Complete Characterization of Fairness in Secure Two-Party Computation of Boolean Functions

Gilad Asharov* Amos Beimel† Nikolaos Makriyannis‡ Eran Omri§

December 16, 2014

Abstract

Fairness is a desirable property in secure computation; informally it means that if one party gets the output of the function, then all parties get the output. Alas, an implication of Cleve's result (STOC 86) is that when there is no honest majority, in particular in the important case of the two-party setting, there exist functions that cannot be computed with fairness. In a surprising result, Gordon et al. (JACM 2011) showed that some interesting functions can be computed with fairness in the two-party setting, and re-opened the question of understanding which Boolean functions can be computed with fairness, and which cannot.

Our main result in this work is a complete characterization of the (symmetric) Boolean functions that can be computed with fairness in the two-party setting; this settles an open problem of Gordon et al. The characterization is quite simple: A function can be computed with fairness *if and only if* the all one-vector or the all-zero vector are in the affine span of either the rows or the columns of the matrix describing the function. This is true for both deterministic and randomized functions. To prove the possibility result, we modify the protocol of Gordon et al. (JACM 2011); the resulting protocol computes with full security (and in particular with fairness) all functions that are computable with fairness.

We extend the above result in two directions. First, we completely characterize the Boolean functions that can be computed with fairness in the multiparty case, when the number of parties is constant and at most half of the parties can be malicious. Second, we consider the two-party setting with asymmetric Boolean functionalities, that is, when the output of each party is one bit, but the outputs are not necessarily the same. We generalize our aforementioned protocol for symmetric functions to handle asymmetric functions, and obtain a sufficient condition for computing such functions with fairness. In addition, we provide a necessary condition for fairness; however, a gap is left between these two conditions. We then consider a specific asymmetric function in this gap area, and by designing a new protocol, we show that it is computable with fairness. However, we do not give a complete characterization for all functions that lie in this gap, and their classification remains open.

Keywords: Fairness, secure two-party computation, foundations, malicious adversaries

*School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem, Israel. Supported by the Israeli Centers of Research Excellence (I-CORE) Program (Center No. 4/11).

†Department of Computer Science, Ben Gurion University, Be'er Sheva, Israel. Partially supported by ISF grant 544/13 and by the Frankel Center for Computer Science.

‡Departament de Tecnologies de la Informació i les Comunicacions, Universitat Pompeu Fabra, Barcelona, Spain.

§Department of Computer Science and Mathematics, Ariel University, Ariel, Israel. Partially supported by ISF grant 544/13.

1 Introduction

Secure multiparty computation is one of the gems of modern cryptography. It enables a set of mutually distrusting parties to securely compute some joint function of their inputs in the presence of an adversarial behaviour. The security requirements of such a computation include privacy, correctness, independence of inputs, and *fairness*. Informally, fairness means that if one party gets the output of the function, then all parties get the output. For example, when two parties are signing a contract, it is reasonable to expect that one party signs the contract if and only if the second party signs the contract.

The study of secure multiparty protocols (MPC) started with the works of Yao [13] for the two-party setting and Goldreich, Micali, and Wigderson [9] for the multiparty setting. When a strict majority of honest parties can be guaranteed, protocols for secure computation provide full security, i.e., they provide all the security properties mentioned above including fairness. However, this is no longer the case when there is no honest majority, and in particular in the case of two-party computation where one of parties may be corrupted. In these settings, protocols for secure computation provide a weaker notion of security, which is known as "security with abort". Specifically, these protocols still provide important security requirements such as correctness and privacy, but can not guarantee fairness – and the adversary can get its output while preventing the honest parties from getting their output. Relaxing the security requirement when there is no honest majority is unavoidable as was shown by Cleve [7].

To elaborate further, Cleve [7] proved that there exist two-party functions that cannot be computed with fairness when there is no honest majority. In particular, he showed that the coin-tossing functionality, where two parties toss an unbiased fair coin, cannot be computed with complete fairness. He proved that in any two-party coin-tossing protocol there exists an adversary that can bias the output of the honest party. A ramification of Cleve's impossibility is that any function that implies coin-tossing cannot be computed with full security without an honest majority. An example for such a function is the exclusive-or (XOR) function; a fully secure coin-tossing protocol can be easily constructed assuming the existence of a fair protocol for XOR.

For years, the common interpretation of Cleve's impossibility was that no interesting functions can be computed with full security without an honest majority. In a recent surprising result, Gordon et al. [10] showed that this interpretation is inaccurate, and there exist interesting functions can indeed be computed with full security in the two-party setting, e.g., the millionaires' problem with polynomial size domain. This result re-opened the question of understanding which Boolean functions can be computed with fairness and which cannot.

In more detail, Gordon et al. [10] showed that all functions with polynomial size domain that do not contain an embedded XOR can be computed with full security in the two-party setting; this class of functions contains the AND function and Yao's millionaires' problem. They also presented a protocol, later referred to as the GHKL protocol, which computes with full security a large class of Boolean functions containing embedded XORs. However, the analysis of this protocol is rather involved, and the exact class of function that can be computed using this protocol was unclear.

In this paper, we focus on the characterization of fairness for Boolean functions, and provide a *complete* characterization of Boolean two-party functions that can be computed with full security.

1.1 Previous Works

As mentioned above, Cleve [7] proved that coin-tossing cannot be computed with full security without an honest majority, and in particular, in the two-party setting. A generalization of Cleve's result was given recently by Agrawal and Prabhakaran [1]. They showed that any non-trivial sampling functionality cannot be computed with fairness and correctness in the two-party setting, where a non-trivial sampling functionality is a randomized functionality in which two parties, with no inputs, sample two correlated bits.

Gordon et al. [10] re-opened the question of characterizing fairness in secure two-party and multiparty computation. This question was studied in a sequence of works [2, 3, 12]. Asharov, Lindell, and Rabin [3] focused on the work of Cleve [7] and fully identified the class of functions that imply fair coin-tossing, and are thus ruled out by Cleve's impossibility. Later, Asharov [2] studied the functions that can be computed with full security using the GHKL protocol. He identified three classes of functions: Functions that can be computed using the GHKL protocol, functions that cannot be computed with full security using this specific protocol, and a third class of functions that remained unresolved. Intuitively, [2] showed that if a function is somewhat asymmetric, in the sense that, in the ideal model, one party exerts more influence on the output compared to the other party, then the function can be computed with full security.

Makriyannis [12] has recently shown that the class of functions that by [2] cannot be compute fairly using the GHKL protocol is inherently unfair. He showed a beautiful reduction from sampling functionalities, for which fair computation had already been ruled out in [1], to any function in this class. Indeed, in this class of functions, the influence that each party exerts over the output in the ideal world is somewhat the same. However, the works of [2, 12] left the aforementioned third class of functions unresolved. Specifically, this class of functions is significantly different than the class of functions that was shown to be fair, and it contains functions where the parties exert the same amount of influence over the output in the ideal model and yet do not imply sampling, at least not by the construction of [12].

The characterization of fairness was studied in scenarios other than symmetric two-party Boolean functions where the output of the parties is the same. In particular, Gordon and Katz [11] considered fully-secure computation in the multiparty setting without an honest majority and constructed a fully-secure three-party protocol for the majority function and an m-party protocol for the AND of m bits. Asharov [2] also studied asymmetric functions where the parties' outputs are not necessarily the same, as well as functions with non-Boolean outputs, and showed some initial possibility results for these classes of functions.

1.2 Our Results

In this work, we study when functions can be computed with full security without an honest majority. Our main result in this work is a complete characterization of the Boolean function that can be computed with full security in the two-party setting. This solves an open problem of [10]. We focus on the third class of functions that was left unresolved by [2, 12], and show that *all* functions in this class can be computed with full security (thus, with fairness). This includes functions where the parties' influences on the output in the ideal world are completely equalized, showing that the classification is more subtle than one might have expected.

In order to show possibility for this class of functions, we provide a new protocol (based on the protocol of GHKL), and show that it can compute all the functions in this class. We note that the GHKL protocol had to be modified in order to show this possibility; In particular, we show that there exist functions in this class that cannot be computed using the GHKL protocol for any set of parameters, but can be computed using our modified protocol (see Example 1.2). In addition, this protocol computes with full security all functions that were already known to be fair. Combining the result with the impossibility result of [12], we obtain a quite simple characterization for fairness:

Theorem 1.1 (informal). *A Boolean function $f : X \times Y \to \{0,1\}$ can be computed with full security* if and only if *the all one-vector or the all-zero vector are in the affine span of either the rows or the columns of the matrix describing the function.*

The above informally stated theorem is true for both deterministic and randomized functions. Alternatively, our characterization can be stated as follows: either a function implies non-trivial sampling, thus cannot be computed with fairness, or the function can be computed with full security (and, in particular, with complete fairness), assuming the existence of an oblivious transfer protocol.

	y_1	y_2	y_3	y_4
x_1	0	0	0	1
x_2	0	0	1	1
x_3	0	1	1	0
x_4	1	1	0	1

Figure 1: A function that can be computed with our protocol, and cannot be computed using [10]

Example 1.2. The function whose matrix is described in Figure 1 is a concrete example of a function that was left as unresolved in [2, 12]. The all-one vector is an affine combination of the rows of the matrix described in Figure 1 (taking the first row with coefficient -1, the second and the forth with coefficient 1, and the third row with coefficient 0; the sum of the coefficients is 1, as required by an affine combination). Thus, this function can be computed with full security by our protocol. However, we show that the GHKL protocol is susceptible to an attack for this particular function.

We extend the above result in two directions. First, we completely characterize the Boolean functions that can be computed with full security when the number of parties is constant and at most half of the parties can be malicious. We show that a function can be computed with full security when at most half of the parties can be malicious if and only if every partition of the parties' inputs into two equal sets results in a fully-secure two-party function. Second, we consider the two-party setting with asymmetric Boolean functionalities, that is, when the output of each party is one bit, but the outputs are not necessarily the same. We generalize our aforementioned protocol for symmetric functions to handle asymmetric functions, and conclude with a sufficient condition for fairness. In addition, we provide a necessary condition for fairness; however, a gap is left between these two conditions. For the functions that lie in the gap, the characterization remains open. We then consider a specific function in this gap area and, by designing a new protocol, we show that it is computable with fairness. This new protocol has some different properties than the generalized protocol, which may imply that the characterization of Boolean asymmetric functions is more involved than the symmetric case.

2 Preliminaries

In this paper all vectors are column vectors over $\mathbb{R}$. Vectors are denoted by bold letters, e.g., $\mathbf{v}$ or $\mathbf{1}$ (the all-one vector). Let $\mathcal{V} \subseteq \mathbb{R}^\ell$ be a set of vectors. A vector $\mathbf{w} \in \mathbb{R}^\ell$ is an *affine combination* of the vectors in $\mathcal{V}$ if there exist scalars $\{a_\mathbf{v} \in \mathbb{R}\}_{\mathbf{v} \in \mathcal{V}}$ such that $\mathbf{w} = \sum_{\mathbf{v} \in \mathcal{V}} a_\mathbf{v} \cdot \mathbf{v}$ and $\sum_{\mathbf{v} \in \mathcal{V}} a_\mathbf{v} = 1$. Let M be a $\ell \times k$ real matrix. A vector $\mathbf{w} \in \mathbb{R}^\ell$ is in the image of M, denoted $\mathbf{w} \in \operatorname{im}(M)$, if there exists a vector $\mathbf{u} \in \mathbb{R}^k$ such that $M\mathbf{u} = \mathbf{w}$. A vector $\mathbf{v} \in \mathbb{R}^k$ is in the kernel of M, denoted $\mathbf{v} \in \ker(M)$, if $M\mathbf{v} = \mathbf{0}_\ell$.

The affine hull of a set $\mathcal{V} \in \mathbb{R}^\ell$, denoted $\textsf{affine-hull}(\mathcal{V})$, is the smallest affine set containing $\mathcal{V}$, or equivalently, the intersection of all affine sets containing $\mathcal{V}$. This is equivalent to the set of all affine combinations of elements in $\mathcal{V}$, that is, $\textsf{affine-hull}(\mathcal{V}) = \{\sum_{\mathbf{v} \in \mathcal{V}} a_\mathbf{v} \cdot \mathbf{v} : \sum_{\mathbf{v} \in \mathcal{V}} a_\mathbf{v} = 1\}$.

Proposition 2.1. *$\mathbf{1}_\ell$ is not a linear combination of the columns of M iff $(\mathbf{0}_k)^T$ is an affine combination of the rows of M.*

Proof. $\mathbf{1}_\ell$ is not a linear combination of the columns of M iff $\operatorname{rank}(\big(M|1\big)) = \operatorname{rank}(M) + 1$ (where $\big(M|1\big)$ is the matrix M with the extra all-one column) iff $\dim(\ker(\big(M|1\big)^T)) = \dim(\ker(M^T)) - 1$ iff there exists a vector $\mathbf{u} \in \mathbb{R}^\ell$ such that $M^T\mathbf{u} = \mathbf{0}_k$ and $\mathbf{1} \cdot \mathbf{u} = 1$ iff $(\mathbf{0}_k)^T$ is an affine combination of the rows of M. $\square$

Definition 2.2. *Given two $\ell \times k$ matrices matrices A and B, we say that $C = A * B$ if C is the entry-wise (Hadamard) product of the matrices, that is, $C_{i,j} = A_{i,j} \cdot B_{i,j}$.*

2.1 Secure Multiparty Computation

In this section we define secure multiparty computation, where the definitions are the standard and based on [6,8]. Let $n \in \mathbb{N}$ denote the security parameter. A function $\mu(\cdot)$ is *negligible* if it vanishes faster than any (positive) inverse-polynomial. A distribution ensemble $X = \{X(a,n)\}_{a \in \Delta, n \in \mathbb{N}}$ is an infinite sequence of random variables indexed by Δ and $\mathbb{N}$. Two distribution ensembles, X and Y, are *computationally indistinguishable* if for every non-uniform polynomial-time algorithm D, there exists a negligible function μ such that for every a and n

$$|\Pr[D(X(a,n)) = 1] - \Pr[D(Y(a,n)) = 1]| \leq \mu(n).$$

Furthermore, we say that X and Y are *statistically close* if for all a and n, the following sum is upper-bounded by a negligible function in n:

$$\frac{1}{2} \cdot \sum_s |\Pr[X(a,n) = s] - \Pr[Y(a,n) = s]|,$$

where s ranges over the support of either $X(a,n)$ or $Y(a,n)$. We write $X \overset{c}{\equiv} Y$ when the ensembles are computationally indistinguishable and $X \overset{s}{\equiv} Y$ when they are statistically close.

