^*))\}_{\kappa,x,x',y,y',z,z'}$, *where* $z_1^* \leftarrow R_{\kappa,z,x,y,z'}$.

6. $\{(f(x, y, z_2^*), \mathsf{S}'_{\mathsf{C}}(z', x', y', z, z_2^*))\}_{\kappa,x,x',y,y',z,z'}$, *where* $z_2^* \leftarrow R'_{\kappa,z',x',y',z}$.

Moreover, if the domain of f is of polynomial size in κ, then the above ensembles are statistically close.

Proof. let π be a three-party protocol computing f with 1-security, and consider an honest execution of $\mathsf{Hex}(\pi)$. Let $(\textsc{out}(\kappa, \mathbf{h}), \textsc{out}'(\kappa, \mathbf{h}))$ denote the joint distribution of the outputs of A and A', respectively, in such execution of $\mathsf{Hex}(\pi)$, where $\mathbf{h} = (x, x', y, y', z, z')$ are the inputs of the parties. We will show how obtain each of the Ensembles 1–6, such that each of them is computationally indistinguishable from (OUT,OUT').

We first show how to obtain Ensemble 1. Ensemble 2 can be obtained using a similar argument. Consider the semi-honest adversary $\mathcal{A}^{\mathsf{B},\mathsf{C}'}_{\mathsf{H}}$ corrupting $\{\mathsf{A}, \mathsf{B}, \mathsf{C}', \mathsf{A}'\}$ with no additional auxiliary information, that outputs the output of A and A' (note that this is well-defined since the adversary is semi-honest). By Item 1 from Corollary 4.3, there exists a non-uniform PPT simulator $\mathsf{Sim}^{\mathsf{B},\mathsf{C}'}_{\mathcal{A}}$ in the ideal world of f corrupting A, such that

$$\left\{\text{IDEAL}_{f,\mathsf{Sim}^{\mathsf{B},\mathsf{C}'}_{\mathcal{A}}(y,z',x')}\left(\kappa, (x, y', z)\right)\right\}_{\kappa,\mathbf{h}} \stackrel{\text{C}}{\equiv} \left\{\text{REAL}_{\mathsf{Hex}(\pi),\mathcal{A}^{\mathsf{B},\mathsf{C}'}_{\mathsf{H}}}(\kappa, \mathbf{h})\right\}_{\kappa,\mathbf{h}}.$$

Since only A receives an output in the idea world of f, it follows that

$$\begin{aligned}\left\{\text{VIEW}^{\mathsf{ideal}}_{f,\mathsf{Sim}^{\mathsf{B},\mathsf{C}'}_{\mathcal{A}}(y,z',x')}\left(\kappa, (x, y', z)\right)\right\}_{\kappa,\mathbf{h}} &\equiv \left\{\text{IDEAL}_{f,\mathsf{Sim}^{\mathsf{B},\mathsf{C}'}_{\mathcal{A}}(y,z',x')}\left(\kappa, (x, y', z)\right)\right\}_{\kappa,\mathbf{h}} \\ &\stackrel{\text{C}}{\equiv} \left\{\text{VIEW}^{\mathsf{real}}_{\mathsf{Hex}(\pi),\mathcal{A}^{\mathsf{B},\mathsf{C}'}_{\mathsf{H}}}(\kappa, \mathbf{h})\right\}_{\kappa,\mathbf{h}} \\ &\equiv \left\{(\textsc{out}, \textsc{out}')\right\}_{\kappa,\mathbf{h}},\end{aligned}$$

where $\mathbf{h} = (x, x', y, y', z, z')$. We let $P_{\kappa,x,y,z',x'}$ denote the distribution over the input x^* that $\mathsf{Sim}^{\mathsf{B},\mathsf{C}'}_{\mathcal{A}}$ sends to the trusted party T, and let $\mathsf{S}^{\mathsf{B},\mathsf{C}'}(x, y, z', x', x_1^*, w)$ output whatever $\mathsf{Sim}^{\mathsf{B},\mathsf{C}'}_{\mathcal{A}}$ outputs given that it sent x_1^* to T and received the output w. Therefore,

$$\begin{aligned}\left\{\mathsf{S}^{\mathsf{B},\mathsf{C}'}(x, y, z', x', x_1^*, f(x_1^*, y', z))\right\}_{\kappa,\mathbf{h}} &\equiv \left\{\text{VIEW}^{\mathsf{ideal}}_{f,\mathsf{Sim}^{\mathsf{B},\mathsf{C}'}_{\mathcal{A}}(y,z',x')}\left(\kappa, (x, y', z)\right)\right\}_{\kappa,\mathbf{h}} \\ &\stackrel{\text{C}}{\equiv} \left\{(\textsc{out}(\kappa, \mathbf{h}), \textsc{out}'(\kappa, \mathbf{h}))\right\}_{\kappa,\mathbf{h}},\end{aligned}$$

where $x_1^* \leftarrow P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y,z',x'}$.

We now show how to obtain Ensemble 3. The rest of the ensembles can be obtained using a similar argument. Similarly to the previous case, we consider the semi-honest adversary $\mathcal{B}^{\mathsf{A},\mathsf{C}}_{\mathsf{H}}$

corrupting $\{\mathsf{B}, \mathsf{A}, \mathsf{C}, \mathsf{B}'\}$ with no additional auxiliary information, that outputs the output of A. By Item 3 from Corollary 4.3, there exists a non-uniform PPT simulator $\mathsf{Sim}_{\mathcal{B}}^{\mathsf{A},\mathsf{C}}$ in the ideal world of f corrupting B, such that

$$\left\{\text{IDEAL}_{\pi,\mathsf{Sim}_{\mathcal{B}}^{\mathsf{A},\mathsf{C}}(x,z,y')}\left(\kappa,(x',y,z')\right)\right\}_{\kappa,\mathbf{h}} \stackrel{\mathrm{C}}{\equiv} \left\{\text{REAL}_{\mathsf{Hex}(\pi),\mathcal{B}_{\mathsf{H}}^{\mathsf{A},\mathsf{C}}}\left(\kappa,\mathbf{h}\right)\right\}_{\kappa,\mathbf{h}}.$$

Since only A and A' receives an output in the execution of $\mathsf{Hex}(\pi)$, it follows that

$$\begin{aligned}\left\{\text{REAL}_{\mathsf{Hex}(\pi),\mathcal{B}_{\mathsf{H}}^{\mathsf{A},\mathsf{C}}}\left(\kappa,\mathbf{h}\right)\right\}_{\kappa,\mathbf{h}} &\equiv \left\{\left(\text{VIEW}^{\mathsf{real}}_{\mathsf{Hex}(\pi),\mathcal{B}_{\mathsf{H}}^{\mathsf{A},\mathsf{C}}}\left(\kappa,(x',y,z')\right), \text{OUT}^{\mathsf{ideal}}_{\mathsf{Hex}(\pi),\mathcal{B}_{\mathsf{H}}^{\mathsf{A},\mathsf{C}}}\left(\kappa,(x',y,z')\right)\right)\right\}_{\kappa,\mathbf{h}}\\ &\equiv \left\{\left(\text{OUT}(\kappa,\mathbf{h}),\text{OUT}'(\kappa,\mathbf{h})\right)\right\}_{\kappa,\mathbf{h}}.\end{aligned}$$

Let $Q_{\kappa,y',z,x,y}$ denote the distribution over the input y^* that $\mathsf{Sim}_{\mathcal{B}}^{\mathsf{A},\mathsf{C}}$ sends to the trusted party T, and let $\mathsf{S}_{\mathsf{B}}(y',z,x,y,y_1^*)$ output whatever $\mathsf{Sim}_{\mathcal{B}}^{\mathsf{A},\mathsf{C}}$ outputs given that it sent y_1^* to T. Then

$$\begin{aligned}\left\{\left(\mathsf{S}_{\mathsf{B}}\left(y',z,x,y,y_1^*\right), f(x',y_1^*,z')\right)\right\}_{\kappa,\mathbf{h}} &\equiv \left\{\text{IDEAL}_{\pi,\mathsf{Sim}_{\mathcal{B}}^{\mathsf{A},\mathsf{C}}(x,z,y')}\left(\kappa,(x',y,z')\right)\right\}_{\kappa,\mathbf{h}}\\ &\stackrel{\mathrm{C}}{\equiv} \left\{\text{REAL}_{\mathsf{Hex}(\pi),\mathcal{B}_{\mathsf{H}}^{\mathsf{A},\mathsf{C}}}\left(\kappa,\mathbf{h}\right)\right\}_{\kappa,\mathbf{h}}\\ &\equiv \left\{\left(\text{OUT}(\kappa,\mathbf{h}),\text{OUT}'(\kappa,\mathbf{h})\right)\right\}_{\kappa,\mathbf{h}},\end{aligned}$$

where $y_1^* \leftarrow Q_{\kappa,y',z,x,y}$.

As for the "moreover" part, observe that if the domain of f is of polynomial size, then the support of all ensembles is of polynomial size. Thus, by Fact 2.3 the ensembles are statistically close. □

4.2 Analyzing The Ensembles

In this section we analyze the six distribution ensembles given by Lemma 4.4. For the sake of brevity, throughout the entire section we fix a deterministic solitary output three-party functionality that can be computed with 1-security $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$, where $|\mathcal{X}|, |\mathcal{Y}|, |\mathcal{Z}| = \text{poly}(\kappa)$. Additionally, we fix all distribution ensembles and PPT algorithms from Lemma 4.4, using the same notations.

It will be convenient in the proof to use the following notion of statistical independence. Roughly, a distribution ensemble is statistically independent of one of its variables, if changing the variable results in a statistically close distribution ensemble.

Definition 4.5 (Statistical independence)**.** *Let $X = \{X_{a,b,n}\}_{a\in\mathcal{D}_n, b\in\mathcal{D}'_n, n\in\mathbb{N}}$ be a distribution ensemble. We say that X is* statistically independent *of $\{\mathcal{D}'_n\}_{n\in\mathbb{N}}$ if*

$$\{X_{a,b,n}\}_{a\in\mathcal{D}_n, b,b'\in\mathcal{D}'_n, n\in\mathbb{N}} \stackrel{\mathrm{S}}{\equiv} \{X_{a,b',n}\}_{a\in\mathcal{D}_n, b,b'\in\mathcal{D}'_n, n\in\mathbb{N}}.$$

For the sake of simplifying the presentation, we will usually say that X is statistically independent of b, rather than referring to its domain.

Theorem 3.6 follows from the following two claims, stating the conditions specified in it.

Claim 4.6. *For all sufficiently large $\kappa \in \mathbb{N}$, if χ_{C} and χ_{B} are C-minimal and B-minimal, respectively, then there exists an R-minimal $\chi \in \mathcal{X}$ such that $\chi_{\mathsf{C}} \equiv_{\mathsf{C}} \chi \equiv_{\mathsf{B}} \chi_{\mathsf{B}}$.*

Claim 4.7. *For every $\kappa \in \mathbb{N}$ and every $i \in [n_\wedge]$ we let y_i denote the lexicographically smallest element of $\mathcal{Y}_i^\wedge$. Similarly, for $j \in [m_\wedge]$ we let z_j denote the lexicographically smallest element of $\mathcal{Z}_j^\wedge$. Then there exists two ensembles of efficiently samplable distributions $\mathcal{Q} = \{Q_{\kappa,i}\}_{\kappa\in\mathbb{N}, i\in[n_\wedge]}$ and $\mathcal{R} = \{R_{\kappa,j}\}_{\kappa\in\mathbb{N}, j\in[m_\wedge]}$ over $\mathcal{Y}$ and $\mathcal{Z}$, respectively, such that the following holds.*

$$\{f(x,y^*,z_j)\}_{\kappa,x,i,j,y,z} \overset{\mathrm{S}}{\equiv} \{f(x,y^*,z)\}_{\kappa,x,i,j,y,z} \overset{\mathrm{S}}{\equiv} \{f(x,y,z^*)\}_{\kappa,x,i,j,y,z} \overset{\mathrm{S}}{\equiv} \{f(x,y_i,z^*)\}_{\kappa,x,i,j,y,z} \quad (6)$$

where $y^* \leftarrow Q_{\kappa,i}$, $z^* \leftarrow R_{\kappa,j}$.

Moreover, suppose that f has the property that for all sufficiently large κ, it holds that either $y \equiv_x y'$ for all $x \in \mathcal{X}$ and $y, y' \in \mathcal{Y}$, or $z \equiv_x z'$ for all $x \in \mathcal{X}$ and $z, z' \in \mathcal{Z}$. Then there exists an ensemble of efficiently samplable distributions $\mathcal{P} = \{P_{\kappa,x}\}_{\kappa\in\mathbb{N}, x\in\mathcal{X}}$ and a PPT *algorithm S such that*

$$\{\mathsf{S}(1^\kappa, x, x^*, f(x^*,y,z))\}_{\kappa,x,i,j,y,z} \overset{\mathrm{S}}{\equiv} \{f(x,y^*,z)\}_{\kappa,x,i,j,y,z},$$

where $x^* \leftarrow P_{\kappa,x}$ and $y^* \leftarrow Q_{\kappa,i}$, where $Q_{\kappa,i}$ is the distribution given the $\mathrm{CORE}_\wedge$*-forced property.*

We prove Claims 4.6 and 4.7 below. We first make the following simple yet useful observation, which states that each of the marginal distributions of the ensembles are statistically independent of several of the inputs.

Claim 4.8. *Consider the* PPT *algorithms $\mathsf{S}^{\mathsf{B},\mathsf{C}'}$ and $\mathsf{S}^{\mathsf{B}',\mathsf{C}}$ from Lemma 4.4, and write them as $\mathsf{S}^{\mathsf{B},\mathsf{C}'} = (\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}, \mathsf{S}_2^{\mathsf{B},\mathsf{C}'})$ and $\mathsf{S}^{\mathsf{B}',\mathsf{C}} = (\mathsf{S}_1^{\mathsf{B}',\mathsf{C}}, \mathsf{S}_2^{\mathsf{B}',\mathsf{C}})$. Then both $\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}$ and $\mathsf{S}_1^{\mathsf{B}',\mathsf{C}}$ are statistically independent of x', y', and z'. Similarly, both $\mathsf{S}_2^{\mathsf{B},\mathsf{C}'}$ and $\mathsf{S}_2^{\mathsf{B}',\mathsf{C}}$ are statistically independent of x, y, and z.*

Proof. We prove that $\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}$ is statistically independent of x', y', and z'The second statement can proven using an analogous argument. Observe that by Lemma 4.4, it follows that

$$\begin{aligned}\{\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}(x,y,z',x',x^*,f(x^*,y',z))\}_{\kappa,x,x',y,y',z,z'} &\overset{\mathrm{S}}{\equiv} \{\mathsf{S}_\mathsf{B}(y',z,x,y,y^*)\}_{\kappa,x,x',y,y',z,z'} \\ &\overset{\mathrm{S}}{\equiv} \{\mathsf{S}_\mathsf{C}(z,x,y,z',z^*)\}_{\kappa,x,x',y,y',z,z'},\end{aligned}$$

where $y^* \leftarrow Q_{\kappa,y',z,x,y}$ and $z^* \leftarrow R_{\kappa,z,x,y,z'}$. As S_B and S_C are statistically independent of x', z' and x', y', respectively, it follows that $\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}$ is statistically independent of them as well. $\square$

The following two lemmata are the main ingredients in our proof. The first lemma roughly identifies the support of the inputs x_1^* and x_2^* used by PPT algorithms $\mathsf{S}^{\mathsf{B},\mathsf{C}'}$ and $\mathsf{S}^{\mathsf{B}',\mathsf{C}}$ (up to negligible probability). The second lemma identifies when it is possible to change some of the inputs, such that the at least one of the marginal distributions of the outcome of the PPT algorithms $\mathsf{S}^{\mathsf{B},\mathsf{C}'}$ and $\mathsf{S}^{\mathsf{B}',\mathsf{C}}$ remains similar.

Lemma 4.9. *Consider the distribution ensembles $\mathcal{P}^{\mathsf{B},\mathsf{C}'}$ and $\mathcal{P}^{\mathsf{B}',\mathsf{C}}$ from Lemma 4.4. Then for every $x, x' \in \mathcal{X}$, every $y \in \mathcal{Y}$, and every $z' \in \mathcal{Z}$, it holds that*

$$\Pr_{x_1^* \leftarrow P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y,z',x'}}\left[x_1^* \not\preceq_\mathsf{C} x \vee x_1^* \not\preceq_\mathsf{B} x'\right] = \mathrm{neg}(\kappa).$$

Similarly, for every $x, x' \in \mathcal{X}$, every $y' \in \mathcal{Y}$, and every $z \in \mathcal{Z}$, it holds that

$$\Pr_{x_2^* \leftarrow P^{\mathsf{B}',\mathsf{C}}_{\kappa,x',y',z,x}}\left[x_2^* \not\preceq_\mathsf{B} x \vee x_2^* \not\preceq_\mathsf{C} x'\right] = \mathrm{neg}(\kappa).$$

In particular, for all sufficiently large κ and every $x, x' \in \mathcal{X}$, there exists $x^* \in \mathcal{X}$ such that

$$x^* \preceq_\mathsf{C} x \wedge x^* \preceq_\mathsf{B} x'.$$

Lemma 4.10. *The following hold.*

1.

$$\left\{\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x_1^*,f(x_1^*,y',z_1)\right)\right\}_{\kappa,x,x',y,y',z_1,z_2,z'} \stackrel{\mathrm{S}}{\equiv} \left\{\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x_1^*,f(x_1^*,y',z_2)\right)\right\}_{\kappa,x,x',y,y',z_1,z_2,z'},$$

where $z_1 \equiv_{\tilde{x}} z_2$ *for all* $\tilde{x} \preceq_{\mathsf{C}} x$*, and where* $x_1^* \leftarrow P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y,z',x'}$.

2.

$$\left\{\mathsf{S}_2^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x_1^*,f(x_1^*,y_1',z)\right)\right\}_{\kappa,x,x',y,y_1',y_2',z,z'} \stackrel{\mathrm{S}}{\equiv} \left\{\mathsf{S}_2^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x_1^*,f(x_1^*,y_2',z)\right)\right\}_{\kappa,x,x',y,y_1',y_2',z,z'},$$

where $y_1' \equiv_{\tilde{x}} y_2'$ *for all* $\tilde{x} \preceq_{\mathsf{C}} x$*, and where* $x_1^* \leftarrow P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y,z',x'}$.

3.

$$\left\{\mathsf{S}_1^{\mathsf{B}',\mathsf{C}}\left(x',y',z,x,x_2^*,f(x_2^*,y,z_1')\right)\right\}_{\kappa,x,x',y,y',z,z_1',z_2'} \stackrel{\mathrm{S}}{\equiv} \left\{\mathsf{S}_1^{\mathsf{B}',\mathsf{C}}\left(x',y',z,x,x_2^*,f(x_2^*,y,z_2')\right)\right\}_{\kappa,x,x',y,y',z,z_1',z_2'},$$

where $z_1' \equiv_{\tilde{x}} z_2'$ *for all* $\tilde{x} \preceq_{\mathsf{C}} x$*, and where* $x_2^* \leftarrow P^{\mathsf{B}',\mathsf{C}}_{\kappa,x',y',z,x}$.

4.

$$\left\{\mathsf{S}_2^{\mathsf{B}',\mathsf{C}}\left(x',y',z,x,x_2^*,f(x_2^*,y_1,z')\right)\right\}_{\kappa,x,x',y_1,y_2,y',z,z'} \stackrel{\mathrm{S}}{\equiv} \left\{\mathsf{S}_2^{\mathsf{B}',\mathsf{C}}\left(x',y',z,x,x_2^*,f(x_2^*,y_2,z')\right)\right\}_{\kappa,x,x',y_1,y_2,y',z,z'},$$

where $y_1 \equiv_{\tilde{x}} y_2$ *for all* $\tilde{x} \preceq_{\mathsf{C}} x$*, and where* $x_2^* \leftarrow P^{\mathsf{B}',\mathsf{C}}_{\kappa,x',y',z,x}$.

Lemmas 4.9 and 4.10 are proved in Sections 4.2.1 and 4.2.2, respectively. Before providing the proofs, we first show that they imply Claims 4.6 and 4.7, and thus they imply Theorem 3.6.

Proof of Claim 4.6. Let χ_{B} and χ_{C} be B-minimal and C-minimal, respectively. We assume without loss of generality that $\chi_{\mathsf{B}} \not\equiv \chi_{\mathsf{C}}$, as otherwise the claim is trivial. By Lemma 4.9 there exists $\chi \in \mathcal{X}$ satisfying $\chi \preceq_{\mathsf{B}} \chi_{\mathsf{B}}$ and $\chi \preceq_{\mathsf{C}} \chi_{\mathsf{C}}$. By the minimality of χ_{B} and χ_{C} it follows that $\chi \equiv_{\mathsf{B}} \chi_{\mathsf{B}}$ and $\chi \equiv_{\mathsf{C}} \chi_{\mathsf{C}}$. It is left to show that χ is R-minimal. Let $\tilde{\chi} \preceq \chi$. By Lemma 4.9 there exists $\tilde{x}$ satisfying $\tilde{x} \preceq_{\mathsf{B}} \tilde{\chi} \preceq_{\mathsf{B}} \chi \equiv_{\mathsf{B}} \chi_{\mathsf{B}}$ and $\tilde{x} \preceq_{\mathsf{C}} \tilde{\chi} \preceq_{\mathsf{C}} \chi \equiv_{\mathsf{C}} \chi_{\mathsf{C}}$. By the minimality of χ_{B} and χ_{C} it follows that $\chi_{\mathsf{B}} \equiv_{\mathsf{B}} \tilde{x} \equiv_{\mathsf{C}} \chi_{\mathsf{C}}$. Therefore $\tilde{x} \equiv_{\mathsf{B}} \chi$ and $\tilde{x} \equiv_{\mathsf{C}} \chi$, hence $\tilde{x} \equiv \chi$. ☐

Proof of Claim 4.7. We first define the distributions $Q_{\kappa,i}$ and $R_{\kappa,j}$, for $i \in [n_\wedge]$ and $j \in [m_\wedge]$. Let $\mathcal{Q}'$ and $\mathcal{R}'$ be the distribution ensembles from Lemma 4.4. In the following, we fix x_0, y_0, and z_0 to be the lexicographically smallest elements of $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$, respectively. We let $Q_{\kappa,i}$ be the distribution $Q'_{\kappa,y_i,z_0,x_0,y_0}$ and let $R_{\kappa,j}$ be the distribution $R'_{\kappa,z_0,x_0,y_0,z_j}$.

We now prove Equation (6). The second transition follows from Lemma 4.4. We prove the first transition. The last transition can be proved using an analogous argument. Let $\mathsf{S}^{\mathsf{B},\mathsf{C}'}$ be as in Lemma 4.4, and let $\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}$ be the first entry in its output. First, observe that if $z \equiv_\wedge z_j$, then $z \equiv_\chi z_j$ for all R-minimal χ. The minimality of all such χ implies that $z \equiv_{\tilde{x}} z_j$ for all $\tilde{x} \preceq_{\mathsf{C}} x$. Second, by Lemma 4.9 $x^* \preceq_{\mathsf{C}} x$ with probability at least $1 - \mathrm{neg}(\kappa)$. Thus, combining with Lemma 4.10 it follows that

$$\left\{\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x^*,f(x^*,y',z)\right)\right\}_{\kappa,j,x,x',y,y',z,z'} \stackrel{\mathrm{S}}{\equiv} \left\{\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x^*,f(x^*,y',z_j)\right)\right\}_{\kappa,j,x,x',y,y',z,z'}, \tag{7}$$

where $x^* \leftarrow P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y,z',x'}$. Furthermore, by Claim 4.8 the above ensembles are statistically independent of x', y', and z', thus the ensembles are statistically close for fixed $x' = x_0$, $y' = y_0$, and $z' = z_0$, i.e., it holds that

$$\left\{\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x, y, z_0, x_0, x^*, f(x^*, y_0, z)\right)\right\}_{\kappa,j,x,y,z} \stackrel{\text{S}}{\equiv} \left\{\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x, y, z_0, x_0, x^*, f(x^*, y_0, z_j)\right)\right\}_{\kappa,j,x,y,z}.$$

Combined with Lemma 4.4 this implies that

$$\begin{aligned}
\left\{f(x, y^*, z)\right\}_{\kappa,j,x,y,z} &\stackrel{\text{S}}{\equiv} \left\{\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x, y, z_0, x_0, x^*, f(x^*, y_0, z)\right)\right\}_{\kappa,j,x,y,z} \\
&\stackrel{\text{S}}{\equiv} \left\{\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x, y, z_0, x_0, x^*, f(x^*, y_0, z_j)\right)\right\}_{\kappa,j,x,y,z} \\
&\stackrel{\text{S}}{\equiv} \left\{f(x, y^*, z_j)\right\}_{\kappa,j,x,y,z},
\end{aligned}$$

where $x^* \leftarrow P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y,z_0,x_0}$ and $y^* \leftarrow Q'_{\kappa,y,z_0,x_0,y_0}$. Finally, observe that this implies that

$$\left\{f(x, y^*, z_j)\right\}_{\kappa,x,i,j,y,z} \stackrel{\text{S}}{\equiv} \left\{f(x, y^*, z)\right\}_{\kappa,x,i,j,y,z},$$

where $y^* \leftarrow Q'_{\kappa,y_i,z_0,x_0,y_0} \equiv Q_{\kappa,i}$.

We now prove the "moreover" part of the claim. Let $\mathcal{K}_{\mathsf{B}} \subseteq \mathbb{N}$ be the set of all $\kappa \in \mathbb{N}$ such that $y \equiv_x y'$ for all $x \in \mathcal{X}$ and $y, y' \in \mathcal{Y}$, and let $\mathcal{K}_{\mathsf{C}} \subseteq \mathbb{N}$ be the set of all $\kappa \in \mathbb{N}$ such that $z \equiv_x z'$ for all $x \in \mathcal{X}$ and $z, z' \in \mathcal{Z}$. For every $\kappa \in \mathcal{K}_{\mathsf{B}}$, we define the distribution $P_{\kappa,x}$ as $P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y_0,z_0,x_0}$ and let $\mathsf{S}(1^\kappa, x, x^*, w)$ output $\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}(x, y_0, z_0, x_0, x^*, w)$. Similarly, for any $\kappa \in \mathcal{K}_{\mathsf{C}}$ we define the distribution $P_{\kappa,x}$ as $P^{\mathsf{B}',\mathsf{C}}_{\kappa,x_0,y_0,z_0,x}$ and let $\mathsf{S}(1^\kappa, x, x^*, w)$ output $\mathsf{S}_1^{\mathsf{B}',\mathsf{C}}(x_0, y_0, z_0, x, x^*, w)$. Observe that since $|\mathcal{X}|, |\mathcal{Y}|, |\mathcal{Z}| = \operatorname{poly}(\kappa)$ identifying whether $\kappa \in \mathcal{K}_{\mathsf{B}}$ or $\kappa \in \mathcal{K}_{\mathsf{C}}$ can be done efficiently.

To conclude the proof, we now show that

$$\left\{\mathsf{S}\left(1^\kappa, x, x^*, f(x^*, y, z)\right)\right\}_{\kappa,x,i,j,y,z} \stackrel{\text{S}}{\equiv} \left\{f(x, y^*, z)\right\}_{\kappa,x,i,j,y,z}, \tag{8}$$

where $x^* \leftarrow P_{\kappa,x}$ and $y^* \leftarrow Q_{\kappa,i}$. Assume for the sake of contradiction that Equation (8) is false. Then the ensembles have statistical distance of at least $1/\operatorname{poly}(\kappa)$, for infinitely many $\kappa \in \mathbb{N}$. By assumption, $\kappa \in \mathcal{K}_{\mathsf{B}} \cup \mathcal{K}_{\mathsf{C}}$ for all sufficiently large κ, hence the distance of $1/\operatorname{poly}(\kappa)$ holds for infinitely many $\kappa \in \mathcal{K}_{\mathsf{B}} \cup \mathcal{K}_{\mathsf{C}}$. We assume without loss of generality that all such infinitely many κ belong to $\mathcal{K}_{\mathsf{B}}$. However, by the definition of $P_{\kappa,x}$ and the PPT algorithm S, it holds that

$$\left\{\mathsf{S}\left(1^\kappa, x, x^*, f(x^*, y, z)\right)\right\}_{\kappa \in \mathcal{K}_{\mathsf{B}},x,i,j,y,z} \equiv \left\{\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x, y_0, z_0, x_0, x^*, f(x^*, y, z)\right)\right\}_{\kappa \in \mathcal{K}_{\mathsf{B}},x,i,j,y,z},$$

where $x^* \leftarrow P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y_0,z_0,x_0}$. Thus,

$$\left\{\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x, y_0, z_0, x_0, x^*, f(x^*, y, z)\right)\right\}_{\kappa \in \mathcal{K}_{\mathsf{B}},x,i,j,y,z} \stackrel{\text{S}}{\not\equiv} \left\{f(x, y^*, z)\right\}_{\kappa \in \mathcal{K}_{\mathsf{B}},x,i,j,y,z},$$

which contradicts Lemma 4.4. □

4.2.1 Proof of Lemma 4.9

Proof of Lemma 4.9. We prove the first part of the claim. The second part follows from an analogous argument. For brevity, we write x^* instead of x_1^*. Assume for the sake of contradiction that there exists $x, x' \in \mathcal{X}$, $y \in \mathcal{Y}$, $z' \in \mathcal{Z}$, and a polynomial p, such that for infinitely many κ's it holds that

$$\Pr_{x^* \leftarrow P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y,z',x'}} \left[x^* \not\geq_{\mathsf{C}} x \vee x^* \not\geq_{\mathsf{B}} x'\right] \geq 1/p(\kappa).$$

Then by the union bound it follows that for infinitely many κ's either

$$\Pr_{x^* \leftarrow P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y,z',x'}} \left[x^* \not\geq_{\mathsf{C}} x\right] \geq 1/2p(\kappa)$$

or

$$\Pr_{x^* \leftarrow P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y,z',x'}} \left[x^* \not\geq_{\mathsf{B}} x'\right] \geq 1/2p(\kappa).$$

Assume the former without loss of generality. We next show that Ensembles 1 and 4 are statistically far, that is, we show that

$$\left\{\mathsf{S}^{\mathsf{B},\mathsf{C}'}\left(x, y, z', x', x^*, f(x^*, y', z)\right)\right\}_{\kappa,x,x',y,y',z,z'} \overset{\mathrm{S}}{\not\equiv} \left\{\left(f(x, y^*, z), \mathsf{S}'_{\mathsf{B}}\left(y, z', x', y', y^*\right)\right)\right\}_{\kappa,x,x',y,y',z,z'}, \quad (9)$$

where $x^* \leftarrow P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y,z',x'}$ and $y^* \leftarrow Q'_{\kappa,y,z',x',y'}$, thus contradicting Lemma 4.4.

By Fact 2.4 it suffices to show a distinguisher. The distinguisher D will simply consider the first entry and infer the equivalence class of z. Formally, let $\mathsf{CClass}_x(w)$ output the unique $j \in [m(x)]$ such that $w \in f(x, \mathcal{Y}, \mathcal{Z}^x_j)$. Observe that CClass_x can be computed in polynomial time since $|\mathcal{Y}|$ and $|\mathcal{Z}|$ are polynomials. Then, given an output $w \in \mathcal{W}$ in the first entry, our distinguisher D outputs 1 if $\mathsf{CClass}_x(w) = j$, where $z \in \mathcal{Z}^x_j$, and outputs 0 otherwise. Clearly, given the output from the ensemble on the right-hand side of Equation (9), D outputs 1 with probability 1. We next analyze the probability of $\mathsf{S}^{\mathsf{B}',\mathsf{C}}$ outputting a value w' satisfying $\mathsf{CClass}_x(w') = j$, and show that it is significantly far from 1, thus proving that D has a noticeable distinguishing advantage.