Let $P_1, P_2, \ldots, P_m$ denote the parties. An *m-party functionality* $f = (f_1, \ldots, f_m)$ is a random process such that f maps m-tuples of inputs (one for each party) to m-tuples of random variables called outputs (one output for each party). The domain of f is denoted by $X = X_1 \times \cdots \times X_m$. When the functionality is deterministic we refer to it as a function. When $f_1 = \cdots = f_m$, we refer to f as symmetric, and denote f as the output of each party (rather than the m identical outputs). If the m functions are not all equal, we refer to f as an asymmetric functionality.

An *m-party protocol* Π, for computing a functionality f, is a polynomial-time protocol such that on a global input 1^n – the security parameter – and on private inputs $x_1 \in X_1, \ldots, x_m \in X_m$, the joint distribution of the outputs of honest executions of $\{\Pi(1^n, x_1, \cdots, x_m)\}_{n\in\mathbb{N}}$ is statistically close to $f(x_1, \cdots, x_m) = (f_1(x_1, \ldots, x_m), \ldots, f_m(x_1, \cdots, x_m))$. The parties in the protocol run in polynomial time in the security parameter n (we consider f as a fixed, finite function).

The Adversary. Following the usual convention, we introduce an adversary $\mathcal{A}$, which is given an auxiliary input z and corrupts a subset of the parties. The adversary is static, that is, it chooses the subset it corrupts prior to the execution of the protocol. The adversary is assumed to be malicious and computationally bounded. For a protocol Π computing f, let

$$(\mathrm{Out}^{\mathsf{Real}}_{\mathcal{A}(z),\Pi}, \mathrm{View}^{\mathsf{Real}}_{\mathcal{A}(z),\Pi})(1^n, x_1, \ldots, x_m)$$

denote the joint distribution of the honest parties' outputs and the adversary's view during an execution of Π, where $x_1, \ldots, x_m$ are the prescribed inputs, 1^n is the security parameter (in unary representation), and the view of the adversary contains its auxiliary input, the inputs of the parties it controls, their random inputs, and the messages they get during the execution of the protocol.

2.1.1 The Ideal Model Paradigm

Let $f = (f_1, \ldots, f_m)$ be an m-party functionality and let Π be a protocol for computing f. Further, assume that an adversary corrupts a fixed subset $B \subseteq \{P_1, \ldots, P_m\}$ of the parties. Security in multiparty computation is defined via an *ideal model.* Namely, we assume that parties have access to a trusted party $\mathcal{T}$ that performs the computation of the functionality for them, and we attempt to show that protocol Π *emulates* this ideal scenario. In Figure 2, we describe the *ideal model with full security.*

Inputs: Each party P_i holds 1^n and $x_i \in X_i$. The adversary is given an auxiliary input $z \in \{0, 1\}^*$. The trusted party $\mathcal{T}$ has no input.

Parties send inputs: Each honest party sends its input to $\mathcal{T}$, each corrupted party sends a value of the adversary's choice. Write $(x'_1, \ldots, x'_m)$ for the tuple of inputs received by $\mathcal{T}$.

The trusted party performs computation: If any x'_i is not in the appropriate domain (or was not sent at all), then $\mathcal{T}$ reassigns the aberrant input to some default value. Write $(x'_1, \ldots, x'_m)$ for the tuple of inputs after (possible) reassignment. The trusted party then chooses a random string r and computes $f(x'_1, \ldots, x'_m; r)$.

Trusted party sends outputs: Each party P_i receives $f_i(x'_1, \ldots, x'_m; r)$.

Outputs: Each honest party outputs whatever $\mathcal{T}$ sent him, the corrupted parties output nothing, and the adversary outputs a probabilistic polynomial-time function of its view.

Figure 2: The Ideal model with full security.

Let $\mathcal{S}$ be an adversary in the ideal world, which is given auxiliary input z and corrupts a subset B of the parties. Denote the honest parties' outputs and the adversary's view by $(\mathrm{Out}^{\mathsf{Ideal}}_{\mathcal{S}(z),f}, \mathrm{View}^{\mathsf{Ideal}}_{\mathcal{S}(z),f})(1^n, x_1, \ldots, x_m)$, where $x_1, \ldots, x_m$ are the prescribed inputs and 1^n is the security parameter. We now define security with respect to the ideal model with full security.

Definition 2.3. *Let* Π *be a protocol for computing* f. *We say that* Π *computes* f with full security *tolerating coalitions of size at most* t *if for every non-uniform polynomial time adversary* $\mathcal{A}$ *controlling a set* B *of at most* t *parties in the real model, there exists a non-uniform polynomial time adversary* $\mathcal{S}$ *(called the simulator) controlling* B *in the ideal model such that*

$$\left\{\left(\mathrm{Out}^{\mathsf{Real}}_{\mathcal{A}(z),\Pi}, \mathrm{View}^{\mathsf{Real}}_{\mathcal{A}(z),\Pi}\right)(1^n, x_1, \dots, x_m)\right\}_{(x_1,\dots,x_m)\in X, z\in\{0,1\}^*, n\in\mathbb{N}}$$
$$\stackrel{c}{\equiv} \left\{\left(\mathrm{Out}^{\mathsf{Ideal}}_{\mathcal{S}(z),f}, \mathrm{View}^{\mathsf{Ideal}}_{\mathcal{S}(z),f}\right)(1^n, x_1, \dots, x_m)\right\}_{(x_1,\dots,x_m)\in X, z\in\{0,1\}^*, n\in\mathbb{N}} .$$

3 Characterization of Fairness for Boolean Two-Party Functions

In this section we provide a complete characterization of the Boolean functions that can be computed with full security. Let $f : X \times Y \to \{0, 1\}$ be a finite Boolean function with the associated $\ell \times k$ matrix M. Without loss of generality, $X = \{1, \dots, \ell\}$ and $Y = \{1, \dots, k\}$. In Figure 3 we describe Protocol FairTwoParty$_\sigma$, a modification of the GHKL protocol [10]. Protocol FairTwoParty$_\sigma$ is described using an on-line dealer.

Remark 3.1. Protocol FairTwoParty$_\sigma$ has a special structure that can be used to transform it to a protocol without a dealer. That is, the protocol is composed of a fixed number r of rounds (which depends on the security parameter and the function computed by the protocol), the dealer prepares $2r$ values $a_1, \dots, a_r$ and $b_1, \dots, b_r$ and in round i it first gives a_i to P_1, if P_1 does not abort it gives b_i to P_2. The values a_i and b_i are called backup outputs and the respective value is outputted by a party if the other party aborts. If one of the parties aborts in round i, then the dealer aborts; otherwise it continues to round $i + 1$. Using, by now, standard techniques, and assuming that a secure protocol for OT exists,[1] one can transform this protocol to a protocol without a dealer. This transformation includes executing a two-party protocol, guaranteeing only security with abort, computing authenticated 2-out-of-2 shares of $a_1, \dots, a_r$ and $b_1, \dots, b_r$, and then in round i, party P_2 sends to P_1 its authenticated share of a_i and then party P_1 sends to P_2 its authenticated share of b_i. A standard proof shows that if the protocol with the on-line dealer guarantees full security, then the resulting protocol guarantees full security. See further discussion in Section 5.

The main difference in Protocol FairTwoParty$_\sigma$ compared to the GHKL protocol is in Step 3, where $b_{i^*-1} = \sigma$ (compared to $b_{i^*-1} = f(x, \widetilde{y})$ for a random $\widetilde{y}$ in the GHKL protocol). For some functions f we choose $\sigma = 0$ and for some we choose $\sigma = 1$; the choice of σ depends only on the function and is independent of the inputs. This seemingly small change enables to compute with full security a larger class of functions, i.e., all Boolean functions that can be computed with full security. We note that this change is somewhat counter intuitive as we achieve security against a malicious P_1 by giving P_2 less information. The reason why this change works is that it enables the simulator for P_1 to choose its input to the trusted party in a clever way.

Remark 3.2. There are two parameters in Protocol FairTwoParty$_\sigma$ that are unspecified – the parameter α of the geometric distribution and the number of rounds r. We show that there exists

[1] As in OT only one party gets the output, it suffices that the OT protocols is secure-with abort.

Protocol $\textsc{FairTwoParty}_\sigma$

1. The parties P_1 and P_2 hand their inputs, denoted x and y respectively, to the dealer.[a]
2. The dealer chooses $i^* \geq 2$ according to the geometric distribution with probability α.
3. The dealer computes $\mathsf{out} = f(x, y)$, and for $0 \leq i \leq r$

$$a_i = \begin{cases} f(x, \widetilde{y}^{(i)}) \text{ where } \widetilde{y}^{(i)} \in_U Y & \text{if } i < i^* \\ \mathsf{out} & \text{otherwise} \end{cases}$$

and

$$b_i = \begin{cases} f(\widetilde{x}^{(i)}, y) \text{ where } \widetilde{x}^{(i)} \in_U X & \text{if } i < i^* - 1 \\ \sigma & \text{if } i = i^* - 1 \\ \mathsf{out} & \text{otherwise.} \end{cases}$$

4. The dealer gives b_0 to P_2.
5. For $i = 1, \ldots, r$,
   (a) The dealer gives a_i to P_1. If P_1 aborts, then P_2 outputs b_{i-1} and halts.
   (b) The dealer gives b_i to P_2. If P_2 aborts, then P_1 outputs a_i and halts.

[a]If x is not in the appropriate domain or P_1 does not hand an input, then the dealer sends $f(\hat{x}, y)$ (where $\hat{x}$ is a default value) to P_2, which outputs this value and the protocol is terminated. The case of an inappropriate y is dealt analogously.

Figure 3: Protocol $\textsc{FairTwoParty}_\sigma$ for securely computing a function f, where σ is either 0 or 1.

a constant α_0 (which depends on f) such that taking any $\alpha \leq \alpha_0$ guarantees full security (provided that f satisfies some conditions). As for the number of rounds r, even if both parties are honest the protocol fails if $i^* > r$ (where i^* is chosen with geometric distribution). The probability that $i^* > r$ is $(1 - \alpha)^r$. So, if $r = \alpha^{-1} \cdot \omega(\log n)$ (where n is the security parameter), the probability of not reaching i^* is negligible.

Theorem 3.3. *Let M be the associated matrix of a Boolean function f. If $(\mathbf{0}_k)^T$ is an affine combination of the rows of M, then there is a constant $\alpha_0 > 0$ such that Protocol $\textsc{FairTwoParty}_0$ with $\alpha \leq \alpha_0$ is a fully-secure protocol for f.*

Proof. It is easy to see that the protocol is secure against a corrupted P_2, using a simulator similar to [10]. Intuitively, this follows directly from the fact that P_2 always gets the output after P_1.

We next prove that the protocol is secure against a corrupted P_1 by constructing a simulator for every adversary $\mathcal{A}$ in the real world controlling P_1. The simulator we construct has only black-box access to $\mathcal{A}$ and we denote it by $\mathcal{S}^{\mathcal{A}}$. The simulation is described in Figure 4. It operates along the same lines as in the proof of [10]. The main difference is in the analysis of the simulator, where we prove that there exist distributions $(\mathbf{x}_x^{(a)})_{x \in X, a \in \{0,1\}}$, which are used by the simulator to choose the input that P_1 gives to the trusted party.

We next prove the correctness of the simulator, that is, if $(\mathbf{0}_k)^T$ is an affine combination of the rows of M, then the output of the simulator and and the output of P_2 in the ideal world are distributed as the adversary's view and the output of the honest P_2 in the real world. The simulator

The simulator $\mathcal{S}^{\mathcal{A}}$ for Protocol $\textsc{FairTwoParty}_0$

- The adversary $\mathcal{A}$ gives its input x to the simulator.[a]
- The simulator chooses $i^* \geq 2$ according to the geometric distribution with probability α.
- For $i = 1, \ldots, i^* - 1$:
  - The simulator gives $a_i = f(x, \widetilde{y}^{(i)})$ to the adversary $\mathcal{A}$, where $\widetilde{y}^{(i)}$ is chosen according to the uniform distribution.
  - If $\mathcal{A}$ aborts, then the simulator chooses an input x_0 according to a distribution $\mathbf{x}_x^{(a_i)}$ (which depends on the input x and the last bit that was chosen), gives x_0 to the trusted party, outputs the bits $a_1, \ldots, a_i$, and halts.
- At round $i = i^*$, the simulator gives x to the trusted party and gets the output $a = f(x, y)$.
- For $i = i^*, \ldots, r$: The simulator gives $a_i = a$ to the adversary $\mathcal{A}$, if $\mathcal{A}$ aborts, then the simulator outputs the bits $a_1, \ldots, a_i$ and halts.
- The simulator outputs the bits $a_1, \ldots, a_r$ and halts.

[a]If the adversary gives an inappropriate x (or no x), then the simulator sends some default $\hat{x} \in X$ to the trusted party, outputs the empty string, and halts.

Figure 4: The simulator $\mathcal{S}^{\mathcal{A}}$ for Protocol $\textsc{FairTwoParty}_0$.

generates its output as the view of P_1 in the execution of the protocol in the real world. First, it chooses i^* as in Protocol $\textsc{FairTwoParty}_0$, i.e., with geometric distribution. Up to round $i^* - 1$, the backup outputs are uncorrelated to the input of the honest party. That is, for all $i < i^*$ the output $a_i = f(x, \widetilde{y})$ is chosen with a uniformly random $\widetilde{y}$. Starting with round $i = i^*$, the backup outputs are correlated to the true input of the honest party, and are set as $a_i = f(x, y)$, exactly as in the real execution.

As a result the adversary seeing the same view, $\mathcal{A}$ aborts in the simulation if and only if it aborts in the protocol. Furthermore, if the adversary aborts after round i^*, the output of P_2 in the protocol and in the simulator is identical – $f(x, y)$. The only difference between the simulation and the protocol is the way that the output of P_2 is generated when the adversary aborts in a round $i \leq i^*$. If the adversary aborts in round i^*, the output of P_2 in Protocol $\textsc{FairTwoParty}_0$ is $b_{i^*-1} = 0$, while the the output of P_2 in the simulation is $f(x, y)$. To compensate for this difference, in the simulation the output of P_2 if the adversary aborts in a round $1 \leq i \leq i^* - 1$ is $f(x_0, y)$, where x_0 is chosen according to a distribution $\mathbf{x}_x^{(a_i)}$ to be carefully defined later in the proof.

To conclude, we only have to compare the distributions $(\text{View}_{\mathcal{A}}, \text{Out}_{P_2})$ in the real and ideal worlds given that the adversary aborts in a round $1 \leq i \leq i^*$. Thus, in the rest of the proof we let i be the round in which the adversary aborts and assume that $1 \leq i \leq i^*$. The view of the adversary in both worlds is $a_1, \ldots, a_i$. Notice that in both worlds $a_1, \ldots, a_{i-1}$ are equally distributed and are independent of (a_i, Out_{P_2}). Thus, we only compare the distribution of (a_i, Out_{P_2}) in both worlds.

First, we introduce the following notation

$$(1/\ell, \ldots, 1/\ell)M = (s_1, \ldots, s_k), \qquad M \begin{pmatrix} 1/k \\ \vdots \\ 1/k \end{pmatrix} = \begin{pmatrix} p_1 \\ \vdots \\ p_\ell \end{pmatrix}. \tag{1}$$

For example, s_y is the probability that $f(\widetilde{x}, y) = 1$, when $\widetilde{x}$ is uniformly distributed. Furthermore, for every $x \in X$, $y \in Y$, and $a \in \{0,1\}$, define $q_x^{(a)}(y) \triangleq \Pr\left[f(x_0, y) = 1\right]$, where x_0 is chosen according to the distribution $\mathbf{x}_x^{(a)}$. That is, $q_x^{(a)}(y)$ is the probability that the output of P_2 in the simulation is 1 when the the adversary aborts in round $i < i^*$, the input of P_1 is x, the input of P_2 is y, and $a_i = a$. Finally, define the column vector $\mathbf{q}_x^{(a)} \triangleq (q_x^{(a)}(y))_{y \in Y}$. Using this notation,

$$M^T \mathbf{x}_x^{(a)} = \mathbf{q}_x^{(a)}, \tag{2}$$

where we represent the distribution $\mathbf{x}_x^{(a)} = (\mathbf{x}_x^{(a)}(x_0))_{x_0 \in X}$ by a column vector. Next, we analyze the four options for the values of $(a_i, \mathrm{Out}_{P_2})$.

First case: $(a_i, \mathrm{Out}_{P_2}) = (0,0)$**.** In the real world $(a_i, \mathrm{Out}_{P_2}) = (0,0)$ if one of the following two events occurs:

- $i < i^*$, $a_i = f(x, \widetilde{y}) = 0$, and $\mathrm{Out}_{P_2} = b_{i-1} = f(\widetilde{x}, y) = 0$. The probability of this event is $(1-\alpha)(1-p_x)(1-s_y)$.
- $i = i^*$, $a_{i^*} = f(x,y) = 0$, and $\mathrm{Out}_{P_2} = b_{i^*-1} = 0$. Recall that in Protocol $\textsc{FairTwoParty}_0$ $b_{i^*-1} = 0$ with probability 1. The probability of this event is $\alpha \cdot (1 - f(x,y)) \cdot 1$ (that is, it is 0 if $f(x,y) = 1$ and it is α otherwise).

Therefore, in the real world $\Pr\left[(a_i, \mathrm{Out}_{P_2}) = (0,0)\right] = (1-\alpha)(1-p_x)(1-s_y) + \alpha(1-f(x,y))$. On the other hand, in the ideal world $(a_i, \mathrm{Out}_{P_2}) = (0,0)$ if one of the following two events occurs:

- $i < i^*$, $a_i = f(x, \widetilde{y}) = 0$, and $\mathrm{Out}_{P_2} = f(x_0, y) = 0$. The probability of this event is $(1-\alpha)(1-p_x)(1-q_x^{(0)}(y))$.
- $i = i^*$, $a_{i^*} = f(x,y) = 0$, and $\mathrm{Out}_{P_2} = f(x,y) = 0$. The probability of this event is $\alpha \cdot (1 - f(x,y))$.

Therefore, in the ideal world $\Pr\left[(a_i, \mathrm{Out}_{P_2}) = (0,0)\right] = (1-\alpha)(1-p_x)(1-q_x^{(0)}(y)) + \alpha(1-f(x,y))$. To get full security we need that these two probabilities in the two worlds are the same, that is,

$$(1-\alpha)(1-p_x)(1-s_y) + \alpha(1-f(x,y)) = (1-\alpha)(1-p_x)(1-q_x^{(0)}(y)) + \alpha(1-f(x,y)),$$

i.e.,

$$q_x^{(0)}(y) = s_y. \tag{3}$$

As this is true for every y, we deduce, using Equation (1) and Equation (2), that

$$M^T \mathbf{x}_x^{(0)} = \mathbf{q}_x^{(0)} = M^T \left(1/\ell, \cdots, 1/\ell\right)^T. \tag{4}$$

Thus, taking the uniform distribution, i.e., for every $x_0 \in X$

$$\mathbf{x}_x^{(0)}(x_0) = 1/\ell, \tag{5}$$

satisfies these constraints.

Second case: $(a_i, \mathrm{Out}_{P_2}) = (0,1)$. In the real world $\Pr\left[(a_i, \mathrm{Out}_{P_2}) = (0,1)\right] = (1-\alpha)(1-p_x)s_y$ (in the real world $\mathrm{Out}_{P_2} = 0$ when $i = i^*$). On the other hand, in the ideal world $\Pr\left[(a_i, \mathrm{Out}_{P_2}) = (0,1)\right] = (1-\alpha)(1-p_x)q_x^{(0)}(y)$ (in the ideal world $a_{i^*} = \mathrm{Out}_{P_2} = f(x,y)$). The probabilities in the two worlds are equal if Equation (3) holds (i.e., Equation (5) holds) for every x.

Third case: $(a_i, \mathrm{Out}_{P_2}) = (1,0)$. In the real world $\Pr\left[(a_i, \mathrm{Out}_{P_2}) = (1,0)\right] = (1-\alpha)p_x(1-s_y) + \alpha \cdot f(x,y) \cdot 1$. On the other hand, in the ideal world $\Pr\left[(a_i, \mathrm{Out}_{P_2}) = (1,0)\right] = (1-\alpha)p_x(1-q_x^{(1)}(y))$. The probabilities in the two worlds are equal when

$$(1-\alpha)p_x(1-s_y) + \alpha f(x,y) = (1-\alpha)p_x(1-q_x^{(1)}(y)).$$

If $p_x = 0$, then $f(x,y) = 0$ and we are done. Otherwise, this equality holds iff

$$q_x^{(1)}(y) = s_y - \frac{\alpha f(x,y)}{(1-\alpha)p_x}. \tag{6}$$

As this is true for every y, we deduce, using Equation (1) and Equation (2), that

$$M^T \mathbf{x}_x^{(1)} = \mathbf{q}_x^{(1)} = M^T \left(1/\ell, \cdots, 1/\ell\right)^T - \frac{\alpha}{(1-\alpha)p_x}(\mathsf{row}_x)^T = M^T \left(\left(1/\ell, \cdots, 1/\ell\right)^T - \frac{\alpha}{(1-\alpha)p_x}\mathbf{e_x}\right), \tag{7}$$

where row_x is the row of M labeled by the input x, and $\mathbf{e_x}$ is the x-th unit vector. Before analyzing when there exists a probability vector $\mathbf{x}_x^{(1)}$ solving Equation (7), we remark that if the equalities hold for the first 3 cases of the values for $(a_i, \mathrm{Out}_{P_2})$, the equality of the probabilities must also hold for the case $(a_i, \mathrm{Out}_{P_2}) = (1,1)$. In the rest of the proof we show that there exist probability vectors $\mathbf{x}_x^{(1)}$ for every $x \in X$ solving Equation (7).

Claim 3.4. *Fix $x \in X$ and let α be a sufficiently small constant. If $(\mathbf{0}_k)^T$ is an affine combination of the rows of M, then there exists a probability vector $\mathbf{x}_x^{(1)}$ solving Equation (7).*

Proof. By the conditions of the claim, there exists a vector $u \in \mathbb{R}^\ell$ such that $M^T\mathbf{u} = \mathbf{0}_k$ and $\left(1, \cdots, 1\right)\mathbf{u} = 1$. Consider the vector $\mathbf{y} = \left(1/\ell, \cdots, 1/\ell\right)^T + \frac{\alpha}{(1-\alpha)p_x}(\mathbf{u} - \mathbf{e_x})$. The vector $\mathbf{y}$ is a solution to Equation (7) since $M^T\mathbf{u} = \mathbf{0}_k$. We need to show that it is a probability vector. First,

$$\left(1, \cdots, 1\right)\mathbf{y} = \left(1, \cdots, 1\right)\left(1/\ell, \cdots, 1/\ell\right)^T + \left(1, \cdots, 1\right)\frac{\alpha}{(1-\alpha)p_x}(\mathbf{u} - \mathbf{e_x}) = 1 + \frac{\alpha}{(1-\alpha)p_x}(1-1) = 1.$$

Second, $y_i \geq 1/\ell - \frac{\alpha}{(1-\alpha)p_x}(1 + |u_i|) \geq 1/\ell - 2k\alpha(1 + \max\{|u_i|\})$ (recall that $p_x \geq 1/k$ as $p_x > 0$ and p_x is a multiple of $1/k$). Thus, by taking $\alpha \leq 1/(2k\ell(1 + \max\{|u_i|\}))$, the vector $\mathbf{y}$ is non-negative, and, therefore, it is a probability vector solving Equation (7).[2] □

To conclude, we have showed that if $(\mathbf{0}_k)^T$ is an affine combination of the rows of M, then there are probability vectors solving Equation (7). Furthermore, these probability vectors can be efficiently found. Using this probability distributions in the simulator we constructed proves that protocol FAIRTWOPARTY$_0$ is fully secure for f. □

[2]As the size of the domain of f is considered as a constant, the vector $\mathbf{u}$ is a fixed vector and α is a constant.

We provide an alternative proof of Claim 3.4 in the appendix. Furthermore, in the appendix we prove the converse of Claim 3.4. This converse claim shows that the condition that $(\mathbf{0}_k)^T$ is an affine combination of the rows of M, which is sufficient for our protocol to securely compute f, is necessary for our simulation of protocol $\textsc{FairTwoParty}_0$.

Changing the Constant. In the above proof, we have fixed b_{i^*-1} (i.e., P_2's backup output at round i^*-1) to be 0. By changing this constant and setting it to 1 (that is, by executing Protocol $\textsc{FairTwoParty}_1$), we can securely compute additional functions.

Corollary 3.5. *Let M be the associated matrix of a Boolean function f. If $(\mathbf{1}_k)^T$ is an affine combination of the rows of M, then there is a constant $\alpha_0 > 0$ such that Protocol $\textsc{FairTwoParty}_1$ with $\alpha \leq \alpha_0$ is a fully-secure protocol for f.*

Proof. Let $\overline{M} = \big(1\big) - M$. Executing Protocol $\textsc{FairTwoParty}_1$ on f is equivalent to executing Protocol $\textsc{FairTwoParty}_0$ with the function $\bar{f}(x,y) = 1 - f(x,y)$ (whose associated matrix is $\overline{M}$), and flipping the output. Thus, Protocol $\textsc{FairTwoParty}_1$ is fully secure for f if $(\mathbf{0}_k)^T$ is an affine combination of the rows of $\overline{M}$. By the conditions of the corollary, there is an affine combination of the rows of M that equals $(\mathbf{1}_k)^T$ and the same affine combination applied to the rows of $\overline{M}$ equals $(\mathbf{0}_k)^T$, thus, Protocol $\textsc{FairTwoParty}_0$ is a fully secure protocol for $\bar{f}$. □

Corollary 3.6. *Assume that there is a secure protocol for OT. A Boolean two-party function f with an associated matrix M is computable with full security if:*

- *either $\mathbf{0}_\ell$ or $\mathbf{1}_\ell$ is an affine combination of the columns of M, or*
- *either $(\mathbf{0}_k)^T$ or $(\mathbf{1}_k)^T$ is an affine combination of the rows of M.*

Proof. By Theorem 3.3, a function can be computed with full security by protocol $\textsc{FairTwoParty}_0$ if $(\mathbf{0}_k)^T$ is an affine combination of the rows of M. Likewise, by Corollary 3.5, a function can be computed with protocol $\textsc{FairTwoParty}_1$ if $(\mathbf{1}_k)^T$ is an affine combination of the rows of M. By changing the roles of the parties in the protocol (in particular, P_2 gets the correct output before P_1), we obtain the other two possibilities in the corollary.

Assuming that there exists a secure protocol for OT, we can transform protocol $\textsc{FairTwoParty}_\sigma$ to a protocol without a dealer providing full security. □

3.1 Limits of the GHKL Protocol

We consider the function that was given in Example 1.2 and show that it cannot be computed using the GHKL protocol. In fact, we show a concrete attack on the protocol, and construct an adversary that succeeds to influent the output of the honest party. We then explain how the modified protocol is not susceptible to this attack.

Assume that P_2 chooses its input from $\{y_1, y_2, y_4\}$ uniformly at random (each with probability 1/3), and in case input y_1 was chosen it flips the output that was received. If the protocol computes f with full security, then P_2 receives a bit that equals 1 with probability 2/3, no matter what input distribution (malicious) P_1 may use.

Now, assume that the adversary $\mathcal{A}$ corrupts P_1, always uses input x_1, and follows the following strategy: If $a_1 = 1$ it aborts immediately (in which case, P_2 outputs b_0). Otherwise, it aborts at round 2 (in which case, P_2 outputs b_1).

We now compute the probability that P_2 outputs 1 when it runs the GHKL protocol with $\mathcal{A}$. We have the following cases:

1. If $i^* \neq 1$, then both b_0 and b_1 are random values chosen by the online-dealer, and thus both possible values of a_1 yield the same result. In this case, the output of P_2 is 1 with probability 2/3.

2. If $i^* = 1$, then the value a_1 is the correct output (i.e., $f(x, y)$ where $y \in \{y_1, y_2, y_4\}$). Moreover, the output of P_2 depends on a_1: If $a_1 = 1$, then it outputs b_0, which is random value chosen by the online-dealer. In case $a_1 = 0$, it outputs b_1, which is the correct output.