Intuitively, if $x^* \not\geq_{\mathsf{C}} x$ then $\mathsf{S}^{\mathsf{B},\mathsf{C}'}$ lacks information about the equivalence class of z with respect to the input x. Therefore it will have to guess it. In the following, we abuse notations and for $z \in \mathcal{Z}$ we let $\mathsf{CClass}_x(z)$ output the value $j \in [m(x)]$ satisfying $z \in \mathcal{Z}^x_j$. Let $\mathsf{S}^{\mathsf{B},\mathsf{C}'}_1$ be the first entry in the output of $\mathsf{S}^{\mathsf{B},\mathsf{C}'}$. The next formalize the above intuition. First observe that by the union bound, for each of the infinitely many κ's considered there exists $\tilde{x} \in \mathcal{X}$ satisfying $\tilde{x} \not\geq_{\mathsf{C}} x$, such that

$$\Pr_{x^* \leftarrow P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y,z',x'}} \left[x^* = \tilde{x}\right] \geq \frac{1}{2p(\kappa) \cdot |\mathcal{X}|}.$$

We let $z_1, z_2 \in \mathcal{Z}$ satisfy $z_1 \equiv_{\tilde{x}} z_2$ and $z_1 \not\equiv_x z_2$. The next claim roughly states that for $\mathsf{S}^{\mathsf{B},\mathsf{C}'}_1$, changing from $z = z_1$ to $z = z_2$ will not change its output with noticeable probability.

Claim 4.11. *For all $x' \in \mathcal{X}$, $y, y' \in \mathcal{Y}$, and $z' \in \mathcal{Z}$, it holds that*

$$\Pr\left[\mathsf{S}^{\mathsf{B},\mathsf{C}'}_1\left(x, y, z', x', x^*, f(x^*, y', z_1)\right) = \mathsf{S}^{\mathsf{B},\mathsf{C}'}_1\left(x, y, z', x', x^*, f(x^*, y', z_2)\right)\right] \geq \frac{1}{2p(\kappa) \cdot |\mathcal{X}|} - \operatorname{neg}(\kappa),$$

where the probability is taken over the random coins of $\mathsf{S}^{\mathsf{B},\mathsf{C}'}_1$, and where $x^* \leftarrow P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y,z',x'}$.

The claim is proven below. We first use it to show how D can distinguish with non-negligible probability. Consider the case where $z \leftarrow \{z_1, z_2\}$ is sampled uniformly at random and $x^* \leftarrow P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y,z',x'}$, where both are sampled independently. We denote

$$q := \Pr\left[\mathsf{S}^{\mathsf{B},\mathsf{C}'}_1\left(x, y, z', x', x^*_1, f(x^*_1, y', z_1)\right) = \mathsf{S}^{\mathsf{B},\mathsf{C}'}_1\left(x, y, z', x', x^*_2, f(x^*_2, y', z_2)\right)\right].$$

Then

$$\begin{aligned}
&\Pr\left[\mathsf{CClass}_x\left(\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x^*,f(x^*,y',z)\right)\right)=\mathsf{CClass}_x\left(z\right)\right]\\
&=\frac{1}{2}\cdot\Pr\left[\mathsf{CClass}_x\left(\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x^*,f(x^*,y',z_1)\right)\right)=\mathsf{CClass}_x\left(z_1\right)\right]\\
&+\frac{1}{2}\cdot\Pr\left[\mathsf{CClass}_x\left(\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x^*,f(x^*,y',z_2)\right)\right)=\mathsf{CClass}_x\left(z_2\right)\right]\\
&\le\frac{1}{2}\cdot\left(q\cdot\Pr\left[\mathsf{CClass}_x\left(\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x^*,f(x^*,y',z_2)\right)\right)=\mathsf{CClass}_x\left(z_1\right)\right]+1-q\right)\\
&+\frac{1}{2}\cdot\Pr\left[\mathsf{CClass}_x\left(\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x^*,f(x^*,y',z_2)\right)\right)=\mathsf{CClass}_x\left(z_2\right)\right].
\end{aligned}$$

Now, let

$$a:=\Pr\left[\mathsf{CClass}_x\left(\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x^*,f(x^*,y',z_2)\right)\right)=\mathsf{CClass}_x\left(z_1\right)\right],$$

and let

$$b:=\Pr\left[\mathsf{CClass}_x\left(\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x^*,f(x^*,y',z_2)\right)\right)=\mathsf{CClass}_x\left(z_2\right)\right].$$

Then $a+b\le 1$. Therefore

$$\begin{aligned}
\Pr\left[\mathsf{CClass}_x\left(\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x^*,f(x^*,y',z)\right)\right)=\mathsf{CClass}_x\left(z\right)\right]&\le\frac{1}{2}\cdot(aq+1-q)+\frac{1}{2}\cdot b\\
&\le\frac{1}{2}\cdot(aq+1-q)+\frac{1}{2}-\frac{1}{2}\cdot a\\
&=\frac{1}{2}\cdot(1-q)\,(1-a)+\frac{1}{2}\\
&\le 1-\frac{1}{2}\cdot q\\
&\le 1-\frac{1}{2}\cdot\left(\frac{1}{2p(\kappa)\cdot|\mathcal{X}|}-\mathrm{neg}(\kappa)\right),
\end{aligned}$$

where the last inequality follows from Claim 4.11. Since we assume $|\mathcal{X}|$ to be polynomial in κ, it follows that D has a noticeable distinguishing advantage. □

Proof of Claim 4.11. Since $z_1\equiv_{\tilde{x}} z_2$ there exists $z_{i_1},\ldots,z_{i_k}\in\mathcal{Z}$ such that

$$z_1\sim_{\tilde{x}} z_{i_1}\sim_{\tilde{x}}\ldots\sim_{\tilde{x}} z_{i_k}\sim_{\tilde{x}} z_2,$$

where $k=k(\kappa)$. For convenience, we let $z_{i_0}:=z_1$ and $z_{i_{k+1}}:=z_2$. This implies the existence of $y_{i_j},y'_{i_j}\in\mathcal{Y}$ for every $j\in\{0,\ldots,k+1\}$, such that the following hold.

1. $f(\tilde{x},y',z_{i_0})=f(\tilde{x},y_{i_0},z_{i_0})$.
2. For all $j\in\{0,\ldots,k\}$ it holds that $f(\tilde{x},y'_{i_j},z_{i_j})=f(\tilde{x},y_{i_{j+1}},z_{i_{j+1}})$.
3. $f(\tilde{x},y'_{i_{k+1}},z_{i_{k+1}})=f(\tilde{x},y',z_{i_{k+1}})$.

Now, observe that the event

$$\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x^*,f(x^*,y',z_{i_0})\right)=\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x^*,f(x^*,y',z_{i_{k+1}})\right)$$

is implied by the conjunction of the following four events:

1. $\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}(x,y,z',x',x^*,f(x^*,y',z_{i_0})) = \mathsf{S}_1^{\mathsf{B},\mathsf{C}'}(x,y,z',x',x^*,f(x^*,y_{i_0},z_{i_0}))$.
2. For all $j \in \{0,\ldots,k\}$ it holds that
$$\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x^*,f(x^*,y'_{i_j},z_{i_j})\right) = \mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x^*,f(x^*,y_{i_{j+1}},z_{i_{j+1}})\right).$$
3. For all $j \in [k]$ it holds that
$$\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x^*,f(x^*,y_{i_j},z_{i_j})\right) = \mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x^*,f(x^*,y'_{i_j},z_{i_j})\right).$$
4. $\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}\left(x,y,z',x',x^*,f(x^*,y'_{i_{k+1}},z_{i_{k+1}})\right) = \mathsf{S}_1^{\mathsf{B},\mathsf{C}'}(x,y,z',x',x^*,f(x^*,y',z_{i_{k+1}}))$.

Furthermore, observe that Event 2 is implied by the event $x^* = \tilde{x}$. Let E be the event that Events 1, 3, and 4 occur. Then

$$\Pr\left[\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}(x,y,z',x',x^*,f(x^*,y',z_{i_0})) = \mathsf{S}_1^{\mathsf{B},\mathsf{C}'}(x,y,z',x',x^*,f(x^*,y',z_{i_{k+1}}))\right] \geq \Pr[x^* = \tilde{x} \mid E] \cdot \Pr[E]$$

Additionally, by Claim 4.8 it follows that $\Pr[E] \geq 1 - \mathrm{neg}(\kappa)$, hence

$$\frac{1}{2p(\kappa)\cdot|\mathcal{X}|} \leq \Pr[x^* = \tilde{x}] \leq \Pr[x^* = \tilde{x} \mid E]\cdot\Pr[E] + \mathrm{neg}(\kappa).$$

Therefore

$$\Pr\left[\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}(x,y,z',x',x^*,f(x^*,y',z_{i_0})) = \mathsf{S}_1^{\mathsf{B},\mathsf{C}'}(x,y,z',x',x^*,f(x^*,y',z_{i_{k+1}}))\right] \geq \frac{1}{2p(\kappa)\cdot|\mathcal{X}|} - \mathrm{neg}(\kappa),$$

as claimed. □

4.2.2 Proof of Lemma 4.10

Proof of Lemma 4.10. We prove only the first item. The rest can be proved using a similar argument. We also write x^* instead of x_1^* for the sake fo brevity. Assume towards contradiction that the claim is false. Then by Fact 2.4 there exists a distinguisher D such that for infinitely many κ, there exists $x,x' \in \mathcal{X}$, $y,y' \in \mathcal{Y}$, and $z_1,z_2,z' \in \mathcal{Z}$, satisfying

$$\Pr\left[\mathsf{D}\left(\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}(x,y,z',x',x^*,f(x^*,y',z_1))\right) = 1\right] - \Pr\left[\mathsf{D}\left(\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}(x,y,z',x',x^*,f(x^*,y',z_2))\right) = 1\right] \geq \frac{1}{\mathrm{poly}(\kappa)}.$$

To alleviate notations, we will write $\mathsf{S}(x,y,x^*,w)$ instead of $\mathsf{S}_1^{\mathsf{B},\mathsf{C}'}(x,y,z',x',x^*,w)$. First, we claim that there exists $\tilde{x} \in \mathcal{X}$, such that $x^* = \tilde{x}$ occurs with noticeable probability, and D distinguishes the two ensembles for fixed $x^* = \tilde{x}$. That is, it holds that

$$\Pr[\mathsf{D}(\mathsf{S}(x,y,\tilde{x},f(\tilde{x},y',z_1))) = 1] - \Pr[\mathsf{D}(\mathsf{S}(x,y,\tilde{x},f(\tilde{x},y',z_2))) = 1] \geq \frac{1}{\mathrm{poly}(\kappa)}.$$

Indeed,

$$\begin{aligned}
\frac{1}{\mathrm{poly}(\kappa)} &\leq \Pr[\mathsf{D}(\mathsf{S}(x,y,x^*,f(x^*,y',z_1))) = 1] - \Pr[\mathsf{D}(\mathsf{S}(x,y,x^*,f(x^*,y',z_2))) = 1] \\
&= \sum_{\tilde{x}\in\mathcal{X}} \Pr[x^* = \tilde{x}] \cdot (\Pr[\mathsf{D}(\mathsf{S}(x,y,\tilde{x},f(\tilde{x},y',z_1))) = 1] - \Pr[\mathsf{D}(\mathsf{S}(x,y,\tilde{x},f(\tilde{x},y',z_2))) = 1]) \\
&\leq |\mathcal{X}| \cdot \max_{\tilde{x}\in\mathcal{X}} \left\{\Pr[x^* = \tilde{x}] \cdot (\Pr[\mathsf{D}(\mathsf{S}(x,y,\tilde{x},f(\tilde{x},y',z_1))) = 1] - \Pr[\mathsf{D}(\mathsf{S}(x,y,\tilde{x},f(\tilde{x},y',z_2))) = 1])\right\}.
\end{aligned}$$

Since $|\mathcal{X}|$ is polynomial in κ, it follows that such $\tilde{x}$ exists.

Now, let $z_{i_1}, \ldots, z_{i_k} \in \mathcal{Z}$ satisfy

$$z_1 \sim_{\tilde{x}} z_{i_1} \sim_{\tilde{x}} \ldots \sim_{\tilde{x}} z_{i_k} \sim_{\tilde{x}} z_2,$$

and denote $z_{i_0} := z_1$ and $z_{i_{k+1}} := z_2$. Since $|\mathcal{Z}| = \mathrm{poly}(\kappa)$, by a hybrid argument there exists $\ell \in \{0, \ldots, k\}$ such that

$$\Pr\left[\mathsf{D}\left(\mathsf{S}\left(x, y, \tilde{x}, f(\tilde{x}, y', z_{i_\ell})\right)\right) = 1\right] - \Pr\left[\mathsf{D}\left(\mathsf{S}\left(x, y, \tilde{x}, f(\tilde{x}, y', z_{i_{\ell+1}})\right)\right) = 1\right] \geq \frac{1}{\mathrm{poly}(\kappa)}.$$

Let $y'' \in \mathcal{Y}$ satisfy $f(\tilde{x}, y', z_{i_\ell}) = f(\tilde{x}, y'', z_{i_{\ell+1}})$. We now show that D can distinguish $\mathsf{S}(x, y, x^*, f(x^*, y', z_{i_{\ell+1}}))$ from $\mathsf{S}(x, y, x^*, f(x^*, y'', z_{i_{\ell+1}}))$, where $x^* \leftarrow P^{\mathsf{B},\mathsf{C}'}_{\kappa,x,y,z',x'}$, thus contradicting Claim 4.8. Indeed,

$$\begin{aligned}
&\Pr\left[\mathsf{D}\left(\mathsf{S}\left(x, y, x^*, f(x^*, y'', z_{i_{\ell+1}})\right)\right) = 1\right] - \Pr\left[\mathsf{D}\left(\mathsf{S}\left(x, y, x^*, f(x^*, y', z_{i_{\ell+1}})\right)\right) = 1\right] \\
&\quad \geq \Pr\left[x^* = \tilde{x}\right] \cdot \left(\Pr\left[\mathsf{D}\left(\mathsf{S}\left(x, y, \tilde{x}, f(\tilde{x}, y'', z_{i_{\ell+1}})\right)\right) = 1\right] - \Pr\left[\mathsf{D}\left(\mathsf{S}\left(x, y, \tilde{x}, f(\tilde{x}, y', z_{i_{\ell+1}})\right)\right) = 1\right]\right) \\
&\quad = \Pr\left[x^* = \tilde{x}\right] \cdot \left(\Pr\left[\mathsf{D}\left(\mathsf{S}\left(x, y, \tilde{x}, f(\tilde{x}, y', z_{i_\ell})\right)\right) = 1\right] - \Pr\left[\mathsf{D}\left(\mathsf{S}\left(x, y, \tilde{x}, f(\tilde{x}, y', z_{i_{\ell+1}})\right)\right) = 1\right]\right) \\
&\quad \geq \frac{1}{\mathrm{poly}(\kappa)}.
\end{aligned}$$

□

5 Positive Results For the Point-to-Point Model

In this section we prove Theorem 3.9 and the positive direction of Theorem 3.7, which give sufficient conditions for a functionality f to be computable with 1-security. We prove Theorem 3.7 in Section 5.1, and prove positive direction of Theorem 3.9 in Section 5.2.

5.1 Proving Theorem 3.7

In this section, we prove Theorem 3.7 by constructing a protocol for any functionality f satisfying the properties given in the theorem. Interestingly, our protocol is a slight variant of the one given by [2], where in case an attack is detected, A and one of the other parties interact in a two-party computation, while ignoring the third party (even if it was honest). In particular, we get a characterization for all functionalities that can be securely computed with such protocol.[12]

In the following section, we let $\mathcal{K}_{\mathsf{B}} \subseteq \mathbb{N}$ be the set of all $\kappa \in \mathbb{N}$ such that $y \equiv_x y'$ for all $x \in \mathcal{X}$ and $y, y' \in \mathcal{Y}$, and let $\mathcal{K}_{\mathsf{C}} \subseteq \mathbb{N}$ be the set of all $\kappa \in \mathbb{N}$ such that $z \equiv_x z'$ for all $x \in \mathcal{X}$ and $z, z' \in \mathcal{Z}$. Recall that we assume that f has the property where $\mathbb{N} \setminus (\mathcal{K}_{\mathsf{B}} \cup \mathcal{K}_{\mathsf{C}})$ is finite. Define the families of sets $\mathcal{D} = \{\mathcal{D}_\kappa\}_{\kappa \in \mathbb{N}}$ as follows: let $\mathcal{D}_\kappa = \mathcal{Z}$ for all $\kappa \in \mathcal{K}_{\mathsf{B}}$ or $\kappa \in \mathbb{N} \setminus (\mathcal{K}_{\mathsf{B}} \cup \mathcal{K}_{\mathsf{C}})$, and let $\mathcal{D}_\kappa = \mathcal{Y}$ for all $\kappa \in \mathcal{K}_{\mathsf{C}}$. The two-party functionality $g : \mathcal{X} \times \mathcal{D}_\kappa \mapsto \mathcal{W}$ is defined as

$$g(x, d) = \begin{cases} f(x, d, z^*) & \text{if } \kappa \in \mathcal{K}_{\mathsf{C}} \\ f(x, y^*, d) & \text{otherwise} \end{cases}$$

[12] The slight variant we use, is that in our protocol, the identity of the party that will interact with A depends on the security parameter κ. To get a characterization as to which functionalities can be securely computed with the protocol of [2] directly, we need to reposition the quantifier over x in the first property from Theorem 3.7: For all sufficiently large κ and for all $x \in \mathcal{X}$, it holds that either $y \equiv_x y'$ for all $y, y' \in \mathcal{Y}$, or $z \equiv_x z'$ for all $z, z' \in \mathcal{Z}$.

where $y^* \leftarrow Q_{\kappa,1}$ and $z^* \leftarrow R_{\kappa,1}$ (recall that in each case, there is only one equivalent class for the inputs of B or C). Observe that since we assume the domain of f to be of polynomial size in κ, it is possible to efficiently verify whether or not $\kappa \in \mathcal{K}_{\mathsf{C}}$, and efficiently compute g.

We now present a protocol for computing f in the $\{(f, \mathsf{fair}), (g, \mathsf{g.o.d.})\}$-hybrid model. By Fact 2.11 (f, fair) can be computed in the plain model, and since g is a solitary output two-party functionality, the result of Kilian [21] states it can be securely computed assuming OT. Thus, Theorem 3.7 follows from the composition theorem.

Protocol 5.1 (π_{ACOS})**.**

Private inputs: A *holds* $x \in \mathcal{X}$, B *holds* $y \in \mathcal{Y}$, *and* C *holds* $z \in \mathcal{Z}$.

Common input: the parties hold the security parameter 1^κ.

1. *The parties invoke* (f, fair) *with their inputs. Let* w_1, w_2, *and* w_2 *be the outputs of* A, B, *and* C, *respectively.*
2. *If* $w_1, w_2, w_3 \neq \perp$ *then* A *outputs* w_1.
3. *Otherwise, if* $\kappa \in \mathcal{K}_{\mathsf{C}}$ *then parties* A *and* B *invoke* $(g, \mathsf{g.o.d.})$ *with their inputs. If* $\kappa \notin \mathcal{K}_{\mathsf{C}}$ *then parties* A *and* C *invoke* $(g, \mathsf{g.o.d.})$ *with their inputs.*
4. *Party* A *outputs whatever it received from* g.

Theorem 3.7 follows from the following lemma, stating the security of π_{ACOS}.

Lemma 5.2. *Let* $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ *be a deterministic solitary output three-party functionality. Assume that oblivious transfer exists,* $|\mathcal{X}|, |\mathcal{Y}|, |\mathcal{Z}| = \text{poly}(\kappa)$, *and that the following hold.*

1. *For all sufficiently large* κ, *either* $y \equiv_x y'$ *for all* $x \in \mathcal{X}$ *and* $y, y' \in \mathcal{Y}$, *or* $z \equiv_x z'$ *for all* $x \in \mathcal{X}$ *and* $z, z' \in \mathcal{Z}$.
2. f *is* $\text{CORE}_\wedge$*-forced.*
3. *There exists an ensemble of efficiently samplable distributions* $\mathcal{P} = \{P_{\kappa,x}\}_{\kappa \in \mathbb{N}, x \in \mathcal{X}}$ *and a* PPT *algorithm* S *such that*
$$\{\mathsf{S}(1^\kappa, x, x^*, f(x^*, y, z))\}_{\kappa,x,i,j,y,z} \stackrel{\text{S}}{\equiv} \{f(x, y^*, z)\}_{\kappa,x,i,j,y,z},$$
where $x^* \leftarrow P_{\kappa,x}$ *and* $y^* \leftarrow Q_{\kappa,i}$, *where* $Q_{\kappa,i}$ *is the distribution given the* $\text{CORE}_\wedge$*-forced property.*

Then π_{ACOS} *computes* f *with statistical 1-security in the* $\{(f, \mathsf{fair}), (g, \mathsf{g.o.d.})\}$*-hybrid model.*

Proof. π_{ACOS} is clearly correct since if all parties are honest, the fair computation of f will never abort. We next show that the protocol is secure against any adversary $\mathcal{B}$ corrupting B. The case of a corrupt C follows from a similar argument. We define the simulator $\mathsf{Sim}_{\mathcal{B}}$ as follows.

1. Query $\mathcal{B}$ for its input y' to (f, fair).

2. If $y' \neq \bot$, then send y' to the trusted party T, output whatever $\mathcal{B}$ outputs, and halt.

3. Otherwise, if $\kappa \in \mathcal{K}_\mathsf{C}$, query $\mathcal{B}$ for its input y'' to $(g, \mathsf{g.o.d.})$. If $\kappa \notin \mathcal{K}_\mathsf{C}$, then set y'' to be a default value.

4. Find the unique value $i \in [n_\wedge]$ such that $y'' \in \mathcal{Y}_i^\wedge$, sample $y^* \leftarrow Q_{\kappa,i}$, and send y^* to the trusted party.

5. Output whatever $\mathcal{B}$ outputs and halt.

Since B receives no messages in the protocol, the inputs y' and y'' chosen by the adversary in the real world, are identically distributed to their ideal world counterparts. Furthermore, it suffices to show that the output of A in both worlds are statistically close. Clearly, given that $y' \neq \bot$ or $\kappa \notin \mathcal{K}_\mathsf{C}$ the output of A in both worlds is identical. Otherwise, the output of A in the real world is $f(x, y'', z^*)$, where $z^* \leftarrow R_{\kappa,1}$ (recall that all z are equivalent with respect to $\equiv_x$). On the other hand, in the ideal world, the output of A is $f(x, y^*, z)$, where $y^* \leftarrow Q_{\kappa,i}$ and $i \in [n_\wedge]$ is such that $y'' \in \mathcal{Y}_i^\wedge$. By the CORE$_\wedge$-forced property of f, the two distributions are statistically close.

We next fix an adversary $\mathcal{A}$ corrupting A. We define the simulator $\mathsf{Sim}_\mathcal{A}$ as follows.

1. Query $\mathcal{A}$ for its input x' to (f, fair).

2. If $x' \neq \bot$, then send x' to the trusted party T, pass the received output to $\mathcal{A}$, output whatever it outputs, and halt.

3. Otherwise, query $\mathcal{A}$ for its input x'' to $(g, \mathsf{g.o.d.})$.

4. Sample $x^* \leftarrow P_{\kappa,x''}$ and send it to the trusted party T.

5. Given an output w from T, send to $\mathcal{A}$ the result of $\mathsf{S}(1^\kappa, x'', x^*, w)$, output whatever $\mathcal{A}$ outputs, and halt.

Since no honest party has an output, it suffices to show that the view of $\mathcal{A}$ in both worlds are statistically close. First, since $\mathcal{A}$ does not receive any message before the invocation of g, it follows that x' and x'' are identically distributed in both worlds. Now, in the real world, the only message that $\mathcal{A}$ receives is $f(x, y, z^*)$ if $\kappa \in \mathcal{K}_\mathsf{C}$ or $f(x, y^*, z)$ if $\kappa \notin \mathcal{K}_\mathsf{C}$, where $y^* \leftarrow Q_{\kappa,1}$ and $z^* \leftarrow R_{\kappa,1}$. In the ideal world, on the other hand, it receives $\mathsf{S}(1^\kappa, x'', x^*, f(x^*, y, z))$, where $x^* \leftarrow P_{\kappa,x''}$. By the assumption on f, this is statistically close to $f(x, y^*, z)$, thus security holds with respect to all $\kappa \notin \mathcal{K}_\mathsf{C}$. For all $\kappa \in \mathcal{K}_\mathsf{C}$, the CORE$_\wedge$-forced property implies that $f(x, y^*, z)$ is statistically close to $f(x, y, z^*)$, concluding the proof. □

5.2 Proving The Positive Direction of Theorem 3.9

In this section we present a protocol for computing the functionalities captured by Theorem 3.9 with 1-security. We first present an intuitive description of the protocol.

Similarly to π_{ACOS}, the parties first compute f fairly and if the computation followed through, then A outputs the result. Otherwise, the parties do the following. Both B and C (locally) compute the equivalence classes of their respective inputs with respect to the lexicographically smallest minimum input χ (i.e., smaller than all $x \in \mathcal{X}$ with respect to $\preceq$). They then send these values to A, who samples their inputs y^* and z^* according to the appropriate distribution given by the $\text{CORE}_\wedge$-forced assumption, and outputs $f(x, y^*, z^*)$.[13]

Intuitively, the only information a corrupt A obtains from the above interaction is the equivalence classes of the inputs of B and C with respect to χ. This can be simulated by sending χ to the trusted party, and search (by brute-force) for the equivalence classes. This can be done, since by the definition of CORE partition, the equivalence classes are fully determined by the output and the input χ.

We next formalize the above intuition. We present the protocol in the $\{(f, \mathsf{fair})\}$-hybrid model. By Fact 2.11 (f, fair) can be computed in the plain model. Thus, Theorem 3.9 follows from the composition theorem. In the following we let χ be the lexicographically smallest minimum input with respect to $\preceq$.

Protocol 5.3 (π)**.**

Private inputs: A *holds* $x \in \mathcal{X}$, B *holds* $y \in \mathcal{Y}$, *and* C *holds* $z \in \mathcal{Z}$.

Common input: the parties hold the security parameter 1^κ.

1. *The parties invoke* (f, fair) *with their inputs. Let* w_1, w_2, *and* w_2 *be the outputs of* A, B, *and* C, *respectively.*
2. *If* $w_1, w_2, w_3 \neq \perp$ *then* A *outputs* w_1.
3. *Otherwise, party* B *finds the (unique) index* $i \in [n(\chi)]$ *such that* $y \in \mathcal{Y}_i^\chi$ *and sends it to* A. *Similarly,* C *sends the index* $j \in [m(\chi)]$ *such that* $z \in \mathcal{Z}_j^\chi$.
4. A *samples and outputs* $w = f(x, y^*, z_j)$, *where* $y^* \leftarrow Q_{\kappa,i}$ *and* $Q_{\kappa,i}$ *is the distribution given by the* $\text{CORE}_\wedge$*-forced property, and where* z_j *is the lexicographically smallest element of* $\mathcal{Z}_j^\chi$.

The next lemma immediately proves Theorem 3.9.

Lemma 5.4. *Let* $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ *be a deterministic solitary output three-party functionality. Assume that* $|\mathcal{X}|, |\mathcal{Y}|, |\mathcal{Z}| = \text{poly}(\kappa)$, *that for all sufficiently large* κ, *there exists* $\chi \in \mathcal{X}$ *such that for all* $x \in \mathcal{X}$ *it holds that* $\chi \preceq x$, *and that* f *is* $\text{CORE}_\wedge$*-forced. Then* π *computes* f *with statistical 1-security in the* (f, fair)*-hybrid model.*

Proof. Clearly, π is correct since if all parties are honest, the fair computation of f will never abort. We next show that the protocol is secure against any adversary $\mathcal{B}$ corrupting B. The case of a corrupt C follows from a similar argument. We define the simulator $\mathsf{Sim}_\mathcal{B}$ as follows.

1. Query $\mathcal{B}$ for its input y' to (f, fair).
2. If $y' \neq \perp$, then send y' to the trusted party T, output whatever $\mathcal{B}$ outputs, and halt.
3. Otherwise, $\mathcal{B}$ sends to A a value $i' \in [n(\chi)]$.
4. Sample $y^* \leftarrow Q_{\kappa,i'}$, send it to T, output whatever $\mathcal{B}$ outputs, and halt.

[13] In the formal description of the protocol below, we let A set one the random inputs to be the lexicographically smallest element in its equivalence class. This is only for the sake presentation and it does not affect the security of the protocol.

Since $\mathcal{B}$ receives no messages, it suffices to show that the outputs of A in both worlds are statistically close. Now, observe that the messages $\mathcal{B}$ sends are identically distributed in both worlds. For the case where $\mathcal{B}$ sends and input $y' \neq \perp$ to (f, fair), the output of A in both worlds is $f(x, y', z)$. Next, assume that $\mathcal{B}$ sends $y' = \perp$ and then sends i'. Then the output of A in the real world is $f(x, y^*, z_j)$, where $y^* \leftarrow Q_{\kappa,i'}$ and where z_j in the lexicographically smallest element in $\mathcal{Z}_j^{\chi}$. In the ideal world, the output of A is $f(x, y^*, z)$, where y^* is distributed as before. By the $\mathrm{CORE}_{\wedge}$-forced property of f, the two distributions are statistically close.

We next consider an adversary $\mathcal{A}$ corrupting A. We define the simulator $\mathsf{Sim}_{\mathcal{A}}$ as follows.

1. Query $\mathcal{A}$ for its input x' to (f, fair).
2. If $x' \neq \perp$, then send x' to the trusted party T, pass the received output to $\mathcal{A}$, output whatever it outputs, and halt.
3. Otherwise, send χ to T.
4. Let w be the output sent by T.
5. Find the (unique) $i \in [n(\chi)]$ and $j \in [m(\chi)]$ such that there exists $y' \in \mathcal{Y}_i^{\chi}$ and $z' \in \mathcal{Z}_j^{\chi}$ satisfying $w = f(x, y', z')$.
6. Send i and j to $\mathcal{A}$, output whatever it outputs, and halt.

Since B and C have no output, it suffices to show that the view of $\mathcal{A}$ in both worlds are close (in fact, they are identically distributed). Now, if $\mathcal{A}$ sent an input $x' \neq \perp$ to (f, fair), then in both worlds the only message that $\mathcal{A}$ sees is $f(x,', y, z)$. Otherwise, it obtains two values i and j representing equivalence classes over $\mathcal{Y}$ and $\mathcal{Z}$, respectively. Observe that the classes are the same in both worlds since they where computed with respect to the minimum input χ. $\square$

6 Computation With Broadcast and a Dishonest Majority

In this section we show that all three-party functionalities captured by our positive results from the previous section, i.e., Theorems 3.7 and 3.9, can be computed given a broadcast channel, tolerating two corruptions. In fact, we can even relax some of the requirements. Both of our results (stated below) improve the results of Halevi et al. [20], who identify several classes of functionalities that can be securely computed.

We next state our two results. We state the results only for deterministic functionalities, as the randomized case can be handled with using a standard reduction. The first result states that a generalized class of functionalities of those captured by Theorem 3.7, can be computed with full security.