   Since $\mathcal{A}$ uses input x_1, the case of $a_1 = 1$ may occur only if the input of P_2 is y_4, and thus $b_0 = 1$ with probability 3/4. On the other hand, if $a_1 = 0$ then P_2's input is either y_1 or y_2, and it receives correct output b_1. It outputs 1 only in the case where its input was y_1.

We therefore conclude:

$$\begin{aligned}
\Pr[\text{Out}_2 = 1] &= \Pr[i^* \neq 1] \cdot \Pr[\text{Out}_2 = 1 \mid i^* \neq 1] + \Pr[i^* = 1] \cdot \Pr[\text{Out}_2 = 1 \mid i^* = 1] \\
&= (1-\alpha) \cdot \frac{2}{3} + \alpha \cdot \Big(\Pr[\text{Out}_2 = 1 \wedge a_1 = 0 \mid i^* = 1] + \Pr[\text{Out}_2 = 1 \wedge a_1 = 1 \mid i^* = 1] \Big) \\
&= (1-\alpha) \cdot \frac{2}{3} + \alpha \cdot \left(\frac{2}{3} \cdot \frac{1}{2} + \frac{1}{3} \cdot \frac{3}{4} \right) = \frac{2}{3} - \frac{1}{12}\alpha \neq \frac{2}{3} \ .
\end{aligned}$$

Protocol FairTwoParty$_1$**.** Since $\mathbf{1}_k^T$ is an affine combination of the rows of M, we use the protocol FairTwoParty$_1$, that is, use the backup output at round $i^* - 1$ to be 1. Assume that the adversary corrupts P_1 and instructs him to conduct the same attack in some round $i \geq 2$ (as $i^* \geq 2$ such attack in round 1 is meaningless in our protocol) . Now, if $i = i^*$, the value of b_{i^*-1} is always 1, and the value of b_{i^*} is correct. If $i < i^*$, then b_i is some random value chosen by the dealer. In the former case ($i^* = i$), since P_2 flips its output in case its input was y_1, its output is 1 with probability 2/3. In the latter case, P_2 outputs 1 with probability 2/3. We get that the final output of P_2 is 1 with probability 2/3, and conclude that the protocol is not susceptible to the attack described above.

The intuition why the GHKL protocol is susceptible to this attack whereas our protocol is immune to it, is the following. In the GHKL protocol, for each input y the distributions of b_{i^*-1} is different. At round i^*, the value a_{i^*} is "correct", and leaks information about the distribution of b_{i^*-1}. This gives the adversary the ability to bias the output of P_2 once it guesses correctly the round i^*. In contrary, in our protocol, we detach the correlation between a_{i^*} and b_{i^*-1}; although a_{i^*} leaks information about the input of P_2, all distributions of b_{i^*-1} are exactly the same for all possible inputs y, and this attack is bypassed.

3.2 Characterization of Fairness via Non Semi-Balanced Functions

Recall the definition of *semi-balanced* functions given in [12]:

Definition 3.7. *A function $f : X \times Y \to \{0, 1\}$ with an associated $\ell \times k$ matrix M is right semi-balanced if $\exists \mathbf{p} \in \mathbb{R}^k$ such that $M\mathbf{p} = \mathbf{1}_\ell$ and $\sum_{i=1}^{k} p_i \neq 1$. Similarly, f is left semi-balanced if $\exists \mathbf{q} \in \mathbb{R}^\ell$ such that $M^T\mathbf{q} = \mathbf{1}_k$ and $\sum_{i=1}^{\ell} q_i \neq 1$. A function f is semi-balanced if it is right semi-balanced and left semi-blanced.*

Makriyannis [12] proved that semi-unbalanced functions are inherently unfair, by showing that a fully-secure protocol for a semi-balanced function implies fair-sampling. We claim that if f is not semi-balanced then f is computable with full security by Protocol $\textsc{FairTwoParty}_\sigma$.

Lemma 3.8. *If f is* not *right semi-balanced, then either $(\mathbf{0}_k)^T$ is an affine combination of the rows of M or $\mathbf{1}_\ell$ is an affine combination of the columns of M.*

If f is not *left semi-balanced, then either $(\mathbf{1}_k)^T$ is an affine combination of the rows of M or $\mathbf{0}_\ell$ is an affine combination of the columns of M.*

Proof. We show only the case where f is not right semi-balanced, the case of left semi-balanced is proven analogously. If f is not right semi-balanced, then one of the following is implied: Either $\mathbf{1}_\ell \notin \text{im}(M)$, which, by Proposition 2.1, implies that $(\mathbf{0}_k)^T$ is an affine combination of the rows of M. Alternatively, it can be that $\mathbf{1}_\ell \in \text{im}(M)$. In this case the vector $\mathbf{p}$ for which $M \cdot \mathbf{p} = \mathbf{1}_\ell$ satisfies $\sum_{i=1}^{\ell} p_i = 1$ (since f is not right semi-balanced). This in particular implies that $\mathbf{1}_\ell$ is an affine combination of the columns of M. □

Theorem 3.9. *Assume an that there is a secure protocol for OT. Let f be a Boolean two-party function f with an associated matrix M. The function f can be computed with full-security if and only if f is not semi-balanced if and only if at least one of the following conditions holds*

I. $(\mathbf{0}_k)^T$ is an affine combination of the rows of M or $\mathbf{1}_\ell$ is an affine combination of the columns of M,

II. $(\mathbf{1}_k)^T$ is an affine combination of the rows of M or $\mathbf{0}_\ell$ is an affine combination of the columns of M.

Proof. If f is semi-balanced, then by [12] it cannot be computed with complete fairness, hence, it cannot be computed with full security.

If f is not semi-balanced, then, by Lemma 3.8, at least one of the conditions (I) or (II) holds. By Corollary 3.6, conditions (I) or (II) imply (assuming that there is a secure protocol for OT) that f can be computed with full security. □

3.3 A Geometric Interpretation of the Characterization

We review the geometric representation of our characterization, which may shed some light on on our understanding of full security. We start by linking between semi-balanced functions and linear hyperplanes.

A *linear hyperplane* in $\mathbb{R}^m$ is an $(m-1)$-dimensional affine subspace of $\mathbb{R}^m$, and is defined as all the points $X = (x_1, \dots, x_m) \in \mathbb{R}^m$ that are a solution of some linear equation $a_1x_1 + \ldots + a_mx_m = b$, for some constants $\mathbf{a} = (a_1, \dots, a_m) \in \mathbb{R}^m$ and $b \in \mathbb{R}$. We denote this hyperplane by $\mathcal{H}(\mathbf{a}, b) \triangleq \{X \in \mathbb{R}^m \mid \langle X, \mathbf{a}\rangle = b\}$. We show alternative representations for the semi-balanced property:

Claim 3.10. *Let f be a Boolean function, let $X_1, \dots, X_\ell \in \mathbb{R}^k$ denote the rows of the matrix M, and let $Y_1, \dots, Y_k \in \mathbb{R}^\ell$ denote the columns of M. The following are equivalent:*

1. *The function is semi-balanced.*

2. $\mathbf{0}_k, \mathbf{1}_k \notin \textsf{affine-hull}\{X_1, \dots, X_\ell\}$ *and* $\mathbf{0}_\ell, \mathbf{1}_\ell \notin \textsf{affine-hull}\{Y_1, \dots, Y_k\}$.

3. *There exists an hyperplane $\mathcal{H}(\mathbf{q},1)$ that contains all the rows $X_1,\ldots,X_\ell$, and there exists yet another hyperplane $\mathcal{H}(\mathbf{p},1)$ that contains all the columns $Y_1,\ldots,Y_k$. In addition, $\mathbf{1}_k,\mathbf{0}_k \notin \mathcal{H}(\mathbf{q},1)$ and $\mathbf{1}_\ell,\mathbf{0}_\ell \notin \mathcal{H}(\mathbf{p},1)$.*

Proof. <u>**3 ⇒ 2**:</u> Recall that the affine hull of set S of vectors is the smallest affine set containing S, or, equivalently, the intersection of all affine sets containing S. Since hyperplanes are affine sets, and since all the rows lie on the same hyperplane $\mathcal{H}(\mathbf{p},1)$, then $\mathsf{affine\text{-}hull}\{X_1,\ldots,X_\ell\} \subseteq \mathcal{H}(\mathbf{p},1)$. Since $\mathbf{1}_k,\mathbf{0}_k \notin \mathcal{H}(\mathbf{p},1)$, it holds that $\mathbf{1}_k,\mathbf{0}_k \notin \mathsf{affine\text{-}hull}\{X_1,\ldots,X_\ell\}$. The case of $\mathbf{1}_\ell,\mathbf{0}_\ell \notin \mathsf{affine\text{-}hull}\{Y_1,\ldots,Y_k\}$ is shown analougesly.

<u>**2 ⇒ 1**:</u> This case is implied by Lemma 3.8.

<u>**1 ⇒ 3**:</u> Clearly, if there exists a vector $\mathbf{p} \in \mathbb{R}^k$ such that $M \cdot \mathbf{p} = \mathbf{1}_\ell$ and $\sum_{i=1}^k p_i \neq 1$, then for each row it holds that $\langle X_i,\mathbf{p}\rangle = 1$, and so all the rows lie on the hyperplane $\mathcal{H}(\mathbf{p},1)$. In addition, $\langle \mathbf{0},\mathbf{p}\rangle = 0 \neq 1$ and $\langle \mathbf{1},\mathbf{p}\rangle = \sum_{i=1}^k p_i \neq 1$ and so both $\mathbf{1}_\ell,\mathbf{0}_\ell \notin \mathcal{H}(\mathbf{p},1)$. The case of left semi-balanced is equivalent. □

The following theorem divides the Boolean functions to three categories. Each category has some different properties which we will discuss

Theorem 3.11. *Let f be a Boolean function. Then:*

1. **The function is balanced:**
   There exists an hyperplane $\mathcal{H}(\mathbf{q},1)$ that contains all the rows $X_1,\ldots,X_\ell$, and there exists yet another hyperplane $\mathcal{H}(\mathbf{p},1)$ that contains all the columns $Y_1,\ldots,Y_k$. Then:
   (a) *If $\mathbf{0}_\ell,\mathbf{1}_\ell \notin \mathcal{H}(\mathbf{p},1)$ and $\mathbf{0}_k,\mathbf{1}_k \notin \mathcal{H}(\mathbf{q},1)$, then the function is semi-balanced and cannot be computed fairly.*
   (b) *If either $H(\mathbf{p},1),\mathcal{H}(\mathbf{q},1)$ contains one of the vectors $\mathbf{1}_k,\mathbf{0}_k,\mathbf{1}_\ell,\mathbf{0}_\ell$, then f can be computed with full-security.*
2. **The function is unbalanced (full-dimensional):**
   If the rows do not lie on a single hyperplane, or the columns do no lie on a single hyperplane, then the function is unbalanced and can be computed with full-security.

Case 2 was proven as possible in [2,12]. Case 1a was proven as impossible to be computed with the GHKL protocol in [2] (with respected to a specific simulation strategy), and later was proven to be inherently unfair, and was reduced to fair sampling [12]. Finally, case 1b was left unresolved in [2,12], and is finally proven to be possible here.

Intuition. Intuitively, fairness is possible to achieve in functions which are more "unbalanced" and is impossible in functions that are "balanced". In our context, the term "balanced" relates to the question whether each party can influence and bias the output of the other party in a single (and fair) invocation of the function. In functions that are "balanced", both parties cannot bias the output of the other, and thus these functions can be used as a reliable source for sampling correlated coins. In contrast , functions that are unbalanced, one party has significantly more ability to bias the output than the other, and thus they are not reducible to sampling.

The fact that one party has the ability to influence the output of the other party with a single ideal invocation of the function, is what enables constructing a fair protocol for the two parties. In the possibility, the simulator uses the ability that the function enables it, to bias the output of the honest party in the ideal execution (where, an ideal execution is close in its nature to a single fair invocation of the function). Thus, the exact same property that enables or disables reducible to sampling, is the property that enables or disables simulation.

The intuition above is demonstrated in the above classes. Consider the class of functions 1a. In this class, every function is very balanced; From the fact that all the rows lie on a single hyperplane, any affine combination of them, and in particular, any convex combination of them is also in the hyperplane. This implies that in any single invocation of the function, no matter out input distribution P_1 uses (what convex combination it uses), the possible outputs of P_2 satisfy the exact same constraint. We have precisely the same guarantee for the case of malicious P_2 and honest P_1. These are the key properties that enable reducing these functions to sampling, since each party has some guarantee regarding its output no matter what input distribution the other party uses. Two other properties that show that these functions are balanced are the following. First, the dimensions of the affine hulls of the rows and the columns are exactly the same, and both equal $\min\{k, \ell\} - 1$. Second, as pointed out in [2,12], almost all functions in this class satisfy $|X| = |Y|$.

The other extreme case is the class 2. In this class of functions, one party has significant more power over the other. In particular, assume that the rows do not lie on a single hyperplane. As a result, any convex combination of the rows may result in some different hyperplane, and thus no constraint is guaranteed on the outputs of P_2, and in particular a malicious P_1 can influent them. We use this ability to influence the output in the proof of possibility, where the simulator chooses the input distribution $\mathbf{x}_x^{(b)}$ to influent the output of the honest party, so that the simulation will be identical to the real. In addition, for this class of function it holds that the dimension of the affine hulls of the rows and columns are always distinct, where one of the dimensions is full ($\min\{\ell, k\}$), and the other is strictly lower. Moreover, all functions in this class satisfy $|X| \neq |Y|$.

The third class of functions 1b is where the things become less clearer. This class contains functions where the affine dimensions of the rows and columns are exactly the same. Moreover, this class contains also functions that are even symmetric and satisfy $M_f^T = M_f$ (see, for instance, Example 1.2). Still, fairness is possible. Overall, since all the rows and columns lie on hyperplanes, all the functions in this class can be reduced to sampling; however, the sampling is a trivial one (where the resulting coins of the parties are uncorrelated). In addition, although the influent that a party may have on the output of the other is significantly less than the case of class 2, it turns out that simulation is still possible, and the simulator can "smudge" the exact advantage that the real world adversary has in the real execution.