Theorem 6.1. *Let $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ be a deterministic solitary output three-party functionality. Assume that oblivious transfer exists, that $|\mathcal{X}|, |\mathcal{Y}|, |\mathcal{Z}| = \mathrm{poly}(\kappa)$, and that one of the following holds.*

1. *For all sufficiently large κ, either $y \equiv_x y'$ for all $x \in \mathcal{X}$ and $y, y' \in \mathcal{Y}$, or $z \equiv_x z'$ for all $x \in \mathcal{X}$ and $z, z' \in \mathcal{Z}$.*
2. *f is $\mathrm{CORE}_{\wedge}$-forced.*

Then f can be computed with full security.

The proof is given in Section 6.1.1. We next state our second result, which states that a generalized class of functionalities of those captured by Theorem 3.9, can be computed with full security. This result directly improves one of the results by Halevi et al. [20], who showed that any *all-but-one forced* solitary output functionality (i.e., either B or C but not necessarily both, can fix the output distribution), can be computed with full security.

For this result, we require to strengthen the definition of (all-but-P) $\mathrm{CORE}_{\wedge}$-forced. Intuitively, all-but-P strong $\mathrm{CORE}_{\wedge}$-forced requires that the output distributions in some of the combinatorial rectangles in $\mathcal{R}_{\wedge}$ to be close. Roughly speaking, for every $x \in \mathcal{X}$ there exists a minimal input χ smaller than x with respect to an appropriate partial order, such that the output distribution in the rectangles in $\mathcal{R}_{\chi}$ can be fixed by the parties. Note that for $\mathrm{CORE}_{\wedge}$-forced, the distributions for different rectangles could be far. Similarly to [20], for our construction it suffices to consider an all-but-P strong $\mathrm{CORE}_{\wedge}$-forced functionality, for some $\mathsf{P} \in \{\mathsf{B}, \mathsf{C}\}$, where the remaining party in $\{\mathsf{B}, \mathsf{C}\} \setminus \{\mathsf{P}\}$ can fix the output distributions in the rectangles.

Definition 6.2 (All-but-P strong $\mathrm{CORE}_{\wedge}$-forced)**.** *Let $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ be a deterministic solitary output three-party functionality. We say that f is* all-but-B strong $\mathrm{CORE}_{\wedge}$-forced*, if there exists an ensemble of efficiently samplable distributions $\{R_{\kappa,j}\}_{\kappa\in\mathbb{N},j\in[m_{\wedge}]}$ such that the following holds. For every sequence of inputs $\mathbf{x} = \{x_\kappa \in \mathcal{X}\}_{\kappa\in\mathbb{N}}$, there exists a sequence of B-minimal inputs $\chi_{\mathsf{B}} = \{\chi_\kappa \in \mathcal{X}\}_{\kappa\in\mathbb{N}}$, such that $\chi_\kappa \preceq x_\kappa$ for all $\kappa \in \mathbb{N}$, and*

$$\{f(x_\kappa, y, z^*)\}_{\kappa\in\mathbb{N},x_\kappa\in\mathcal{X},i\in[n_\wedge],j\in[m_\wedge],y\in\mathcal{Y}_i^\wedge,z\in\mathcal{Z}_j^\wedge} \stackrel{\mathrm{S}}{\equiv} \{f(x_\kappa, y_\chi, z^*)\}_{\kappa\in\mathbb{N},x_\kappa\in\mathcal{X},i\in[n_\wedge],j\in[m_\wedge],y\in\mathcal{Y}_i^\wedge,z\in\mathcal{Z}_j^\wedge},$$

where y_χ is the lexicographically smallest element such that $y_\chi \equiv_{\chi_\kappa} y$ and where $z^* \leftarrow R_{\kappa,j}$. We define C-strong $\mathrm{CORE}_{\wedge}$-forced similarly.

Observe that for any functionality with a minimum element for A, strong-$\mathrm{CORE}_{\wedge}$-forced is equivalent to standard $\mathrm{CORE}_{\wedge}$-forced (since the minimum input satisfies the conditions stated above).

We are now ready to state our result.

Theorem 6.3. *Let $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ be a deterministic solitary output three-party functionality. Assume that oblivious transfer exists, that $|\mathcal{X}|, |\mathcal{Y}|, |\mathcal{Z}| = \mathrm{poly}(\kappa)$ and that f is all-but-P strong $\mathrm{CORE}_{\wedge}$-forced, for some $\mathsf{P} \in \{\mathsf{B}, \mathsf{C}\}$. Then f can be computed with full security.*

The proof of Theorem 6.3 is given in Section 6.1.2. We first discuss an interesting consequence of Theorems 6.1 and 6.3. Observe that the conditions stated in Theorems 6.1 and 6.3 are relaxations of the conditions stated in Theorems 3.7 and 3.9, respectively. Therefore, for the families of functionalities discussed in Section 3.2.1 (e.g., ternary-output), for which we have a complete characterization in the point-to-point model, it holds that if a functionality f can be computed assuming an honest majority but without a broadcast channel, then f can also be computed with a broadcast channel, but with no honest majority. Thus, we have the following corollary.

Corollary 6.4. *Let $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ be a deterministic solitary output three-party functionality. Assume that oblivious transfer exists, that $|\mathcal{X}|, |\mathcal{Y}|, |\mathcal{Z}| = \mathrm{poly}(\kappa)$ and that either $|\mathcal{W}| \leq 3$ or $|\mathcal{X}| \leq 2$. Then, if f can be computed with 1-security in the point-to-point model (without broadcast), then f can be computed with full security given a broadcast channel.*

Furthermore, since the conditions stated in Theorems 6.1 and 6.3 are *strict* relaxations of the conditions stated in Theorems 3.7 and 3.9, it follows that the converse is not true. Thus, the more common broadcast assumption is a *strictly stronger* than the honest majority assumption, for the above families of functionalities. A concrete example that showcase the separation is the following solitary output three-party variant of the GHKL function [19]. That is, the function $\mathsf{soGHKL} : \emptyset \times \{0,1,2\} \times \{0,1\} \mapsto \{0,1\}$ given by the matrix

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$$

where B chooses a row, C chooses a column, and the output of A is the value written in the chosen entry. Indeed, all inputs of B and C are equivalent, yet soGHKL is not forced and thus cannot be computed in the point-to-point model. On the other hand, Theorem 6.3 requires that only one of the parties needs to be able to fix the distribution of the output.

6.1 Proofs of the Results

In this section we proof Theorems 6.1 and 6.3. We start with proving Theorem 6.1 in Section 6.1.1, and then proving Theorem 6.3 in Section 6.1.2.

6.1.1 Proof of Theorem 6.1

The idea of the protocol is as follows. The parties first compute a 2-out-of-2 secret sharing of the output, where one share is given to party A, and the other share is given to either B or C depending on κ. The second party, denoted P, to hold a share then sends it to A who reconstructs the output. In case, P does not send the share, A replaces the aborting party's input with a default input, samples the input of the third party according the distribution associated with it as given by the $\mathrm{CORE}_\wedge$-forced property, and compute the function on these inputs.

Intuitively, the party that does not receive any share provides no advantage to the adversary. Additionally, corrupting A and P gives to the adversary only the output, hence it cannot attack the protocol. Finally, when A is honest, corrupting and aborting P can be simulated by sending an input sampled according the appropriate distribution associated with the default input of P, as given by the $\mathrm{CORE}_\wedge$-forced property of f.

We next formalize the above intuition. Similarly to Section 5.1, let $\mathcal{K}_\mathsf{B} \subseteq \mathbb{N}$ be the set of all $\kappa \in \mathbb{N}$ such that $y \equiv_x y'$ for all $x \in \mathcal{X}$ and $y, y' \in \mathcal{Y}$, and let $\mathcal{K}_\mathsf{C} \subseteq \mathbb{N}$ be the set of all $\kappa \in \mathbb{N}$ such that $z \equiv_x z'$ for all $x \in \mathcal{X}$ and $z, z' \in \mathcal{Z}$. Recall that we assume that f has the property where $\mathbb{N} \setminus (\mathcal{K}_\mathsf{B} \cup \mathcal{K}_\mathsf{C})$ is finite. Let $\mathsf{P} = \mathsf{B}$ if $\kappa \in \mathcal{K}_\mathsf{C}$, and let $\mathsf{P} = \mathsf{C}$ otherwise. Further let P' be the remaining party in $\{\mathsf{B}, \mathsf{C}\} \setminus \{\mathsf{P}\}$.

We let $\mathsf{ShrGen}_f(x, y, z)$ be the three-party functionality that computes a 2-out-of-2 additive secret sharing of the output $f(x, y, z)$. The functionality gives the shares to only A either B or C depending on κ (see Algorithm 6.5 below for a formal description). Additionally, it signs each of the shares using a one-time MAC. To simplify the presentation, we assume that a corrupted party will not modify it share, but may abort and not send it at all.

Algorithm 6.5 (Functionality ShrGen_f)**.**

Private inputs: A *holds* $x \in \mathcal{X}$, B *holds* $y \in \mathcal{Y}$, *and* C *holds* $z \in \mathcal{Z}$.

Common input: the parties hold the security parameter 1^κ.

1. *Compute* $w = f(x, y, z)$.
2. *Share* w *in an 2-out-of-2 secret sharing scheme. For* $\mathsf{Q} \in \{\mathsf{A}, \mathsf{P}\}$, *let* $w[\mathsf{Q}]$ *denote the share associated with party* Q.
3. *Party* A *receives* $w[\mathsf{A}]$, *and party* P *receives* $w[\mathsf{P}]$.

We next present a protocol for computing f in the $\{(\mathsf{ShrGen}_f, \mathsf{s.w.i.a})\}$-hybrid model.

Protocol 6.6 (π_{bc})**.**

Private inputs: A *holds* $x \in \mathcal{X}$, B *holds* $y \in \mathcal{Y}$, *and* C *holds* $z \in \mathcal{Z}$.

Common input: the parties hold the security parameter 1^κ.

1. *The parties invoke* $(\mathsf{ShrGen}_f, \mathsf{s.w.i.a})$ *with their inputs.*
   - *If* A *aborts then the computation halts.*
   - *Otherwise, if* P' *aborts then the parties restart without it, and with their input being set to default values. If* P *aborts, go to Step 3.*
2. *If* P *is still active, it sends* $w[\mathsf{P}]$ *to* A.
3. *If* P *aborts during any step of the computation, then* A *does the following.*
   (a) *If* $\mathsf{P} = \mathsf{B}$, *set* $y_0 \in \mathcal{Y}$ *to be the lexicographically smallest element, sample* $z^* \leftarrow R_{\kappa,1}$, *and output* $f(x, y_0, z^*)$.
   (b) *If* $\mathsf{P} = \mathsf{C}$, *set* $z_0 \in \mathcal{Z}$ *to be the lexicographically smallest element, sample* $y^* \leftarrow Q_{\kappa,1}$, *and output* $f(x, y^*, z_0)$.
4. *Otherwise,* A *outputs* $w[\mathsf{A}] + w[\mathsf{P}]$.

The next lemma immediately proves Theorem 6.1

Lemma 6.7. *Let* $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ *be a deterministic solitary output three-party functionality. Assume that* $|\mathcal{X}|, |\mathcal{Y}|, |\mathcal{Z}| = \mathrm{poly}(\kappa)$, *and that the following hold.*

1. *For all sufficiently large* κ, *either* $y \equiv_x y'$ *for all* $x \in \mathcal{X}$ *and* $y, y' \in \mathcal{Y}$, *or* $z \equiv_x z'$ *for all* $x \in \mathcal{X}$ *and* $z, z' \in \mathcal{Z}$.
2. f *is* $\mathrm{CORE}_\wedge$*-forced.*

Then π_{bc} *computes* f *with statistical full security in the* $(\mathsf{ShrGen}_f, \mathsf{s.w.i.a})$*-hybrid model.*

Proof. Clearly, the protocol is correct. Fix an adversary $\mathcal{A}$ corrupting a subset of the parties. Observe that if A is corrupted, then the adversary sees at most two shares whose sum is the output $f(x, y, z)$. Therefore, this case can be simulated.

We now assume that A is honest. In this case, $\mathcal{A}$ (possibly) sees only the random value $w[\mathsf{P}]$. If P' is corrupted and aborts during the call to $(\mathsf{ShrGen}_f, \mathsf{s.w.i.a})$, then the simulator replaces its input with a default value. If P is corrupted and aborts during any step of the protocol, then the simulator replaced its input with a random input as follows: If $\mathsf{P} = \mathsf{B}$ then send $y^* \leftarrow Q_{\kappa,i}$ where $i \in [n_\wedge]$ is the unique index satisfying $y_0 \in \mathcal{Y}_i^\wedge$. Otherwise, if $\mathsf{P} = \mathsf{C}$ then send $z^* \leftarrow R_{\kappa,j}$ where $j \in [m_\wedge]$ is the unique index satisfying $z_0 \in \mathcal{Z}_j^\wedge$.

Then for all $\kappa \in \mathcal{K}_\mathsf{C}$ (i.e., $\mathsf{P} = \mathsf{B}$) it holds that the output of A in the ideal world is $f(x, y^*, z)$, which by the $\text{CORE}_\wedge$-forced assumption, is statistically close to $f(x, y_0, z^*)$ that is the output of A in the real world. Similarly, for all $\kappa \notin \mathcal{K}_\mathsf{C}$ the outputs are statistically close. □

6.1.2 Proof of Theorem 6.3

We first present an intuitive description of the protocol. Towards constructing the protocol, we use an algorithm, denoted Str_P for $\mathsf{P} \in \{\mathsf{B}, \mathsf{C}\}$, which computes efficiently the sequence of minimal inputs that satisfy the conditions from Definition 6.2, for any all-but-P strong $\text{CORE}_\wedge$-forced three-party solitary output functionality. We present the algorithm in Section 6.1.3 below. We first use it to construct the protocol.

Assume without loss of generality that the functionality f is all-but-B strong $\text{CORE}_\wedge$-forced. The idea is for the parties to compute a 3-out-of-3 secret sharing of the output. Additionally, A and B will receive shares of the equivalence class of the input y held by B, with respect to the input $\chi \preceq x$ guaranteed to exist by the strong $\text{CORE}_\wedge$-forced assumption.

The protocol proceeds as follows. First, B sends its two shares to A. In case of abort, A and C restart the protocol with the input of B set to a default value. Otherwise, C sends its input to A, which reconstructs the output. In case C aborts, A reconstructs B's equivalence class and chooses any input from it. It then samples an input for C according to a default distribution, and computes f on these inputs. Intuitively, a corrupted B or C can be simulated by sending to the trusted party either a default input or sample an input according to the distribution guaranteed to exist by the $\text{CORE}_\wedge$-forced assumption. Additionally, a corrupt A only learns the output and the equivalence class of y with respect to χ, which can be inferred from the output since $\chi \preceq x$.

We next present a formal description of the protocol. Denote $N = N_\kappa = \prod_{x \in \mathcal{X}} n(x)$. We next define the three-party share generator functionality $\mathsf{ShrGen}'_f(x, y, z)$. Roughly speaking, it computes a three-out-of-three additive secret sharing of the output $f(x, y, z)$, and it shares between A and B the equivalence class of y with respect to to χ, for the B-minimal χ that is computed by the algorithm from Claim 6.12 (see Algorithm 6.8 below for a formal description). Additionally, it signs each of the shares using a one-time MAC. To simplify the presentation, we assume that a corrupted party will not modify it shares, but may abort and not send them at all. We now present a formal description of ShrGen'_f.

Algorithm 6.8 (Functionality ShrGen'_f)**.**

Private inputs: A *holds* $x \in \mathcal{X}$*,* B *holds* $y \in \mathcal{Y}$*, and* C *holds* $z \in \mathcal{Z}$*.*

Common input: the parties hold the security parameter 1^κ*.*

Computation:

1. *Compute* $w = f(x, y, z)$.
2. *Compute* $\chi = \mathsf{Str_B}(1^\kappa, x)$.
3. *Find the unique index* $i \in [n(\chi)]$ *satisfying* $y \in \mathcal{Y}_i^\chi$.

Sharing phase:

1. *Share* w *in an 3-out-of-3 secret sharing scheme. For* $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ *let* $w[\mathsf{P}]$ *denote the share associated with party* P.
2. *Sample* $i[\mathsf{A}] \leftarrow [N]$, *and let* $i[\mathsf{B}] = i - i[\mathsf{A}] \mod N$.[14]

Output: Party A *receives* $(\chi, w[\mathsf{A}], i[\mathsf{A}])$, *party* B *receives* $(w[\mathsf{B}], i[\mathsf{B}])$, *and party* C *receives* $w[\mathsf{C}]$ *(note that* χ *can also be computed locally by* A*).*

We next present a protocol for computing f in the $\{(\mathsf{ShrGen}'_f, \mathsf{s.w.i.a})\}$-hybrid model.

Protocol 6.9 (π'_{bc})**.**

Private inputs: A *holds* $x \in \mathcal{X}$, B *holds* $y \in \mathcal{Y}$, *and* C *holds* $z \in \mathcal{Z}$.

Common input: the parties hold the security parameter 1^κ.

1. *The parties invoke* $(\mathsf{ShrGen}'_f, \mathsf{s.w.i.a})$ *with their inputs.*
   - *If* A *aborts then the computation halts.*
   - *Otherwise, if any other party aborts the parties restart without it, and their input being set to a default value.*
2. *If* B *is still active, it sends* $(w[\mathsf{B}], i[\mathsf{B}])$ *to* A.
   - *In case* B *aborts, the parties set its input to a default value and restart the protocol without it.*
3. *If* C *is still active, it sends* $w[\mathsf{C}]$ *to* A.
   - *In case* C *aborts,* A *does the following.*
     (a) *Set* $i = i[\mathsf{A}] + i[\mathsf{B}] \mod n(\chi)$ *if* B *is active, and set* $i = 1$ *otherwise.*
     (b) *Compute and output* $w^* = f(x, y_\chi, z^*)$, *where* y_χ *is the lexicographically smallest element of* $\mathcal{Y}_i^\chi$, *and where* $z^* \leftarrow R_{\kappa,1}$.
4. *If no party aborts,* A *reconstructs the output.*

The next lemma immediately proves Theorem 6.3

Lemma 6.10. *Let* $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ *be a deterministic solitary output three-party functionality. Assume that* $|\mathcal{X}|, |\mathcal{Y}|, |\mathcal{Z}| = \mathrm{poly}(\kappa)$, *and that* f *is all-but-*B *strong. Then* π'_{bc} *computes* f *with statistical full security in the* $(\mathsf{ShrGen}'_f, \mathsf{s.w.i.a})$*-hybrid model.*

[14]We let $\bmod n$ output in $[n]$ instead of $\{0, \dots, n-1\}$ for convenience.

Proof. Clearly, the protocol is correct. Fix an adversary $\mathcal{A}$ corrupting a subset of the parties. We separate the proof into two cases. For the first case, let us assume that A is honest. We assume that both B and C are corrupted. The case where exactly one of them is corrupted can be handled similarly. The simulator $\mathsf{Sim}_{\mathcal{A}}$ does the following.

1. Query $\mathcal{A}$ for its input y and z to $(\mathsf{ShrGen}'_f, \mathsf{s.w.i.a})$.

2. Send to $\mathcal{A}$ the values $(w[\mathsf{B}], i[\mathsf{B}])$ and $w[\mathsf{C}]$, where $w[\mathsf{B}], w[\mathsf{C}] \leftarrow \mathcal{W}$ and where $i[\mathsf{B}] \leftarrow [N]$. If $\mathcal{A}$ replies with $(\mathsf{abort}, \mathsf{P})$, for some $\mathsf{P} \in \{\mathsf{B}, \mathsf{C}\}$, then go back to Step 1 with the input of P set to a default value.

3. Otherwise, if B aborts at Step 2, then go back to Step 1 with the input of B set to a default value.

4. If C aborts, sample $z^* \leftarrow R_{\kappa,1}$ and send to the trusted party (y', z^*), where $y' = y$ if B is active, and y' is a default value otherwise. Output whatever $\mathcal{A}$ outputs, and halt.

5. Otherwise, if C does not abort, send y and z to the trusted party T, output whatever $\mathcal{A}$ outputs, and halt.

It's clear that the views of $\mathcal{A}$ in both worlds are identically distributed, and in particular, its responses are identically distributed as well. First, consider the case where C does not abort at Step 3. Then the output of A in both worlds is $f(x, y', z)$, where $y' = y$ if B is active, and y' is a default value otherwise.

Now, consider the case where C does abort at Step 3. Let us first consider the real world. Then the value i set by A is $i = 1$ if B is inactive, and $i \in [n(\chi)]$ is the unique index satisfying $y \in \mathcal{Y}_i^{\chi}$ if B is active. Then the output of A is of the form $f(x, y'_{\mathsf{real}}, z^*)$, where $z' \leftarrow R_{\kappa,1}$, and where y'_{real} is the lexicographically smallest element in $\mathcal{Y}_i^{\chi}$. Let us now consider the ideal world. The of A in this case is $f(x, y'_{\mathsf{ideal}}, z^*)$, where $z^* \leftarrow R_{\kappa,1}$ as before, and where $y'_{\mathsf{ideal}} = y$ if B is active, and is a default value otherwise. Observe that, regardless of whether or not B is active, it holds that $y'_{\mathsf{real}} \equiv_{\chi} y'_{\mathsf{ideal}}$. By Claim 6.12, it follows that both outputs are statistically close.

We now assume that A is corrupted. We only deal with the case where C is also corrupted, since the other cases are simpler. We define the simulator $\mathsf{Sim}_{\mathcal{A}}$ as follows.

1. Query $\mathcal{A}$ for its inputs to $(\mathsf{ShrGen}'_f, \mathsf{s.w.i.a})$. If the adversary aborts then restart without the aborting party. Let x and z be the inputs used in the last call.

2. Send to $\mathcal{A}$ random shares of the output $w[\mathsf{A}], w[\mathsf{C}] \leftarrow \mathcal{W}$, the random share $i[\mathsf{A}] \leftarrow [N]$, and the R-minimal element χ as computed by ShrGen'_f (recall that χ depends only on x).

3. Send x and z to the trusted party T, and let w be the output received from T.

4. Set $w[\mathsf{B}] = w - w[\mathsf{A}] - w[\mathsf{C}]$.

5. Find $i \in [n(\chi)]$ for which there exists $y' \in \mathcal{Y}_i^{\chi}$ and $z' \in \mathcal{Z}$ such that $w = f(x, y', z')$.

6. Let $i[\mathsf{B}] = i - i[\mathsf{A}] \mod n(\chi)$.

7. Send to $\mathcal{A}$ the pair $(w[\mathsf{B}], i[\mathsf{B}])$, output whatever $\mathcal{A}$ outputs and halt.

Clearly, since $n(\chi)$ divides N, the view of $\mathcal{A}$ in both worlds are identically distributed. Since no honest party obtains an output from T, it follows that the real and ideal world are identically distributed. $\square$

6.1.3 The $\mathsf{Str_P}$ Algorithm For Finding χ_B and χ_C

We next present the idea behind the algorithm $\mathsf{Str_P}$ for $\mathsf{P} = \mathsf{B}$ (the case where $\mathsf{P} = \mathsf{C}$ is analogous). For a given input x, the algorithm searches for a B-minimal input χ_B and two rectangles $\mathcal{Y}_i^{\chi_\mathsf{B}} \times \mathcal{Z}_j^{\chi_\mathsf{B}}$ and $\mathcal{Y}_{i'}^{\chi_\mathsf{B}} \times \mathcal{Z}_j^{\chi_\mathsf{B}}$ such that statistical distance between the output distributions that are associated with the rectangles (as given by sampling either y or z according to the corresponding distributions and computing f over these input) is maximized. The algorithm then outputs χ_B. Intuitively, if χ_B does not satisfy the properties from Definition 6.2, then this contradicts the maximality of the statistical distance. Note that the algorithm is efficient since we assume the domain of f to be of polynomial size. We now formalize the above intuition.

Algorithm 6.11 ($\mathsf{Str_P}$)**.**

Setting: Suppose that $f : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \mapsto \mathcal{W}$ is a three-party solitary output all-but-P strong $\mathrm{CORE}_\wedge$-forced functionality, and let $\mathcal{Q}$ and $\mathcal{R}$ be the associated distribution ensembles.

Input: The security parameter 1^κ and $x \in \mathcal{X}$.

Computation:

- *If $\mathsf{P} = \mathsf{B}$, find a B-minimal $\chi \preceq_\mathsf{B} x$, $\hat{y}, \hat{y}' \in \mathcal{Y}$, and $j \in [m(\chi_\mathsf{B})]$ such that $\hat{y} \not\equiv_\chi \hat{y}'$ and $\hat{y} \equiv_x \hat{y}'$, that maximizes*
$$\mathrm{SD}\left(f(x, \hat{y}, z_1^*),\ f(x, \hat{y}', z_2^*)\right),$$
where $z_1^*, z_2^* \leftarrow R_{\kappa,j}$ are independent.
- *If $\mathsf{P} = \mathsf{C}$ find a C-minimal $\chi \preceq_\mathsf{C} x$, $i \in [n(\chi_\mathsf{C})]$ and $\hat{z}, \hat{z}' \in \mathcal{Z}$ such that $\hat{z} \not\equiv_\chi \hat{z}'$ and $\hat{z} \equiv_x \hat{z}'$, that maximizes*
$$\mathrm{SD}\left(f(x, y_1^*, \hat{z}),\ f(x, y_2^*, \hat{z}')\right),$$
where $y_1^*, y_2^* \leftarrow Q_{\kappa,i}$ are independent.

Output: χ.

Claim 6.12. *Suppose that f is all-but-P strong $\mathrm{CORE}_\wedge$-forced, and fix a sequence of inputs $\mathbf{x} = \{x_\kappa \in \mathcal{X}\}_{\kappa\in\mathbb{N}}$. Define the sequence $\chi = \{\chi_\kappa\}_{\kappa\in\mathbb{N}}$, where χ_κ is the output of $\mathsf{Str_P}(1^\kappa, x_\kappa)$. Then χ is the sequence guaranteed to exists by the all-but-P strong $\mathrm{CORE}_\wedge$-forced assumption. That is, if $\mathsf{P} = \mathsf{B}$ then*
$$\{f(x_\kappa, y, z^*)\}_{\kappa, x_\kappa, i, j, y, z} \stackrel{\mathrm{S}}{\equiv} \{f(x_\kappa, y_\chi, z^*)\}_{\kappa, x_\kappa, i, j, y, z},$$
where y_χ is the lexicographically smallest element such that $y_\chi \equiv_{\chi_\kappa} y$ and where $z^* \leftarrow R_{\kappa,j}$. Similarly, if $\mathsf{P} = \mathsf{C}$ then an analogous statement holds.

Proof. We prove the statement only for the case where $\mathsf{P} = \mathsf{B}$, as the other case is analogous. Assume that the claim is false. Then for infinitely many κ, there exists $i \in [n_\wedge]$, $j \in [m_\wedge]$, $y \in \mathcal{Y}_i^\wedge$, and $z \in \mathcal{Z}_j^\wedge$, such that
$$\mathrm{SD}\left(f(x_\kappa, y, z_1^*),\ f(x_\kappa, y_\chi, z_2^*)\right) > 1/\operatorname{poly}(\kappa),$$
where $z_1^*, z_2^* \leftarrow R_{\kappa,j}$ are independent. Since $y \equiv_{\chi_\kappa} y_\chi$, by the $\mathrm{CORE}_\wedge$-forced property of f, it follows that there exists a different sequence $\chi' = \{\chi'_\kappa\}_{\kappa\in\mathbb{N}}$ such that $y \not\equiv_{\chi'_\kappa} y_\chi$. However this contradict the maximality assumption over χ. $\square$

7 Various Interesting Examples

In this section we provide some interesting examples of functionalities, and identify which can be securely computed with 1-security in the point-to-point model. Our examples include variants of private-set intersection. Throughout the section, for natural numbers $k, \ell, m \in \mathbb{N}$ satisfying $k \leq \ell \leq m$, we denote

$$\binom{[m]}{k} = \{\mathcal{S} \subseteq [m] : |\mathcal{S}| = k\}$$

and we denote

$$\binom{[m]}{k, \ell} = \{\mathcal{S} \subseteq [m] : k \leq |\mathcal{S}| \leq \ell\}.$$

Claim 7.1. *For two natural numbers $k \leq m$, let $\mathsf{disj}_{k,m} : \binom{[m]}{k}^3 \mapsto \{0,1\}$ be the solitary output three-party* disjointness *functionality defined as*

$$\mathsf{disj}_{k,m}(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3) = \begin{cases} 1 & \textit{if } \mathcal{S}_1 \cap \mathcal{S}_2 \cap \mathcal{S}_3 = \emptyset \\ 0 & \textit{otherwise} \end{cases}$$

Then $\mathsf{disj}_{k,m}$ can be computed with 1-security if and only if $k > 2m/3$ or $k = 0$.

Proof. Observe that if $k > 2m/3$ or $k = 0$, then $\mathsf{disj}_{k,m}$ is constant, and thus can be securely computed. We now assume that $0 < k \leq 2m/3$ and show that $\mathsf{disj}_{k,m}$ is not $\text{CORE}_\wedge$-forced, and thus cannot be computed securely. We separate the proof into two cases.

Case 1: $m/2 < k \leq 2m/3$. We first show that for any $\mathcal{S}_1$, it holds that $\mathcal{S}_2 \equiv_{\mathcal{S}_1} \mathcal{S}_2'$ for all $\mathcal{S}_2, \mathcal{S}_2' \in \binom{[m]}{k}$. Indeed, since $k > m/2$ it follows that $\mathcal{S}_1 \cap \mathcal{S}_2 \neq \emptyset$ and that $\mathcal{S}_1 \cap \mathcal{S}_2' \neq \emptyset$. Therefore, for any $\mathcal{S}_3 \supseteq \mathcal{S}_1 \cap \mathcal{S}_2$ it holds that $\mathsf{disj}_{k,m}(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3) = 0$. Similarly, for any $\mathcal{S}_3' \supseteq \mathcal{S}_1 \cap \mathcal{S}_2'$ it holds that $\mathsf{disj}_{k,m}(\mathcal{S}_1, \mathcal{S}_2', \mathcal{S}_3') = 0$. By symmetry, $\mathcal{S}_3 \equiv_{\mathcal{S}_1} \mathcal{S}_3'$ for all $\mathcal{S}_3, \mathcal{S}_3' \in \binom{[m]}{k}$ as well.

Now, assume towards contradiction that there exists a distribution $\mathcal{R} = \{R_\kappa\}_{\kappa \in \mathbb{N}}$ over $\binom{[m]}{k}$ such that

$$\left\{\mathsf{disj}_{k,m}\left(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3^*\right)\right\}_{\kappa, \mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_2'} \stackrel{\text{S}}{\equiv} \left\{\mathsf{disj}_{k,m}\left(\mathcal{S}_1, \mathcal{S}_2', \mathcal{S}_3^*\right)\right\}_{\kappa, \mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_2'}, \tag{10}$$

where $\mathcal{S}_3^* \leftarrow R_\kappa$. Since the domain of $\mathsf{disj}_{k,m}$ is finite, there exists $\mathcal{S}_3 \in \binom{[m]}{k}$ such that $\Pr_{\mathcal{S}_3^* \leftarrow R_\kappa}[\mathcal{S}_3^* = \mathcal{S}_3] \geq p$ infinitely often for some constant $p > 0$. Consider $\mathcal{S}_1$ that minimizes $|\mathcal{S}_1 \cap \mathcal{S}_3|$, i.e., it holds that $|\mathcal{S}_1 \cap \mathcal{S}_3| = 2k - m$. Since $2k - m \leq m/3$, there exists $\mathcal{S}_2$ such that $\mathcal{S}_1 \cap \mathcal{S}_2 \cap \mathcal{S}_3 = \emptyset$. Therefore

$$\Pr_{\mathcal{S}_3^* \leftarrow R_\kappa}\left[\mathsf{disj}_{k,m}\left(\mathcal{S}_1 \cap \mathcal{S}_2 \cap \mathcal{S}_3^*\right) = 1\right] \geq \Pr_{\mathcal{S}_3^* \leftarrow R_\kappa}\left[\mathcal{S}_3^* = \mathcal{S}_3\right] \geq p,$$

holds infinitely often. On the other hand, for $\mathcal{S}_2' = \mathcal{S}_1$ it holds that $\mathsf{disj}_{k,m}(\mathcal{S}_1, \mathcal{S}_2', \cdot)$ is the constant 0 function, hence

$$\Pr_{\mathcal{S}_3^* \leftarrow R_\kappa}\left[\mathsf{disj}_{k,m}\left(\mathcal{S}_1 \cap \mathcal{S}_2' \cap \mathcal{S}_3^*\right) = 1\right] = 0$$

for all κ, contradicting Equation (10).