Randomized Functions. Let $f : X \times Y \to \Delta(\{0, 1\})$ be a randomized finite Boolean function and define associated $\ell \times k$ matrix M such that $M_{i,j} = \Pr[f(x_i, y_j) = 1]$. By modifying Protocol $\textsc{FairTwoParty}_\sigma$ such that the dealer now computes randomized outputs, we note that the analysis above still holds. In particular, the following theorem is true.

Theorem 3.12. *Assume that there is a secure protocol for OT. A randomized Boolean two-party function f is computable with complete security if and only if it is not semi-balanced.*

4 On the Characterization of Fairness for Asymmetric Functions

In this section, we study asymmetric Boolean functions. Namely, we consider functions $f = (f_1, f_2)$, where $f_i : X \times Y \to \{0,1\}$ for each $i \in \{1,2\}$. As two-party functionalities, P_1 and P_2's input domains correspond to sets X and Y respectively, like the symmetric case, however, their outputs now are computed according to f_1 and f_2 respectively. In other words, the parties are computing different functions on the same inputs.

Our goal is to extend the characterization of full security to this class of functions. In particular, we show how the feasibility result (Theorem 3.3) and the impossibility result (reduction from non-trivial sampling) translate to the asymmetric case. Unfortunately, while these two results provide a tight characterization for symmetric functionalities, the same cannot be said about asymmetric ones. As a first step toward the characterization the latter functions, we consider a particular function that lies in the gap and describe a new protocol that computes this function with full security. While this protocol may be considered as another variant of the GHKL-protocol, we believe that it departs considerably from the protocols we have considered thus far. Consequently, it seems that the characterization for asymmetric functionalities is more involved than the symmetric case.

4.1 Sufficient Condition for Fairness of Two-Party Asymmetric Functionalities

We analyze when Protocol FairTwoParty$_\sigma$, described in Figure 3, provides full security for Boolean asymmetric functionalities $f = (f_1, f_2)$. We modify Protocol FairTwoParty$_\sigma$ such that the backup values of P_i are computed with f_i; we call the resulting protocol FairTwoPartyAsymm$_\sigma$.

Theorem 4.1. *Let f_1, f_2 be functions with associated matrices M_1 and M_2 respectively. If $\mathbf{0}_k$ is an affine combination of the rows of M_2, and all the rows of $M_1 * M_2$ are linear combinations of the rows of M_2, then there is a constant $\alpha_0 > 0$ such that Protocol* FairTwoPartyAsymm$_0$ *with $\alpha \leq \alpha_0$ is a fully-secure protocol for (f_1, f_2).*

In Theorem 4.1, we require that all the rows of the matrix $M_1 * M_2$ are in the row-span of M_2, and that the vector $\mathbf{0}_k$ is an affine combination of the rows of M_2. Note that when the function is symmetric, and thus, $M_1 = M_2$, the first requirement always holds and the only requirement is the second, exactly as in Theorem 3.3.

Proof of Theorem 4.1. We only discuss the required changes in the proof of Protocol FairTwoPartyAsymm$_0$ compared to the proof of Protocol FairTwoParty$_0$. We use the same simulator for P_1. The difference are in its proof. As in the proof of Theorem 3.3, we only need to compare the distribution of $(a_i, \mathrm{Out}_{P_2})$ in both worlds given that the adversary aborts in round $i \leq i^*$.

First, we introduce the following notation

$$(1/\ell, \ldots, 1/\ell)M_2 = (s_1, \ldots, s_k), \qquad M_1 \begin{pmatrix} 1/k \\ \vdots \\ 1/k \end{pmatrix} = \begin{pmatrix} p_1 \\ \vdots \\ p_\ell \end{pmatrix}. \tag{8}$$

For example, s_y is the probability that $f_2(\widetilde{x}, y) = 1$, when $\widetilde{x}$ is uniformly distributed. Furthermore, for every $x \in X$, $y \in Y$, and $a \in \{0,1\}$, define $q_x^{(a)}(y)$ as the probability that the output of P_2 in the simulation is 1 (given that the adversary has aborted in round $i \leq i^*$) when the input of P_1 is

x, the input of P_2 is y, and $a_i = a$, that is $q_x^{(a)}(y) \triangleq \Pr[f(x_0, y) = 1]$, where x_0 is chosen according to the distribution $\mathbf{x}_x^{(a)}$. Finally, define the column vector $\mathbf{q}_x^{(a)} \triangleq (q_x^{(a)}(y))_{y \in Y}$. Using this notation,

$$M_2^T \mathbf{x}_x^{(a)} = \mathbf{q}_x^{(a)}, \tag{9}$$

where we represent the distribution $\mathbf{x}_x^{(a)} = (\mathbf{x}_x^{(a)}(x_0))_{x_0 \in X}$ by a column vector.

We next analyze the four options for the values of $(a_i, \mathrm{Out}_{P_2})$.

First case: $(a_i, \mathrm{Out}_{P_2}) = (0,0)$**.** In the real world $\Pr[(a_i, \mathrm{Out}_{P_2}) = (0,0)] = (1-\alpha)(1-p_x)(1-s_y) + \alpha(1 - f_1(x,y))$. On the other hand, in the ideal world $\Pr[(a_i, \mathrm{Out}_{P_2}) = (0,0)] = (1-\alpha)(1-p_x)(1-q_x^{(0)}(y)) + \alpha(1-f_1(x,y))(1-f_2(x,y))$. To get full security we need that these two probabilities in the two worlds are the same, that is,

$$(1-\alpha)(1-p_x)(1-s_y)+\alpha(1-f_1(x,y)) = (1-\alpha)(1-p_x)(1-q_x^{(0)}(y))+\alpha(1-f_1(x,y))(1-f_2(x,y)). \tag{10}$$

If $p_x = 1$, then $f_1(x,y) = 1$ and (10) holds. Otherwise, (10) is equivalent to

$$q_x^{(0)}(y) = s_y - \frac{\alpha \cdot (1 - f_1(x,y)) f_2(x,y)}{(1-\alpha)(1-p_x)}. \tag{11}$$

As this is true for every y, we deduce, using Equations (2) and (1), that

$$M_2^T \mathbf{x}_x^{(0)} = \mathbf{q}_x^{(0)} = M_2^T \left(1/\ell, \cdots, 1/\ell\right)^T - \frac{\alpha}{(1-\alpha)(1-p_x)} (\overline{M_1} * M_2)^T \mathbf{e_x}. \tag{12}$$

Claim 4.2. *Fix $x \in X$ and let α be a sufficiently small constant. If $(\mathbf{0}_k)^T$ is an affine combination of the rows of M_2 and all the rows of $M_1 * M_2$ are linear combinations of the rows of M_2, then there exists a probability vector $\mathbf{x}_x^{(0)}$ solving Equation (12).*

Proof. Let $\mathbb{1}$ denote the $\ell \times k$ all-one matrix, and let $\lambda_\alpha = \frac{\alpha}{(1-\alpha)(1-p_x)}$. We get that $(\overline{M_1} * M_2)^T \mathbf{e_x} = ((\mathbb{1} - M_1) * M_2)^T \mathbf{e_x} = (M_2 - M_1 * M_2)^T \mathbf{e_x} = M_2^T \mathbf{e_x} - (M_1 * M_2)^T \mathbf{e_x}$. Literally, this is the subtraction of some row x in matrix M_2, and the row x in the matrix $M_1 * M_2$. However, from the conditions in the statement, this is just a vector in the linear-span of the rows of M_2, and can be represented as $M^T \cdot \mathbf{v}$ for some constants $\mathbf{v} \in \mathbb{R}^\ell$. We therefore are looking for a probability vector $\mathbf{x}_x^{(0)}$ solving:

$$M_2^T \mathbf{x}_x^{(0)} = M_2^T \left(1/\ell, \cdots, 1/\ell\right)^T - \lambda_\alpha \cdot M_2^T \mathbf{v}. \tag{13}$$

Let $\beta = \langle \mathbf{1}, \mathbf{v} \rangle$, and recall that $\mathbf{0}_k$ is in the affine hull of the rows of M_2, and thus there exists a vector $\mathbf{u} \in \mathbb{R}^\ell$ such that $M^T \cdot \mathbf{u} = \mathbf{0}_k$ and $\langle \mathbf{1}, \mathbf{u} \rangle = 1$. Consider the vector $\mathbf{y} = (1/\ell, \ldots, 1/\ell)^T - \lambda_\alpha \mathbf{v} + \lambda_\alpha \beta \mathbf{u}$. This vector is a solution for Eq. (13) since $M_2^T \cdot \mathbf{u} = \mathbf{0}_k$. Moreover, it sums-up to 1 since:

$$\langle \mathbf{1}, \mathbf{y} \rangle = \langle \mathbf{1}, (1/\ell, \ldots, 1/\ell)^T \rangle - \lambda_\alpha \langle \mathbf{1}, \mathbf{v} \rangle + \lambda_\alpha \beta \langle \mathbf{1}, \mathbf{u} \rangle = 1 - \lambda_\alpha \beta + \lambda_\alpha \beta \cdot 1 = 1$$

Finally, for appropriate choice of α, all the coordinates of $\mathbf{y}$ are non-negative, and thus $\mathbf{y}$ is a probability vector solving Eq. (13). $\square$

Second case: $(a_i, \mathrm{Out}_{P_2}) = (0,1)$**.** In the real world $\Pr[(a_i, \mathrm{Out}_{P_2}) = (0,1)] = (1-\alpha)(1-p_x)s_y$ (in the real world $\mathrm{Out}_{P_2} = b_{i^*-1} = 0$ when $i = i^*$). On the other hand, in the ideal world $\Pr[(a_i, \mathrm{Out}_{P_2}) = (0,1)] = (1-\alpha)(1-p_x)q_x^{(0)}(y) + \alpha \cdot (1 - f_1(x,y)) f_2(x,y)$ (in the ideal world $a_{i^*} = f_1(x,y)$ and $\mathrm{Out}_{P_2} = f_2(x,y)$). The probabilities in the two worlds are equal if (11) holds.

Third case: $(a_i, \mathrm{Out}_{P_2}) = (1, 0)$. In the real world $\Pr\left[(a_i, \mathrm{Out}_{P_2}) = (1,0)\right] = (1-\alpha)p_x(1-s_y) + \alpha \cdot f_1(x,y) \cdot 1$. On the other hand, in the ideal world $\Pr\left[(a_i, \mathrm{Out}_{P_2}) = (1,0)\right] = (1-\alpha)p_x(1-q_x^{(1)}(y)) + \alpha \cdot f_1(x,y)(1-f_2(x,y))$. The probabilities in the two worlds are equal when

$$(1-\alpha)p_x(1-s_y) + \alpha f_1(x,y) = (1-\alpha)p_x(1-q_x^{(1)}(y)) + \alpha f_1(x,y)(1-f_2(x,y)).$$

If $p_x = 0$, then $f_1(x,y) = 0$ and we are done. Otherwise, this equality holds iff

$$q_x^{(1)}(y) = s_y - \frac{\alpha f_1(x,y) f_2(x,y)}{(1-\alpha)p_x}. \tag{14}$$

As this is true for every y, we deuce, using Equations (9) and (8), that

$$M_2^T \mathbf{x}_x^{(1)} = \mathbf{q}_x^{(1)} = M^T \left(1/\ell, \cdots, 1/\ell\right)^T - \frac{\alpha}{(1-\alpha)p_x}(M_1 * M_2)^T \mathbf{e}_\mathbf{x}, \tag{15}$$

Analogically to case $(0,0)$, there exists a probability vector $\mathbf{x}_x^{(1)}$ solving Equation (15) if each row of $M_1 * M_2$ is in the row span of M_2, and $\mathbf{0}_k$ is in the affine hull of the rows of M_2.

If the equalities hold for the first 3 cases of the values for $(a_i, \mathrm{Out}_{P_2})$, the equality of the probabilities must also hold for the case $(a_i, \mathrm{Out}_{P_2}) = (1,1)$.

To conclude, the conditions of the theorem imply that there are probability vectors solving Equations (12) and (15) for every x. Using these probability distributions in the simulator we constructed, we conclude that protocol FairTwoPartyAsymm$_0$ is fully secure for $f = (f_1, f_2)$.

□

Changing the Matrices. In the symmetric setting, we have showed that by flipping the matrix M associated with the function (i.e., by taking the matrix $\overline{M} = \left(1\right) - M$) we can construct protocols with full security for a richer class of functions. In the asymmetric setting we can go even further: P_1 and P_2 can flip some of the rows of M_1 and obtain a matrix $\hat{M}_1$ and flip some of the columns of M_2 and obtain a matrix $\hat{M}_2$. The parties now execute FairTwoPartyAsymm$_0$ on the flipped matrices and the parties obtain outputs a and b respectively. If the input of P_1 corresponds to a row that was flipped, then P_1 outputs $1-a$, otherwise, it outputs a. Similarly, if the input of P_2 corresponds to a column that was flipped, then P_2 outputs $1-b$, otherwise, it outputs b. Call the resulting protocol FairTwoPartyAsymm'. Thus, we obtain the following corollary.

Corollary 4.3. *Let M_1, M_2 be the associated matrices of f_1 and f_2 respectively. Assume that $\hat{M}_1$ is computed from M_1 by flipping some of its rows and $\hat{M}_2$ is computed from M_2 by flipping some of its columns. If $\mathbf{0}_k$ is an affine-combination of the rows of $\hat{M}_2$, and all the rows of $\hat{M}_1 * \hat{M}_2$ are in the linear span of the rows of $\hat{M}_2$, then then there is a constant $\alpha_0 > 0$ such that Protocol* FairTwoPartyAsymm*' with $\alpha \le \alpha_0$ is a fully-secure protocol for $f = (f_1, f_2)$.*

4.2 Necessary Condition

In this section, we provide a necessary condition for fully secure computation of asymmetric functionalities. It consists of a natural generalization of the semi-balanced criterion for symmetric functionalities. Namely, we show that certain functions imply (non-private) non-trivial sampling and are thus unfair. Informally, the next theorem states that if both parties have some distribution over their inputs that "cancels out" the other party's choice of input, then, assuming the resulting bits are statistically dependent, the function cannot be computed with complete fairness.