Case 2: $0 < k \leq m/2$**.** The proof follows similar arguments to the previous case. We first show that for any $\mathcal{S}_1$, it holds that $\mathcal{S}_2 \equiv_{\mathcal{S}_1} \mathcal{S}_2'$ for all $\mathcal{S}_2, \mathcal{S}_2' \in \binom{[m]}{k}$. Indeed, since $k \leq m/2$ it follows that there exists $\mathcal{S}_3 \in \binom{[m]}{k}$ such that $\mathcal{S}_1 \cap \mathcal{S}_3 = \emptyset$, and in particular, $\mathsf{disj}_{k,m}(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3) = \mathsf{disj}_{k,m}(\mathcal{S}_1, \mathcal{S}_2', \mathcal{S}_3) = 1$. By symmetry, $\mathcal{S}_3 \equiv_{\mathcal{S}_1} \mathcal{S}_3'$ for all $\mathcal{S}_3, \mathcal{S}_3' \in \binom{[m]}{k}$ as well.

Assume towards contradiction that there exists a distribution ensemble $\mathcal{R} = \{R_\kappa\}_{\kappa \in \mathbb{N}}$ over $\binom{[m]}{k}$ such that

$$\left\{\mathsf{disj}_{k,m}\left(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3^*\right)\right\}_{\kappa, \mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_2'} \overset{\text{S}}{\equiv} \left\{\mathsf{disj}_{k,m}\left(\mathcal{S}_1, \mathcal{S}_2', \mathcal{S}_3^*\right)\right\}_{\kappa, \mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_2'}, \tag{11}$$

where $\mathcal{S}_3^* \leftarrow R_\kappa$. Since the domain of $\mathsf{disj}_{k,m}$ is finite, there exists $\mathcal{S}_3 \in \binom{[m]}{k}$ such that $\Pr_{\mathcal{S}_3^* \leftarrow R_\kappa}[\mathcal{S}_3^* = \mathcal{S}_3] \geq p$ infinitely often for some constant $p > 0$. Consider $\mathcal{S}_1 = \mathcal{S}_2 = \mathcal{S}_3$. Then, as $k \neq 0$ it follows that $\mathcal{S}_3 \neq \emptyset$. Thus

$$\Pr_{\mathcal{S}_3^* \leftarrow R_\kappa}\left[\mathsf{disj}_{k,m}\left(\mathcal{S}_1 \cap \mathcal{S}_2 \cap \mathcal{S}_3^*\right) = 0\right] \geq \Pr_{\mathcal{S}_3^* \leftarrow R_\kappa}[\mathcal{S}_3^* = \mathcal{S}_3] \geq p,$$

holds infinitely often. On the other hand, since $k \leq m/2$ there exists $\mathcal{S}_2' \in \binom{[m]}{k}$ such that $\mathcal{S}_1 \cap \mathcal{S}_2' = \emptyset$, hence

$$\Pr_{\mathcal{S}_3^* \leftarrow R_\kappa}\left[\mathsf{disj}_{k,m}\left(\mathcal{S}_1 \cap \mathcal{S}_2' \cap \mathcal{S}_3^*\right) = 0\right] = 0$$

for all κ, contradicting Equation (11). □

Claim 7.2. *For $k_1, \ell_1, k_2, \ell_2, m \in \mathbb{N}$ where $0 \leq k_1 \leq \ell_1 \leq m$ and $0 \leq k_2 \leq \ell_2 \leq m$, let $\mathsf{PSI}^{\ell_1,\ell_2}_{k_1,k_2,m}$: $\emptyset \times \binom{[m]}{k_1,\ell_1} \times \binom{[m]}{k_2,\ell_2} \mapsto 2^{[m]}$ be the solitary output three-party* private set intersection *functionality defined as*

$$\mathsf{PSI}^{\ell_1,\ell_2}_{k_1,k_2,m}(\mathcal{S}_1, \mathcal{S}_2) = \mathcal{S}_1 \cap \mathcal{S}_2.$$

Then $\mathsf{PSI}^{\ell_1,\ell_2}_{k_1,k_2,m}$ can be computed with 1-security if and only if one of the following holds.

1. *$k_1 = k_2 = 0$, or*
2. *$\ell_1 = 0$ or $\ell_2 = 0$, or*
3. *$k_1 = m$ or $k_2 = m$.*

Proof. We write PSI instead of $\mathsf{PSI}^{\ell_1,\ell_2}_{k_1,k_2,m}$ for brevity. We first show the positive direction. If $k_1 = k_2 = 0$, then PSI is forced since both B and C can fix the output to be $\emptyset$. If $\ell_1 = 0$ or $\ell_2 = 0$ then PSI is the constant $\emptyset$ function. If $k_2 = m$, then PSI is independent of its second argument and in particular, is forced. [15]

We now show the negative direction. We first show that $\mathcal{S}_1 \equiv \mathcal{S}_1'$ and $\mathcal{S}_2 \equiv \mathcal{S}_2'$, for all $\mathcal{S}_1, \mathcal{S}_1' \in \binom{[m]}{k_1,\ell_1}$ and $\mathcal{S}_2, \mathcal{S}_2' \in \binom{[m]}{k_2,\ell_2}$. We show only the former as the latter can be proved using an analogous argument. Observe that if $|\mathcal{S}_1 \cap \mathcal{S}_1'| \geq k_2$, then for any $\mathcal{S}_2 \subseteq \mathcal{S}_1 \cap \mathcal{S}_1'$ of size $k_2 \leq |\mathcal{S}_2| \leq \ell_2$ it holds that $\mathsf{PSI}(\mathcal{S}_1, \mathcal{S}_2) = \mathcal{S}_2 = \mathsf{PSI}(\mathcal{S}_1', \mathcal{S}_2)$. On the other hand, if $|\mathcal{S}_1 \cap \mathcal{S}_1'| < k_2$, then there exists $\mathcal{S}_2 \in \binom{[m]}{k_2}$ such that $\mathcal{S}_1 \cap \mathcal{S}_1' \subseteq \mathcal{S}_2$ and $\mathcal{S}_2 \cap (\mathcal{S}_1 \setminus \mathcal{S}_1') = \mathcal{S}_2 \cap (\mathcal{S}_1' \setminus \mathcal{S}_1) = \emptyset$.

[15]Note that in all cases we do not need to assume the existence of OT. For the first case, where $k_1 = k_2 = 0$, we can use the protocol where if the fair computation fails, we let A output $\emptyset$. For the other two cases the computation is trivial, since either the function is constant, or the protocol where B or C send their input to A is secure.

Now, assume towards contradiction that PSI is $\mathrm{CORE}_{\wedge}$-forced. since all inputs are equivalent, it follows that PSI is forced. Then there exists an ensemble of efficiently samplable distributions $\mathcal{R} = \{R_\kappa\}_{\kappa\in\mathbb{N}}$ such that

$$\{\mathcal{S}_1 \cap \mathcal{S}_2^*\}_{\kappa,\mathcal{S}_1,\mathcal{S}_1'} \stackrel{\mathrm{S}}{\equiv} \{\mathcal{S}_1' \cap \mathcal{S}_2^*\}_{\kappa,\mathcal{S}_1,\mathcal{S}_1'}, \tag{12}$$

where $\mathcal{S}_2^* \leftarrow R_\kappa$. Since the domain of PSI is finite, there exists $\mathcal{S}_2 \in \binom{[m]}{k_2,\ell_2}$ such that $\Pr_{\mathcal{S}_2^*\leftarrow R_\kappa}[\mathcal{S}_2^* = \mathcal{S}_2] \geq p$ infinitely often for some constant $p > 0$. Now, recall that $k_1 \neq 0$ or $k_2 \neq 0$. We assume the latter without loss of generality. Thus, $\mathcal{S}_2 \neq \emptyset$. We next separate the proof into two cases.

Case 1: $|\mathcal{S}_2| \leq \ell_1$. In this case, there exists $\mathcal{S}_1 \in \binom{[m]}{k_1,\ell_1}$ such that $\mathcal{S}_2 \subseteq \mathcal{S}_1$. Therefore

$$\Pr_{\mathcal{S}_2^*\leftarrow R_\kappa}[\mathcal{S}_1 \cap \mathcal{S}_2^* = \mathcal{S}_2] \geq \Pr_{\mathcal{S}_2^*\leftarrow R_\kappa}[\mathcal{S}_2^* = \mathcal{S}_2] \geq p$$

infinitely often. However, since $\mathcal{S}_2 \neq \emptyset$ there exists $\mathcal{S}_1' \in \binom{[m]}{k_1,\ell_1}$ such that $\mathcal{S}_2 \not\subseteq \mathcal{S}_1'$. Thus

$$\Pr_{\mathcal{S}_2^*\leftarrow R_\kappa}[\mathcal{S}_1' \cap \mathcal{S}_2^* = \mathcal{S}_2] = 0$$

for all κ, contradicting Equation (12).

Case 2: $|\mathcal{S}_2| > \ell_1$. In this case, there exists $\mathcal{S}_1, \mathcal{S}_1' \in \binom{[m]}{k_1,\ell_1}$ such that $\mathcal{S}_1, \mathcal{S}_1' \subseteq \mathcal{S}_2$. Moreover, since $k_1 \neq m$ and $\ell_1 \neq 0$, it follows that we can take $\mathcal{S}_1 \neq \mathcal{S}_1'$. Thus

$$\Pr_{\mathcal{S}_2^*\leftarrow R_\kappa}[\mathcal{S}_1 \cap \mathcal{S}_2^* = \mathcal{S}_1] \geq \Pr_{\mathcal{S}_2^*\leftarrow R_\kappa}[\mathcal{S}_2^* = \mathcal{S}_2] \geq p$$

infinitely often. However,

$$\Pr_{\mathcal{S}_2^*\leftarrow R_\kappa}[\mathcal{S}_1' \cap \mathcal{S}_2^* = \mathcal{S}_1] = 0$$

for all κ, contradicting Equation (12). □

Claim 7.3. *For $k_1, \ell_1, k_2, \ell_2, m \in \mathbb{N}$ where $0 \leq k_1 \leq \ell_1 \leq m$ and $0 \leq k_2 \leq \ell_2 \leq m$, let $\mathsf{PSIZE}_{k_1,k_2,m}^{\ell_1,\ell_2} : \emptyset \times \binom{[m]}{k_1,\ell_1} \times \binom{[m]}{k_2,\ell_2} \mapsto \{0,\ldots,m\}$ be the solitary output three-party functionality defined as*

$$\mathsf{PSIZE}_{k_1,k_2,m}^{\ell_1,\ell_2}(\mathcal{S}_1,\mathcal{S}_2) = |\mathcal{S}_1 \cap \mathcal{S}_2|\,.$$

Then $\mathsf{PSIZE}_{k_1,k_2,m}^{\ell_1,\ell_2}$ can be computed with 1-security if and only if one of the following holds.

1. *$k_1 = k_2 = 0$, or*
2. *$\ell_1 = 0$ or $\ell_2 = 0$,*
3. *$k_1 = m$ or $k_2 = m$, or*
4. *$k_1 = \ell_1$ and $k_2 = \ell_2$.*

Proof. We write PSIZE instead of $\mathsf{PSIZE}^{\ell_1,\ell_2}_{k_1,k_2,m}$ for brevity. Similarly to the PSI functionality, if $k_1 = k_2 = 0$, then PSIZE is forced since both B and C can fix the output to be 0. If $\ell_1 = 0$ or $\ell_2 = 0$, then PSIZE is the constant 0. If $k_1 = m$ or $k_2 = m$, then PSIZE is independent of one of its arguments and in particular, is forced. Finally, if $k_1 = \ell_1$ and $k_2 = \ell_2$ then PSIZE is forced since the uniform distribution for both parties fixes the output distribution to be uniform.

We now show the negative direction. We first show that $\mathcal{S}_1 \equiv \mathcal{S}'_1$ and $\mathcal{S}_2 \equiv \mathcal{S}'_2$, for all $\mathcal{S}_1, \mathcal{S}'_1 \in \binom{[m]}{k_1,\ell_1}$ and $\mathcal{S}_2, \mathcal{S}'_2 \in \binom{[m]}{k_2,\ell_2}$. We show only the former as the latter can be proved using an analogous argument. Observe that the set of possible outputs for a fixed $\mathcal{S}_1 \in \binom{[m]}{k_1,\ell_1}$ is exactly

$$\{\max\{0, |\mathcal{S}_1| + k_2 - m\}, \ldots, \min\{|\mathcal{S}_1|, \ell_2\}\}.$$

Then, if $|\mathcal{S}_1| = |\mathcal{S}'_1|$ there are $\mathcal{S}_2, \mathcal{S}'_2 \in \binom{[m]}{k_2,\ell_2}$ such that $|\mathcal{S}_1 \cap \mathcal{S}_2| = |\mathcal{S}'_1 \cap \mathcal{S}_2|$. Next, consider the case where $\mathcal{S}'_1 = \mathcal{S} \cup \{a\}$, where $a \notin \mathcal{S}_1$. Then there are no such $\mathcal{S}_2$ and $\mathcal{S}'_2$ if and only if

$$\max\{0, |\mathcal{S}_1| + 1 + k_2 - m\} > \min\{|\mathcal{S}_1|, \ell_2\}.$$

However, since $|\mathcal{S}_1|, \ell_2 \geq 0$ and $|\mathcal{S}_1| + k_2 - m < |\mathcal{S}_1|$ as $k_2 \neq m$, it follows that $|\mathcal{S}_1| + k_2 - m \geq \ell_2$. This is clearly impossible since this implies that $|\mathcal{S}_1| \geq \ell_2 - k + m \geq m$. The case where $|\mathcal{S}'_1| > |\mathcal{S}_1| + 1$ can be done using an inductive argument (over $|\mathcal{S}'_1| - |\mathcal{S}_1|$).

We now show that PSIZE is not forced and hence cannot be computed with 1-security. Recall that for the negative direction, we assume that $k_1 \neq \ell_1$ or $k_2 \neq \ell_2$. Assume the former without loss of generality, and assume towards contradiction that PSIZE is forced. Then there exists an ensemble of efficiently samplable distributions $\mathcal{R} = \{R_\kappa\}_{\kappa\in\mathbb{N}}$ such that

$$\{|\mathcal{S}_1 \cap \mathcal{S}^*_2|\}_{\kappa,\mathcal{S}_1,\mathcal{S}'_1} \stackrel{\text{S}}{\equiv} \{|\mathcal{S}'_1 \cap \mathcal{S}^*_2|\}_{\kappa,\mathcal{S}_1,\mathcal{S}'_1}, \tag{13}$$

where $\mathcal{S}^*_2 \leftarrow R_\kappa$. Since the domain of PSIZE is finite, there exists $\mathcal{S}_2 \in \binom{[m]}{k_2,\ell_2}$ such that $\Pr_{\mathcal{S}^*_2 \leftarrow R_\kappa}[\mathcal{S}^*_2 = \mathcal{S}_2] \geq p$ infinitely often for some constant $p > 0$. We separate the proof into two cases.

Case 1: $k_1 < k_2$**.** Fix $\mathcal{S}_1 \in \binom{[m]}{k_1}$ such that $\mathcal{S}_1 \subseteq \mathcal{S}_2$, and fix some $a \in \mathcal{S}_2 \setminus \mathcal{S}_1$. Let $n = |\mathcal{S}_1 \cap \mathcal{S}_2|$, and let $\mathcal{S}'_1 = \mathcal{S}_1 \cup \{a\}$ (note that $\mathcal{S}'_1 \in \binom{[m]}{k_1,\ell_1}$ since $k_1 \neq \ell_1$). Then

$$\Pr_{\mathcal{S}^*_2 \leftarrow R_\kappa}[|(\mathcal{S}_1 \cup \{a\}) \cap \mathcal{S}^*_2| = n + 1] \geq \Pr_{\mathcal{S}^*_2 \leftarrow R_\kappa}[\mathcal{S}^*_2 = \mathcal{S}_2] \geq p,$$

infinitely often. However, for those exact same κ it holds that

$$\Pr_{\mathcal{S}^*_2 \leftarrow R_\kappa}[|\mathcal{S}_1 \cap \mathcal{S}^*_2| = n + 1] = 0,$$

resulting in a contradiction.

Case 2: $k_1 \geq k_2$**.** Fix $\mathcal{S}_1 \in \binom{[m]}{k_1+1}$ such that $\mathcal{S}_2 \subseteq \mathcal{S}_1$, and fix some $a \in \mathcal{S}_1 \setminus \mathcal{S}_2$. Let $n = |\mathcal{S}_1 \cap \mathcal{S}_2|$. Then

$$\Pr_{\mathcal{S}^*_2 \leftarrow R_\kappa}[|\mathcal{S}_1 \cap \mathcal{S}^*_2| = n] \geq \Pr_{\mathcal{S}^*_2 \leftarrow R_\kappa}[\mathcal{S}^*_2 = \mathcal{S}_2] \geq p,$$

infinitely often. However, since $\ell_2 \neq 0$ it follows that $\mathcal{S}_2 \neq \emptyset$, hence for those exact same κ it holds that

$$\Pr_{\mathcal{S}^*_2 \leftarrow R_\kappa}[|(\mathcal{S}_1 \setminus \{a\}) \cap \mathcal{S}^*_2| = n] = 0,$$

resulting in a contradiction. $\square$

Claim 7.4. *For $k_1, \ell_1, k_2, \ell_2, m \in \mathbb{N}$ where $0 \le k_1 \le \ell_1 \le m$ and $0 \le k_2 \le \ell_2 \le m$, let $\mathsf{disj}^{\ell_1,\ell_2}_{k_1,k_2,m}$: $\emptyset \times \binom{[m]}{k_1,\ell_1} \times \binom{[m]}{k_2,\ell_2} \mapsto \{0,\ldots,m\}$ be the solitary output three-party functionality defined as*

$$\mathsf{disj}^{\ell_1,\ell_2}_{k_1,k_2,m}(\mathcal{S}_1,\mathcal{S}_2) = \begin{cases} 1 & \text{if } \mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Then $\mathsf{disj}^{\ell_1,\ell_2}_{k_1,k_2,m}$ can be computed with 1-security if and only if one of the following holds.

1. *$k_1 = k_2 = 0$, or*
2. *$\ell_1 = 0$ or $\ell_2 = 0$, or*
3. *$k_1 = m$ or $k_2 = m$, or*
4. *$k_1 = \ell_1$ and $k_2 = \ell_2$.*
5. *$\ell_1 + k_2 > m$ and $k_1 + \ell_2 > m$.*

Proof. We write disj instead of $\mathsf{disj}^{\ell_1,\ell_2}_{k_1,k_2,m}$ for brevity. Similarly to the PSI and PSIZE functionality, if $k_1 = k_2 = 0$, then disj is forced since both B and C can fix the output to be 1. If $\ell_1 = 0$ or $\ell_2 = 0$ then disj is the constant 1. If $k_1 = m$ or $k_2 = m$, then disj is independent of one of its arguments and in particular, is forced. If $k_1 = \ell_1$ and $k_2 = \ell_2$ then disj is forced since the uniform distribution for both parties fixes the output distribution to be uniform. Finally, if $\ell_1 + k_2 > m$ and $k_1 + \ell_2 > m$, then both parties can fix the output to be 0.

We now show the negative direction. We first show that $\mathcal{S}_1 \equiv \mathcal{S}_1'$ and $\mathcal{S}_2 \equiv \mathcal{S}_2'$, for all $\mathcal{S}_1, \mathcal{S}_1' \in \binom{[m]}{k_1,\ell_1}$ and $\mathcal{S}_2, \mathcal{S}_2' \in \binom{[m]}{k_2,\ell_2}$. We show only the former as the latter can be proved using an analogous argument. Since disj is Boolean, it suffices to show that there exists $\mathcal{S}_1 \in \binom{[m]}{k_1,\ell_1}$ for which $\mathsf{disj}(\mathcal{S}_1,\cdot)$ is not constant. Clearly, since we assume $\ell_1, \ell_2 \neq 0$, any $\mathcal{S}_1 \in \binom{[m]}{k_1,\ell_1}$ must intersects at least one $\mathcal{S}_2 \in \binom{[m]}{k_2,\ell_2}$. Now, recall that we assume that either $\ell_1 + k_2 \le m$ or $k_1 + \ell_2 \le m$. Either way, it follows that $k_1 + k_2 \le m$. Therefore, for any $\mathcal{S}_1 \in \binom{[m]}{k_1}$ there exists $\mathcal{S}_2 \in \binom{[m]}{k_2}$ that does not intersects $\mathcal{S}_1$.

We now show that disj is not forced and hence cannot be computed with 1-security. Recall that for the negative direction, we assume that $k_1 \neq \ell_1$ or $k_2 \neq \ell_2$. Assume the former without loss of generality, and assume towards contradiction that disj is forced.

Then there exists two ensembles of efficiently samplable distributions $\mathcal{Q} = \{Q_\kappa\}_{\kappa\in\mathbb{N}}$ $\mathcal{R} = \{R_\kappa\}_{\kappa\in\mathbb{N}}$ such that, in particular

$$\left\{\mathsf{disj}\left(\mathcal{S}_1^* \cap \mathcal{S}_2\right)\right\}_{\kappa,\mathcal{S}_2,\mathcal{S}_2'} \stackrel{\mathrm{S}}{\equiv} \left\{\mathsf{disj}\left(\mathcal{S}_1^* \cap \mathcal{S}_2'\right)\right\}_{\kappa,\mathcal{S}_2,\mathcal{S}_2'}, \tag{14}$$

and

$$\left\{\mathsf{disj}\left(\mathcal{S}_1 \cap \mathcal{S}_2^*\right)\right\}_{\kappa,\mathcal{S}_1,\mathcal{S}_1'} \stackrel{\mathrm{S}}{\equiv} \left\{\mathsf{disj}\left(\mathcal{S}_1' \cap \mathcal{S}_2^*\right)\right\}_{\kappa,\mathcal{S}_1,\mathcal{S}_1'}, \tag{15}$$

where $\mathcal{S}_1^* \leftarrow Q_\kappa$ and $\mathcal{S}_2^* \leftarrow R_\kappa$. We next We next separate the proof into two cases.

Case 1: $\ell_1 + k_2 \le m$ and $k_1 + \ell_2 > m$. In this case, $\mathsf{disj}(\cdot,\mathcal{S}_2) \equiv 0$ for any $\mathcal{S}_2 \in \binom{[m]}{\ell_2}$, but $\mathsf{disj}(\mathcal{S}_1,\cdot) \not\equiv 0$ for any $\mathcal{S}_1 \in \binom{[m]}{k_1,\ell_1}$. Similarly to Claim 7.1, this immediately contradicts Equation (14).

Case 2: $k_1 + \ell_2 \le m$**.** We show that for $\mathcal{S}_1 \leftarrow \binom{[m]}{k_1}$ and $\mathcal{S}_1' \leftarrow \binom{[m]}{k_1+1}$ sampled independently, the statistical distance between $\mathsf{disj}\left(\mathcal{S}_1 \cap \mathcal{S}_2^*\right)$ and $\mathsf{disj}\left(\mathcal{S}_1' \cap \mathcal{S}_2^*\right)$ is not negligible. This implies that there exists $\mathcal{S}_1$ and $\mathcal{S}_1'$ for which the statistical distance is not negligible, thus Equation (15) does not hold.

Observe that for any $\mathcal{S}_2 \in \binom{[m]}{n}$, for some $n \in \{k_2, \ldots, \ell_2\}$, it holds that

$$\Pr_{\mathcal{S}_1 \leftarrow \binom{[m]}{k_1}}\left[\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset\right] = \Pr_{\mathcal{S}_1 \leftarrow \binom{[m]}{k_1}}\left[\mathcal{S}_1 \subseteq [m] \setminus \mathcal{S}_2\right] = \frac{\binom{m-n}{k_1}}{\binom{m}{k_1}}.$$

Similarly,

$$\Pr_{\mathcal{S}_1' \leftarrow \binom{[m]}{k_1+1}}\left[\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset\right] = \frac{\binom{m-n}{k_1+1}}{\binom{m}{k_1+1}}.$$

Let $d(n) := \frac{\binom{m-n}{k_1}}{\binom{m}{k_1}} - \frac{\binom{m-n}{k_1+1}}{\binom{m}{k_1+1}}$. Then for any $n \le m - k_1$ it holds that $d(n) > 0$. Since $k_1 + \ell_2 \le m$, it follows that $d(n) > 0$ for any $n \in \{k_2, \ldots, \ell_2\}$. Now, for every $n \in \{k_2, \ldots, \ell_2\}$ and every $\kappa \in \mathbb{N}$, let $p_{n,\kappa} = \Pr_{\mathcal{S}_2^* \leftarrow R_\kappa}[|\mathcal{S}_2^*| = n]$. Then

$$\Pr_{\mathcal{S}_1 \leftarrow \binom{[m]}{k_1}, \mathcal{S}_2^* \leftarrow R_\kappa}\left[\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset\right] - \Pr_{\mathcal{S}_1' \leftarrow \binom{[m]}{k_1+1}, \mathcal{S}_2^* \leftarrow R_\kappa}\left[\mathcal{S}_1' \cap \mathcal{S}_2 = \emptyset\right] = \sum_{n=k_2}^{\ell_2} p_{n,\kappa} \cdot d(n).$$

Since there exists n^* for which $p_{n^*,\kappa} \ge 1/n^*$ infinitely often, it follows that the above difference is at least $d(n^*)/n^*$, which is non-negligible. $\square$

Bibliography

[1] N. Agarwal, S. Anand, and M. Prabhakaran. Uncovering algebraic structures in the mpc landscape. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 381–406. Springer, 2019.

[2] B. Alon, R. Cohen, E. Omri, and T. Suad. On the power of an honest majority in three-party computation without broadcast. In *Theory of Cryptography Conference*, pages 621–651. Springer, 2020.

[3] G. Asharov. Towards characterizing complete fairness in secure two-party computation. In *Proceedings of the 11th Theory of Cryptography Conference(TCC)*, pages 291–316, 2014.

[4] G. Asharov, A. Beimel, N. Makriyannis, and E. Omri. Complete characterization of fairness in secure two-party computation of Boolean functions. In *Proceedings of the 12th Theory of Cryptography Conference(TCC), part I*, pages 199–228, 2015.

[5] S. Badrinarayanan, P. Miao, P. Mukherjee, and D. Ravi. On the round complexity of fully secure solitary mpc with honest majority. *Cryptology ePrint Archive*, 2021.

[6] A. Beimel, A. Gabizon, Y. Ishai, E. Kushilevitz, S. Meldgaard, and A. Paskin-Cherniavsky. Non-interactive secure multiparty computation. In *Annual Cryptology Conference*, pages 387–404. Springer, 2014.

[7] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for noncryptographic fault-tolerant distributed computations. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 1988.

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

[9] R. Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 364–369, 1986.

[10] R. Cohen and Y. Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. *Journal of Cryptology*, 30(4):1157–1186, 2017.

[11] R. Cohen, I. Haitner, E. Omri, and L. Rotem. Characterization of secure multiparty computation without broadcast. *Journal of Cryptology*, 31(2):587–609, 2018.

[12] D. Dachman-Soled. Revisiting fairness in MPC: polynomial number of parties and general adversarial structures. In R. Pass and K. Pietrzak, editors, *Proceedings of the 18th Theory of Cryptography Conference(TCC), part II*, volume 12551, pages 595–620. Springer, 2020.

[13] V. Daza and N. Makriyannis. Designing fully secure protocols for secure two-party computation of constant-domain functions. In *Proceedings of the 15th Theory of Cryptography Conference(TCC), part I*, pages 581–611, 2017.

[14] U. Feige, J. Killian, and M. Naor. A minimal model for secure computation. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 554–563, 1994.

[15] M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1(1):26–39, 1986.

[16] M. Fitzi, J. A. Garay, U. M. Maurer, and R. Ostrovsky. Minimal complete primitives for secure multiparty computation. *Journal of Cryptology*, 18(1):37–61, 2005.

[17] O. Goldreich. *Foundations of Cryptography – VOLUME 2: Basic Applications*. Cambridge University Press, 2004.

[18] S. D. Gordon and J. Katz. Complete fairness in multi-party computation without an honest majority. In *Proceedings of the 6th Theory of Cryptography Conference(TCC)*, pages 19–35, 2009.

[19] S. D. Gordon, C. Hazay, J. Katz, and Y. Lindell. Complete fairness in secure two-party computation. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 413–422, 2008.

[20] S. Halevi, Y. Ishai, E. Kushilevitz, N. Makriyannis, and T. Rabin. On fully secure MPC with solitary output. In *Proceedings of the 17th Theory of Cryptography Conference(TCC), part I*, pages 312–340, 2019.

[21] J. Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 20–31, 1988.

[22] L. Lamport, R. E. Shostak, and M. C. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[23] N. Makriyannis. On the classification of finite Boolean functions up to fairness. In *Proceedings of the 9th Conference on Security and Cryptography for Networks (SCN)*, pages 135–154, 2014.

[24] M. C. Pease, R. E. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.

[25] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 73–85, 1989.

A Definition of Security-With-Identifiable-Abort

We next define an ideal computation with *security-with-identifiable-abort*, where a trusted party performs the computation on behalf of the parties, and where the ideal-model adversary can abort the computation after learning the output, but at the expense of revealing the identity of a corrupted party. An ideal computation of a three-party functionality $f = (f_1, f_2, f_3)$, with $f_1, f_2, f_3 : (\{0,1\}^*)^3 \to \{0,1\}^*$, on inputs $x, y, z \in \{0,1\}^*$ and security parameter κ, with an ideal-world adversary $\mathcal{A}$ running with an auxiliary input aux and corrupting a (strict) subset $\mathcal{I} \subseteq \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ of the parties proceeds as follows:

Parties send inputs to the trusted party: Each honest party sends its input to the trusted party. For each corrupted party, the adversary $\mathcal{A}$ sends a value v from the corresponding domain as the input for the corrupted party. Let (x', y', z') denote the inputs received by the trusted party.

The trusted party performs computation: The trusted party selects a random string r, computes $(w_\mathsf{A}, w_\mathsf{B}, w_\mathsf{C}) = f(x', y', z'; r)$, and sends $\{w_\mathsf{P}\}_{\mathsf{P} \in \mathcal{I}}$ to $\mathcal{A}$.

Malicious adversary instructs trusted party to continue or halt: The adversary $\mathcal{A}$ sends either $\mathsf{continue}$ or $(\mathsf{abort}, \mathsf{P})$ for some $\mathsf{P} \in \mathcal{I}$ to T. If it sent $\mathsf{continue}$, then for every honest party Q the trusted party sends it w_Q. Otherwise, if $\mathcal{A}$ sent $(\mathsf{abort}, \mathsf{P})$, then T sends $(\mathsf{abort}, \mathsf{P})$ to the each honest party Q.

Outputs: Each honest party outputs whatever output it received from the trusted party and the corrupted parties output nothing. The adversary $\mathcal{A}$ outputs some function of its view (i.e., the auxiliary input, its randomness, and the input and output of the corrupted parties).