Theorem 4.4. *Let f_1, f_2 be functions with associated matrices M_1 and M_2 respectively. If there exist $\mathbf{p} \in \mathbb{R}^\ell$, $\mathbf{q} \in \mathbb{R}^k$ such that $\mathbf{p}^T M_1 = \delta_1 \cdot \mathbf{1}_k^T$, $M_2\mathbf{q} = \delta_2 \cdot \mathbf{1}_\ell$ and $\mathbf{p}^T(M_1 * M_2)\mathbf{q} \neq \delta_1\delta_2$, then the functionality $f(x,y) = (f_1(x,y), f_2(x,y))$ implies (non-private) non-trivial sampling.*

Proof. Suppose there exist $\mathbf{p} \in \mathbb{R}^\ell$, $\mathbf{q} \in \mathbb{R}^k$ such that $\mathbf{p}^T M_1 = \delta_1 \cdot \mathbf{1}_k^T$, $M_2\mathbf{q} = \delta_2 \cdot \mathbf{1}_\ell$ and $\mathbf{p}^T(M_1 * M_2)\mathbf{q} \neq \delta_1\delta_2$. Further assume that $\sum_i |p_i| = \sum_i |q_j| = 1$, and define $\delta_{1,2} = \mathbf{p}^T(M_1 * M_2)\mathbf{q}$. Consider the following protocol Π in the hybrid model with ideal access to f (with full security/complete fairness).

- **Inputs:** Empty for both parties.
- **Invoke trusted party:** Parties choose x_i, y_j according to probability vectors $|\mathbf{p}|$ and $|\mathbf{q}|$ respectively, and invoke the trusted party, write a and b for the bits received by the first and second party.
- **Outputs:** P_1 outputs a if $p_i \geq 0$ and $1-a$ otherwise. P_2 outputs b if $q_j \geq 0$ and $1-b$ otherwise.

Claim 4.5. *Write* Out_1, Out_2 *for the outputs of P_1 and P_2 in the above protocol. In an honest execution of Π, the parties' outputs satisfy* $\Pr[\mathrm{Out}_1 = 1] = \delta_1 + p^-$, $\Pr[\mathrm{Out}_2 = 1] = \delta_2 + q^-$ *and* $\Pr[\mathrm{Out}_1 = 1 \wedge \mathrm{Out}_2 = 1] = \delta_{1,2} + p^-\delta_2 + q^-\delta_1 + p^-q^-$, *where $p^- = \sum_{p_i<0} |p_i|$ and $q^- = \sum_{q_j<0} |q_j|$.*

Proof. We begin the proof by introducing some notation. Let $\mathsf{row}_{1,i}$, $\mathsf{row}_{2,i}$ denote the i-th row of M_1 and M_2 respectively, and let $\mathsf{col}_{1,j}$, $\mathsf{col}_{2,j}$ denote the j-th column of M_1 and M_2 respectively. Construct matrices $\hat{M}_1$ and $\hat{M}_2$ such that

- the i-th row of $\hat{M}_1$ is equal to $\mathsf{row}_{1,i}$ if $p_i \geq 0$ and $\mathbf{1}_k^T - \mathsf{row}_{1,i}$ otherwise,
- the j-th column of $\hat{M}_2$ is equal to $\mathsf{col}_{2,j}$ if $q_j \geq 0$ and $\mathbf{1}_\ell - \mathsf{col}_{2,j}$ otherwise.

Let $\hat{\mathsf{row}}_{1,i}$, $\hat{\mathsf{col}}_{1,j}$ and $\hat{\mathsf{row}}_{2,i}$, $\hat{\mathsf{col}}_{2,j}$ denote the rows and columns of $\hat{M}_1$ and $\hat{M}_2$ respectively. The proof consists of a straightforward computation of each probability.

$$\begin{aligned}
\Pr[\mathrm{Out}_1 = 1 \mid y = y_j] &= |\mathbf{p}^T|\hat{M}_1\mathbf{e}_{y_j} \\
&= \left(\sum_{p_i<0} |p_i|\hat{\mathsf{row}}_{1,i} + \sum_{p_i\geq 0} p_i\hat{\mathsf{row}}_{1,i}\right)\mathbf{e}_{y_j} \\
&= \left(\sum_{p_i<0} |p_i|(\mathbf{1}_k^T - \mathsf{row}_{1,i}) + \sum_{p_i\geq 0} p_i\mathsf{row}_{1,i}\right)\mathbf{e}_{y_j}. \\
&= \left(\sum_{p_i<0} |p_i|\mathbf{1}_k^T + \mathbf{p}^T M_1\right)\mathbf{e}_{y_j} = \delta_1 + p^-. \qquad (16)
\end{aligned}$$

The output of P_2 is obtained in a similar fashion. Next,

$$\begin{aligned}
\Pr\left[(\mathrm{Out}_1, \mathrm{Out}_2) = (1,1)\right] &= |\mathbf{p}^T|(\hat{M}_1 * \hat{M}_2)|\mathbf{q}| \\
&= |\mathbf{p}^T| \left(\sum_j (\hat{\mathsf{col}}_{1,j} * \hat{\mathsf{col}}_{2,j})|q_j| \right) \\
&= |\mathbf{p}^T| \left(\sum_{q_j \geq 0} (\hat{\mathsf{col}}_{1,j} * \mathsf{col}_{2,j})q_j + \sum_{q_j<0} \left(\hat{\mathsf{col}}_{1,j} * (\mathbf{1}_\ell - \mathsf{col}_{2,j})\right)|q_j| \right) \\
&= |\mathbf{p}^T| \left((\hat{M}_1 * M_2)\mathbf{q} + \sum_{q_j<0} \hat{\mathsf{col}}_{1,j}|q_j| \right). \qquad (17)
\end{aligned}$$

Now, since by (16), $|\mathbf{p}^T|\hat{\mathsf{col}}_{1,j} = |\mathbf{p}^T|\hat{M}_1\mathbf{e}_{y_j} = \delta_1 + p^-$, we deduce that (17) is equal to

$$\begin{aligned}
&\left(\sum_{p_i \geq 0} p_i(\mathsf{row}_{1,i} * \mathsf{row}_{2,i}) + \sum_{p_i<0} |p_i|\left((\mathbf{1} - \mathsf{row}_{1,i}) * \mathsf{row}_{2,i}\right) \right) \mathbf{q} + (\delta_1 + p^-)q^- \\
&= \mathbf{p}^T(M_1 * M_2)\mathbf{q} + \sum_{p_i<0} |p_i|(\mathsf{row}_{2,i}\mathbf{q}) + (\delta_1 + p^-)q^- \\
&= \delta_{1,2} + p^-\delta_2 + q^-\delta_1 + p^-q^-.
\end{aligned}$$

□

It remains to show that protocol Π is a secure realization of (non-private) non-trivial sampling. First, we note that the parties' outputs above are statistically dependent. This follows from the fact that two bits are independent if and only if $\Pr[\mathrm{Out}_1 = 1]\cdot\Pr[\mathrm{Out}_2 = 1] = \Pr[\mathrm{Out}_1 = 1 \wedge \mathrm{Out}_2 = 1]$. Since, by assumption, $\delta_{1,2} \neq \delta_1\delta_2$, we deduce that in an honest execution the parties' outputs are statistically dependent. To conclude, note that no matter how the adversary (say controlling P_2) chooses his input – or doesn't send one at all, by (16), the probability that the output of P_1 is 1 is equal to $\delta_1 + p^-$. Hence, by [1], we get a contradiction. □

4.3 Special Round Protocol with a Twist

Define an asymmetric functionality $f_{\mathsf{sp}}(x, y) = (f_1(x, y), f_2(x, y))$, where f_1, f_2 are given by the following matrices

$$M_1 = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \quad M_2 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$

In this section we show that the above function can be computed with full security. First, note that function f_{sp} does not satisfy the hypothesis of Theorem 4.4. On the other hand, both the GHKL protocol as well as FairTwoPartyAsymm$_\sigma$ are susceptible to fail-stop attacks for this particular function, as explained below.

4.3.1 On the limits of the known protocols for the function f_{sp}

Suppose that parties execute protocol $\textsc{FairTwoPartyAsymm}_0$ for computing f_{sp}. In addition, suppose that P_2 chose his input uniformly at random from $\{y_1, y_2\}$. If the protocol computes f with full security, then this particular choice of inputs should result in P_2 obtaining an unbiased random bit as an output, regardless of the actions of P_1. We claim that a corrupt P_1 can bias P_2's output toward zero and thus protocol $\textsc{FairTwoPartyAsymm}_0$ does not compute f with full security. Consider an adversary $\mathcal{A}$ that quits immediately upon receiving a_2. Let's compute the probability that P_2's output is equal to 1 under the action of $\mathcal{A}$, and assuming that $y \in_U \{y_1, y_2\}$:

$$\begin{aligned}\Pr[\mathrm{out}_2 = 1] &= \Pr[b_1 = 1] \\ &= \Pr[b_1 = 1 \wedge i^* = 2] + \Pr[b_1 = 1 \wedge i^* \neq 2] \\ &= 0 + \frac{1}{2} \cdot (1 - \alpha).\end{aligned}$$

In summary, knowing that $b_{i^*-1} = 0$ and that $\mathcal{A}$ can guess i^* with non-negligible probability ($\mathcal{A}$ is betting that $i^* = 2$) we deduce that the above attack will result in P_2 outputting a bit that does not satisfy the prescribed probability distribution. If we revert to the original GHKL protocol, i.e., $i^* \geq 1$, and $b_{i^*-1} = f_2(\widetilde{x}^{(i^*-1)}, y)$ where $\widetilde{x}^{(i^*-1)}$ is chosen uniformly at random, then a very similar attack will produce the same result:

- The adversary instructs P_1 to use x_3 and execute the protocol.
- At the first round, if $a_1 = 1$ quit. Otherwise, quit upon receiving a_2.

Again, let's compute the probability that P_2's output is equal to 1 under the action of $\mathcal{A}$ assuming that $y \in_U \{y_1, y_2\}$:

$$\begin{aligned}\Pr[\mathrm{out}_2 = 1] &= \Pr[\mathrm{out}_2 = 1 \wedge i^* \neq 1] + \Pr[\mathrm{out}_2 = 1 \wedge i^* = 1] \\ &= (1 - \alpha)\frac{1}{2} + \alpha \cdot \Big(\Pr[a_1 = 1 \wedge b_0 = 1 \mid i^* = 1] + \Pr[a_1 = 0 \wedge b_1 = 1 \mid i^* = 1]\Big) \\ &= (1 - \alpha)\frac{1}{2} + \alpha \cdot \left(\frac{1}{4} + 0\right) = \frac{1}{2} - \alpha\frac{1}{4}.\end{aligned}$$

Once again, $\mathcal{A}$ can guess i^* with non-negligible probability ($\mathcal{A}$ is betting that $i^* = 1$), and since P_1 obtains a_{i^*} prior to P_2 obtaining b_{i^*}, the adversary can successfully bias P_2's output. We note that an adversary corrupting P_1 can bias P_2's output regardless of how we choose to describe the function. In other words, flipping some of the rows of M_1 and/or some of the columns of M_2 does not offer any solution, since the attacks above can be easily modified to successfully bias P_2's output. Nor is switching the players' roles helpful, i.e., considering (M_2^T, M_1^T), since $M_1 = M_2^T$.

To remedy this, we propose a new protocol, named $\textsc{FairTwoPartySpecial}$ (described in Figure 5), which foils the attacks described above and provides full security for f_{sp}. Our protocol is indistinguishable from the original GHKL protocol for every round, except special round i^*. Recall that in the GHKL protocol (as well as all other protocols we have considered), at round i^*, party P_1 obtains $a_{i^*} = f_1(x, y)$ followed by P_2 who obtains $b_{i^*} = f_2(x, y)$. We can reverse the player's roles, however, there is a party that always gets the correct output before the other party. We now make the following modification for the protocol computing f_{sp} : $a_{i^*} = f_1(x, y)$ only if $x \in \{x_1, x_2\}$, and $a_{i^*} = f_1(x, \widetilde{y}^{(i^*)})$ otherwise, where $\widetilde{y}^{(i^*)}$ is chosen uniformly at random. Thus, for certain inputs, the second player P_2 effectively obtains the output first. We claim that this modification suffices to compute f_{sp} with full security. and we dedicate the rest of the section to the proof this claim.

Protocol FairTwoPartySpecial

1. The parties P_1 and P_2 hand their inputs, denoted x and y respectively, to the dealer.[a]
2. The dealer chooses $i^* \geq 1$ according to a geometric distribution with probability α.
3. The dealer computes $\text{out}_1 = f_1(x,y)$, $\text{out}_2 = f_2(x,y)$ and for $0 \leq i \leq r$

$$a_i = \begin{cases} f_1(x, \widetilde{y}^{(i)}) \text{ where } \widetilde{y}^{(i)} \in_U Y & \text{if } i < i^* \\ f_1(x, \widetilde{y}^{(i)}) \text{ where } \widetilde{y}^{(i)} \in_U Y & \text{if } i = i^* \text{ and } x \in \{x_3, x_4\} \\ \text{out}_1 & \text{otherwise} \end{cases}$$

$$b_i = \begin{cases} f_2(\widetilde{x}^{(i)}, y) \text{ where } \widetilde{x}^{(i)} \in_U X & \text{if } i < i^* \\ \text{out}_2 & \text{otherwise.} \end{cases}$$

4. The dealer gives b_0 to P_2.
5. For $i = 1, \ldots, r$,
   (a) The dealer gives a_i to P_1. If P_1 aborts, then P_2 outputs b_{i-1} and halts.
   (b) The dealer gives b_i to P_2. If P_2 aborts, then P_1 outputs a_i and halts.

[a] If x is not in the appropriate domain or P_1 does not hand an input, then the dealer sends $f_2(\hat{x}, y)$ (where $\hat{x}$ is a default value) to P_2, which outputs this value and the protocol is terminated. The case of an inappropriate y is dealt analogously.

Figure 5: Protocol FairTwoPartySpecial for securely computing a function f_{sp}.

Theorem 4.6. *For every $\alpha \leq 1/9$, protocol* FairTwoPartySpecial *is fully secure for f_{sp}.*

Proof. The proof follows the ideas of symmetric case (and Protocol FairTwoPartyAsymm$_0$) with two significant modifications. First, we need two distinct simulations for P_1 depending on the choice of inputs. In particular, if $\mathcal{A}$ hands x_1 or x_2 to $\mathcal{S}^{\mathcal{A}}$, then the simulation is exactly the same as in the symmetric case (or Protocol FairTwoPartyAsymm$_0$). If not, i.e., if $\mathcal{A}$ hands x_3 or x_4, then we simulate P_1 as if he were P_2 in the symmetric case. On the other hand, regarding the second player, we note that a difficulty arises due to the fact that P_2 obtains the output first depending on P_1's choice of input. Nevertheless, we claim that by simulating P_2 as if he were P_1 in the symmetric case (or Protocol FairTwoPartyAsymm$_0$), i.e., as if he *always* gets the output first, results in the correct simulator. Thus, we will first consider the case of a corrupt P_1 when the adversary hands either x_1 or x_2 to the simulator, followed by the security analysis of a corrupt P_2.

We only discuss the required changes in the security proof of Protocol FairTwoPartySpecial compared to the security proof of Protocol FairTwoPartyAsymm$_0$. As in the proofs of Theorem 3.3 and Theorem 4.1, we only need to compare the distribution of (a_i, Out_{P_2}) in both worlds given that the adversary aborts in round $i \leq i^*$. Similarly to the previous proofs, we compute vectors $\mathbf{q}_x^{(0)}$ and $\mathbf{q}_x^{(1)}$ where $q_x^{(a)}(y)$ is the desired probability that the output of P_2 in the simulation (given that the adversary has aborted in round $i < i^*$) is 1 when the input of P_1 is x, the input of P_2 is y and $a_i = a$. In contrast to the proofs of Theorem 3.3 and Theorem 4.1, we only need to consider $x \in \{x_1, x_2\}$, since we have a separate simulation for $x \in \{x_3, x_4\}$. However, we emphasize that the simulator can choose any input from the entire domain to send to the trusted party. By considering the four possible values of $(a_i, \mathrm{Out}_{P_2})$, we deduce that

$$\mathbf{q}_x^{(0)}(y) = s_y + \frac{\alpha}{(1-\alpha)(1-p_x)}(1 - f_1(x,y))(s_y - f_2(x,y))$$
$$\mathbf{q}_x^{(1)}(y) = s_y + \frac{\alpha}{(1-\alpha)p_x} f_1(x,y)(s_y - f_2(x,y))$$

and thus,

$$\mathbf{q}_{x_1}^{(0)} = \begin{pmatrix} 3/4 \\ 1/4 \\ 1/2 \\ 1/2 \end{pmatrix} + \frac{\alpha}{(1-\alpha)(1-p_{x_1})} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1/2 \end{pmatrix}, \quad \mathbf{q}_{x_1}^{(1)} = \begin{pmatrix} 3/4 \\ 1/4 \\ 1/2 \\ 1/2 \end{pmatrix} + \frac{\alpha}{(1-\alpha)p_{x_1}} \begin{pmatrix} -1/4 \\ 1/4 \\ -1/2 \\ 0 \end{pmatrix},$$

$$\mathbf{q}_{x_2}^{(0)} = \begin{pmatrix} 3/4 \\ 1/4 \\ 1/2 \\ 1/2 \end{pmatrix} + \frac{\alpha}{(1-\alpha)(1-p_{x_2})} \begin{pmatrix} -1/4 \\ 1/4 \\ 1/2 \\ 0 \end{pmatrix}, \quad \mathbf{q}_{x_2}^{(1)} = \begin{pmatrix} 3/4 \\ 1/4 \\ 1/2 \\ 1/2 \end{pmatrix} + \frac{\alpha}{(1-\alpha)p_{x_2}} \begin{pmatrix} 0 \\ 0 \\ 0 \\ -1/2 \end{pmatrix}.$$

We conclude that $M_2^T \mathbf{x}_x^{(a)} = \mathbf{q}_x^{(a)}$ for the following vectors $\mathbf{x}_x^{(a)}$:

$$\mathbf{x}_{x_1}^{(0)} = 1/4 \cdot (1,1,1,1)^T + \frac{\alpha}{(1-\alpha)(1-p_{x_1})} 1/2 \cdot (0,1,-1,0)^T$$
$$\mathbf{x}_{x_1}^{(1)} = 1/4 \cdot (1,1,1,1)^T + \frac{\alpha}{(1-\alpha)p_{x_1}} 1/4 \cdot (-3,-1,3,1)^T$$
$$\mathbf{x}_{x_2}^{(0)} = 1/4 \cdot (1,1,1,1)^T + \frac{\alpha}{(1-\alpha)(1-p_{x_2})} 1/4 \cdot (1,-1,-1,1)^T$$
$$\mathbf{x}_{x_2}^{(1)} = 1/4 \cdot (1,1,1,1)^T + \frac{\alpha}{(1-\alpha)p_{x_2}} 1/2 \cdot (0,-1,1,0)^T.$$

It remains to show that for some $\alpha \in (0,1)$ the above vectors become probability vectors - they already sum to 1, we just need them to be positive. A straightforward computation yields that any $\alpha \leq 1/9$ will do.

Corrupt P_2. We now consider the case of an adversary $\mathcal{A}$ corrupting P_2. As mentioned above, we construct a simulator $\mathcal{S}^{\mathcal{A}}$ given a black-box to $\mathcal{A}$ that is completely analogous to P_1's simulator in protocol FairTwoPartyAsymm$_\sigma$. Namely, say that $\mathcal{A}$ hands $y \in Y$ to $\mathcal{S}^{\mathcal{A}}$ for the computation of f. The simulator chooses i^* according to a geometric distribution with parameter α and,

- for $i = 0, \ldots, i^* - 1$, the simulator hands $b_i = f(\widetilde{x}^{(i)}, y)$ to $\mathcal{A}$, where $\widetilde{x}^{(i)} \in_U X$. If $\mathcal{A}$ decides to quit, $\mathcal{S}^{\mathcal{A}}$ sends y_0 according to distribution $\mathbf{y}_y^{(b_i)}$ (to be defined below), outputs $(b_0, \ldots, b_i)$ and halts.
- for $i = i^*$, the simulator sends y to the trusted party, receives $b = f_2(x,y)$ and hands $b_{i^*} = b$ to $\mathcal{A}$. If $\mathcal{A}$ decides to quit, $\mathcal{S}^{\mathcal{A}}$ outputs $(b_0, \ldots, b_{i^*})$ and halts.
- for $i = i^* + 1, \ldots, r$, the simulator hands $b_i = b$ to $\mathcal{A}$. If $\mathcal{A}$ decides to quit, $\mathcal{S}^{\mathcal{A}}$ outputs $(b_0, \ldots, b_i)$ and halts.

- If $\mathcal{A}$ has not quitted yet, $\mathcal{S}^{\mathcal{A}}$ outputs $(b_0, \dots, b_r)$ and halts.

For reasons mentioned in the proof of Theorem 3.3, we only need to compare the distributions $(\mathrm{View}_{\mathcal{A}}, \mathrm{Out}_{P_1})$ in the real and ideal worlds given that the adversary aborts in a round $1 \le i \le i^*$. Thus, in the rest of the proof we let i be the round in which the adversary aborts and assume that $1 \le i \le i^*$. The view of the adversary in both worlds is $b_0, \dots, b_i$. Notice that in both worlds $b_1, \dots, b_{i-1}$ are equally distributed and are independent of $(b_i, \mathrm{Out}_{P_1})$. Thus, we only compare the distribution of $(b_i, \mathrm{Out}_{P_1})$ in both worlds. Now, for every $x \in X$, $y \in Y$, and $b \in \{0,1\}$, define $q_y^{(b)}(x) \triangleq \Pr[f_1(x, y_0) = 1]$, where y_0 is chosen according to the distribution $\mathbf{y}_y^{(b)}$, and define the column vector $\mathbf{q}_y^{(b)} \triangleq (q_y^{(b)}(x))_{x \in X}$. Using this notation,

$$M_1 \mathbf{y}_y^{(b)} = \mathbf{q}_y^{(b)}, \tag{18}$$

where we represent the distribution $\mathbf{y}_y^{(b)} = (\mathbf{y}_y^{(b)}(y_0))_{y_0 \in Y}$ by a column vector. We now analyze the four options for the values of $(b_i, \mathrm{Out}_{P_1})$. Bare in mind that we need to carefully distinguish between $x \in \{x_1, x_2\}$ and $x \in \{x_3, x_4\}$ in the real world.

First case: $(b_i, \mathrm{Out}_{P_1}) = (0,0)$**.** In the real world, if $x \in \{x_1, x_2\}$, $\Pr[(b_i, \mathrm{Out}_{P_1}) = (0,0)] = (1-\alpha)(1-p_x)(1-s_y) + \alpha(1-f_1(x,y))(1-f_2(x,y))$. Otherwise, $\Pr[(b_i, \mathrm{Out}_{P_1}) = (0,0)] = (1-\alpha)(1-p_x)(1-s_y) + \alpha(1-p_x)(1-f_2(x,y))$. On the other hand, in the ideal world $\Pr[(b_i, \mathrm{Out}_{P_1}) = (0,0)] = (1-\alpha)(1-s_y)(1-q_y^{(0)}(x)) + \alpha(1-f_1(x,y))(1-f_2(x,y))$. To get full security we need that these two probabilities in the two worlds are the same, that is,

$$\begin{aligned}&(1-\alpha)(1-s_y)(1-q_y^{(0)}(x)) + \alpha(1-f_1(x,y))(1-f_2(x,y)) = \\ &\qquad\begin{cases} (1-\alpha)(1-p_x)(1-s_y) + \alpha(1-f_1(x,y))(1-f_2(x,y)) & \text{if } x \in \{x_1, x_2\} \\ (1-\alpha)(1-p_x)(1-s_y) + \alpha(1-p_x)(1-f_2(x,y)) & \text{if } x \in \{x_3, x_4\} \end{cases}.\end{aligned} \tag{19}$$

Equation (19) is equivalent to

$$\mathbf{q}_y^{(0)}(x) = \begin{cases} p_x & \text{if } x \in \{x_1, x_2\} \\ p_x + \dfrac{\alpha(p_x - f_1(x,y))(1-f_2(x,y))}{(1-\alpha)(1-s_y)} & \text{if } x \in \{x_3, x_4\} \end{cases}. \tag{20}$$

Let's compute vectors $\mathbf{q}_y^{(0)}$,

$$\mathbf{q}_{y_1}^{(0)} = \begin{pmatrix} 3/4 \\ 1/4 \\ 1/2 \\ 1/2 \end{pmatrix} + \frac{\alpha}{(1-\alpha)(1-s_{y_1})} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1/2 \end{pmatrix}, \quad \mathbf{q}_{y_2}^{(0)} = \begin{pmatrix} 3/4 \\ 1/4 \\ 1/2 \\ 1/2 \end{pmatrix} + \frac{\alpha}{(1-\alpha)(1-s_{y_2})} \begin{pmatrix} 0 \\ 0 \\ 1/2 \\ 0 \end{pmatrix},$$

$$\mathbf{q}_{y_3}^{(0)} = \begin{pmatrix} 3/4 \\ 1/4 \\ 1/2 \\ 1/2 \end{pmatrix} + \frac{\alpha}{(1-\alpha)(1-s_{y_3})} \begin{pmatrix} 0 \\ 0 \\ 1/2 \\ 0 \end{pmatrix}, \quad \mathbf{q}_{y_4}^{(0)} = \begin{pmatrix} 3/4 \\ 1/4 \\ 1/2 \\ 1/2 \end{pmatrix} + \frac{\alpha}{(1-\alpha)(1-s_{y_4})} \begin{pmatrix} 0 \\ 0 \\ -1/2 \\ 0 \end{pmatrix}.$$

and deduce that that $M_1 \mathbf{y}_y^{(0)} = \mathbf{q}_y^{(0)}$ for the following vectors $\mathbf{y}_y^{(0)}$:

$$\begin{aligned}
\mathbf{y}_{y_1}^{(0)} &= 1/4 \cdot (1,1,1,1)^T + \frac{\alpha}{(1-\alpha)(1-s_{y_1})} 1/2 \cdot (0,1,-1,0)^T \\
\mathbf{y}_{y_2}^{(0)} &= 1/4 \cdot (1,1,1,1)^T + \frac{\alpha}{(1-\alpha)(1-s_{y_2})} 1/2 \cdot (1,0,-1,0)^T \\
\mathbf{y}_{y_3}^{(0)} &= 1/4 \cdot (1,1,1,1)^T + \frac{\alpha}{(1-\alpha)(1-s_{y_3})} 1/2 \cdot (1,0,-1,0)^T \\
\mathbf{y}_{y_4}^{(0)} &= 1/4 \cdot (1,1,1,1)^T + \frac{\alpha}{(1-\alpha)(1-s_{y_4})} 1/2 \cdot (-1,0,1,0)^T.
\end{aligned}$$

Note that, to obtain probability vectors, any $\alpha \leq 9$ will do.

Second case: $(b_i, \mathrm{Out}_{P_1}) = (0,1)$**.** In the real world if $x \in \{x_1, x_2\}$, $\Pr[(b_i, \mathrm{Out}_{P_1}) = (0,1)] = (1-\alpha)p_x(1-s_y) + \alpha f_1(x,y)(1-f_2(x,y))$. Otherwise, $\Pr[(b_i, \mathrm{Out}_{P_1}) = (0,0)] = (1-\alpha)p_x(1-s_y) + \alpha p_x(1-f_2(x,y))$. On the other hand, in the ideal world $\Pr[(b_i, \mathrm{Out}_{P_1}) = (0,1)] = (1-\alpha)(1-s_y)(1-q_y^{(0)}(x)) + \alpha f_1(x,y)(1-f_2(x,y))$. (in the ideal world $a_{i^*} = f_1(x,y)$ and $\mathrm{Out}_{P_2} = f_2(x,y)$). The probabilities in the two worlds are equal if (20) holds.

Third case: $(b_i, \mathrm{Out}_{P_1}) = (1,0)$**.** In the real world, if $x \in \{x_1, x_2\}$, $\Pr[(b_i, \mathrm{Out}_{P_1}) = (1,0)] = (1-\alpha)(1-p_x)s_y + \alpha(1-f_1(x,y))f_2(x,y)$. Otherwise, $\Pr[(b_i, \mathrm{Out}_{P_1}) = (1,0)] = (1-\alpha)(1-p_x)s_y + \alpha(1-p_x)f_2(x,y)$. On the other hand, in the ideal world $\Pr[(b_i, \mathrm{Out}_{P_1}) = (1,0)] = (1-\alpha)s_y(1-q_y^{(0)}(x)) + \alpha(1-f_1(x,y))f_2(x,y)$. The probabilities in the two worlds are equal when

$$(1-\alpha)s_y(1-q_y^{(0)}(x)) + \alpha(1-f_1(x,y))f_2(x,y) = \\ \begin{cases} (1-\alpha)(1-p_x)s_y + \alpha(1-f_1(x,y))f_2(x,y) & \text{if } x \in \{x_1,x_2\} \\ (1-\alpha)(1-p_x)s_y + \alpha(1-p_x)f_2(x,y) & \text{if } x \in \{x_3,x_4\} \end{cases}.$$

This equality holds if and only if

$$\mathbf{q}_y^{(1)}(x) = \begin{cases} p_x & \text{if } x \in \{x_1,x_2\} \\ p_x + \dfrac{\alpha(p_x - f_1(x,y))f_2(x,y)}{(1-\alpha)s_y} & \text{if } x \in \{x_3,x_4\} \end{cases}. \tag{21}$$

Let's compute the vectors $\mathbf{q}_y^{(1)}$,

$$\mathbf{q}_{y_1}^{(1)} = \begin{pmatrix} 3/4 \\ 1/4 \\ 1/2 \\ 1/2 \end{pmatrix} + \frac{\alpha}{(1-\alpha)s_{y_1}} \begin{pmatrix} 0 \\ 0 \\ -1/2 \\ 0 \end{pmatrix}, \quad \mathbf{q}_{y_2}^{(1)} = \begin{pmatrix} 3/4 \\ 1/4 \\ 1/2 \\ 1/2 \end{pmatrix} + \frac{\alpha}{(1-\alpha)s_{y_2}} \begin{pmatrix} 0 \\ 0 \\ 0 \\ -1/2 \end{pmatrix},$$

$$\mathbf{q}_{y_3}^{(1)} = \begin{pmatrix} 3/4 \\ 1/4 \\ 1/2 \\ 1/2 \end{pmatrix} + \frac{\alpha}{(1-\alpha)s_{y_3}} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1/2 \end{pmatrix}, \quad \mathbf{q}_{y_4}^{(1)} = \begin{pmatrix} 3/4 \\ 1/4 \\ 1/2 \\ 1/2 \end{pmatrix} + \frac{\alpha}{(1-\alpha)s_{y_4}} \begin{pmatrix} 0 \\ 0 \\ 0 \\ -1/2 \end{pmatrix}.$$

and deduce that that $M_1\mathbf{y}_y^{(1)} = \mathbf{q}_y^{(1)}$ for the following vectors $\mathbf{y}_y^{(1)}$:

$$\begin{aligned}
\mathbf{y}_{y_1}^{(1)} &= 1/4 \cdot (1,1,1,1)^T + \frac{\alpha}{(1-\alpha)s_{y_1}} 1/2 \cdot (-1,0,1,0)^T \\
\mathbf{y}_{y_2}^{(1)} &= 1/4 \cdot (1,1,1,1)^T + \frac{\alpha}{(1-\alpha)s_{y_2}} 1/2 \cdot (0,-1,1,0)^T \\
\mathbf{y}_{y_3}^{(1)} &= 1/4 \cdot (1,1,1,1)^T + \frac{\alpha}{(1-\alpha)s_{y_3}} 1/2 \cdot (0,1,-1,0)^T \\
\mathbf{y}_{y_4}^{(1)} &= 1/4 \cdot (1,1,1,1)^T + \frac{\alpha}{(1-\alpha)s_{y_4}} 1/2 \cdot (0,-1,1,0)^T.
\end{aligned}$$

Regarding the value of α, in order to obtain probability vectors, any $\alpha \leq 9$ will do. Finally, since the equalities hold for the first 3 cases of the values for $(b_i, \mathrm{Out}_{P_1})$, the same must be true of the case $(b_i, \mathrm{Out}_{P_1}) = (1,1)$.

□

5 Characterization of Fairness when Half of the Parties are Honest

In this section, we extend our discussion to the case of any constant number of parties, where at most half of the parties are corrupted. Using the characterization of symmetric two-party functionalities from Section 3, we fully characterize the set of functions that can be computed with full security in this setting. In this section, we only consider symmetric functions, i.e., where all parties receive the same output.

The main result of this section is a characterization of secure computation with full security, whenever at most half of the parties are corrupted. This characterization is stated in Theorem 5.1 below. The proof of the theorem follows from the combination of Lemma 5.4, given in Section 5.1, and Theorems 5.5 and 5.6, given in Section 5.2. We emphasize that most previous works obtaining fully secure multiparty computation (with the exception of [11]) assumed a strict honest majority. Here, we consider the case, where such a majority is not guaranteed.

Let $X = X_1 \times X_2 \times \cdots \times X_m$ and let $f : X \to R$ be some function. For a subset $\emptyset \subset I \subset [m]$ let $\bar{I} = [m] \setminus I$ and let X_I be the projection of X on I. Similarly, for an input $\mathbf{x} \in X$, let $\mathbf{x}_I$ be the projection of $\mathbf{x}$ on I. We define the function $f_I : X_I \times X_{\bar{I}} \to R$, by $f_I(\mathbf{w}, \mathbf{z}) = f(\mathbf{x})$, where $\mathbf{x}$ is such that $\mathbf{x}_I = \mathbf{w}$ and $\mathbf{x}_{\bar{I}} = \mathbf{z}$.

Theorem 5.1. *Let $m = 2k$ be a constant, let $X = X_1 \times X_2 \times \cdots \times X_m$ be a finite domain, and let $f : X \to R$ be a deterministic function. Then, f is computable with full security in the multiparty setting against an adversary that can corrupt up to k parties if and only if for every subset $I \subset [m]$, with $|I| = k$, the function f_I is computable with full security in the two-party setting. The same is true if f is a randomized Boolean function.*

The proof of the feasibility part of Theorem 5.1 follows by defining a protocol (with an on-line dealer), in which the dealer runs many two-party protocols simultaneously. More specifically, for each subset $I \subset [m]$ of size k, the dealer runs the appropriate two-party protocol Π_I that securely computes the two-party function f_I. The description of Protocol FairMultiParty, proving the feasibility, appears in Figure 6. The proof of its security appears in Section 5.2. The explanation of how the on-line dealer is eliminated appears in Section 5.5. The proof of the necessary condition also appears in Section 5.1.

Protocol FairMultiParty

1. The parties $P_1, \ldots, P_m$ hand their inputs, denoted $\mathbf{x} = x_1, \ldots, x_m$, respectively, to the dealer. If a party P_j does not send an input, then the dealer selects $x_j \in X_j$ uniformly at random. If half of the parties do not send an input, then the dealer sends $f(x_1, \ldots, x_m)$ to the honest parties and halts.
2. The dealer computes for every $I \subset [m]$, such that $|I| = k = m/2$, and for every $0 \leq i \leq r$ the backup outputs $a_i^{(I)}$ and $b_i^{(I)}$ using Π_I.
3. The dealer shares $b_0^{(I)}$ among the parties of $S_I^{(2)}$ for each I as above, in a k-out-of-k Shamir secret-sharing scheme.
4. For $i = 1, \ldots, r_f$,
   (a) The dealer shares $a_i^{(I)}$ among the parties of $S_I^{(1)}$ for each I as above, in a k-out-of-k Shamir secret-sharing scheme. If *all* the parties of some subset $S_I^{(1)}$ abort, then the parties in $S_I^{(2)}$ reconstruct b_{i-1}^I, output it and halt.
   (b) The dealer shares $b_i^{(I)}$ among the parties of $S_I^{(2)}$ for each I as above, in a k-out-of-k Shamir secret-sharing scheme. If *all* the parties of some subset $S_I^{(2)}$ abort, then the parties in $S_I^{(1)}$ reconstruct a_i^I, output it and halt.
5. All subsets $S_I^{(1)}$ and $S_I^{(2)}$ (for all sets I as above) reconstruct a_r^I and b_r^I, respectively, and output it (by correctness, with all but negligible probability in the security parameter – all successfully reconstructed values are equal).

Figure 6: Protocol FairMultiParty for securely computing a function f, if at least half of the parties are honest.

We next introduce some of the notation from Section 5.2 that is necessary for understanding the description of Protocol FairMultiParty in Figure 6. We assume that each Π_I is an r_I-round protocol with an on-line dealer for securely computing the two-party function f_I. By Lemma 5.10 (appearing in Section 5.4), this is without loss of generality. We also assume that for each such I, the protocols Π_I and the protocol $\Pi_{\bar{I}}$ refer to the same protocol (if this is not the case, we fix one arbitrarily). We, therefore, only consider subsets $I \subset [m]$, such that $1 \in I$ and $|I| = k$ (the choice of 1 is arbitrary, made to enforce an ordering on $\{I, \bar{I}\}$).

Notation 5.2. *For each I as above. Let A_I be the party that plays the role of A in Π_I, and let B_I be the other party in this protocol. Denote by $a_i^{(I)}$ (resp. $b_i^{(I)}$) the backup output that part A_I (resp. B_I) receives in round i of Π_I. In addition, let $S_I^{(1)}$ the set of parties whose inputs correspond to the input of A_I, that is, $S_{J_I}^{(1)} = \{P_j : j \in J\}$ such that X_J is the domain from which the input of A is taken; let $S_I^{(2)}$ be the remaining parties (i.e., whose inputs correspond to that of party B). Let r_f be the maximum over all r_I for I as above.*

In this section we give the missing results and proof that yield the characterization of secure computation with full security, whenever at most half of the parties are corrupted. This characterization was stated in Theorem 5.1, restated below.

Theorem 5.3. *[Theorem 5.1, restated] Let $m = 2k$ be a constant, let $X = X_1 \times X_2 \times \cdots \times X_m$ be a finite domain, and let $f : X \to R$ be a deterministic function. Then, f is computable with full security in the multiparty setting against an adversary that can corrupt up to k parties if and only if for every subset $I \subset [m]$, with $|I| = k$, the function f_I is computable with full security in the two-party setting. The same is true if f is a randomized Boolean function.*

The proof of the theorem follows from the combination of Lemma 5.4, given in Section 5.1, and Theorems 5.5 and 5.6, given in Section 5.2.

5.1 Limits on Secure Multiparty Computation with a Dishonest Majority

A straightforward limitation on the feasibility of securely computing multiparty functionalities follows from a simple reduction from the two-party case. For completeness, we introduce the following lemma.

Lemma 5.4. *Let $X = X_1 \times X_2 \times \cdots \times X_m$ and let $f : X \to R$. Let $t \leq m$. Assume that there exists a subset $\emptyset \subset I \subset [m]$, with $|I|, \left|\bar{I}\right| \leq t$, such that f_I is not computable with full security in the two-party setting. Then, f is not computable with full security in the multiparty setting against an adversary that can corrupt up to t parties.*

Proof. Assume towards a contradiction that there exists a secure protocol Π for computing f that is secure against adversaries that may corrupt at most t of the parties $P_1, \ldots, P_m$. Consider the following two-party protocol Π_I for computing f_I. In this protocol party A plays the role of all parties P_j such that $j \in I$, and party B plays the role of the remaining parties. To simulate an adversary corrupting party A or party B in Π_I, apply the simulator of Π for the analogous adversary corrupting the corresponding parties in the original protocol. The simulator of Π can simulate either of these adversaries, since $|I|, \left|\bar{I}\right| \leq t$.

Hence, Π_I is a secure two-party protocol for computing f_I, in contradiction to the assumption of the lemma. □

5.2 Feasibility of Secure Computation with Fairness when Half of the Parties are Honest

In this section, we restrict our discussion to (constant) $m = 2k$ and we consider the feasibility of computing f against adversaries that are able to corrupt up to k of the parties. Indeed, for such adversaries we show that the converse of Lemma 5.4 is true. We start by proving the feasibility result for any deterministic function. Then, we prove a similar result for Boolean (possibly randomized) functions.

Theorem 5.5. *Let $m = 2k$ be a constant, let $X = X_1 \times X_2 \times \cdots \times X_m$ be a finite domain, and let $f : X \to R$ be a deterministic function. Assume that for every subset $I \subset [m]$, with $|I| = k$, the function f_I is computable with complete fairness in the two-party setting. Then, f is computable with complete fairness in the multiparty setting against an adversary that can corrupt up to k parties.*

Th formal proof of Theorem 5.5, appears in Section 5.2.1. Intuitively, the proof follows by defining a protocol (with an on-line dealer), in which the dealer runs many two-party protocols simultaneously. More specifically, for each partitioning of the m parties into two sets of size k, the dealer runs the appropriate two-party protocol, existing by the assumption of the theorem (in Section 5.4, we formally assert that such a protocol also exists in the on-line dealer setting).

Using a similar approach, however, using the specific two-party protocols described in Section 3, we obtain a similar result for randomized Boolean function.

Theorem 5.6. *Let $m = 2k$ be a constant, let $X = X_1 \times X_2 \times \cdots \times X_m$ be a finite domain, and let $f : X \to \{0, 1\}$ be a (possibly randomized) Boolean function. Assume that for every subset $I \subset [m]$, with $|I| = k$, the function f_I is computable with complete fairness in the two-party setting. Then, f is computable with complete fairness in the multiparty setting against an adversary that can corrupt up to k parties.*

As mentioned above, to prove the this theorem, we use the same multiparty protocol, only with instantiations of the two-party protocols presented in Section 3. Doing so, we can ensure that at the end of the protocol, all parties receive the same output of the function (i.e., computed wuth the same random coins). Hence, the adversary cannot see multiple outputs of the function applied to the inputs of the honest parties. The formal proof of Theorem 5.6, appears in Section 5.3.

5.2.1 Proving Theorem 5.5

Recall that, by assumption of the theorem, for every $I \subset [m]$ such that $|I| = k$, there exists a protocol Π_I for securely computing the two-party function f_I. In Section 5.4, we show that this also implies the existence of a secure on-line dealer protocol for securely computing the two-party function f_I.

We, henceforth, assume that each Π_I is an r_I-round protocol with an on-line dealer for securely computing the two-party function f_I. By Lemma 5.10, this is without loss of generality. We also assume that for each such I, the protocols Π_I and the protocol $\Pi_{\bar{I}}$ refer to the same protocol (if this is not the case, we fix one arbitrarily). We, therefore, only consider subsets $I \subset [m]$, such that $1 \in I$ and $|I| = k$ (the choice of 1 is arbitrary, made to enforce an ordering on $\{I, \bar{I}\}$). We introduce the following notation (which is the same as Notation 5.2).

Notation 5.7. *For each I as above. Let A_I be the party that plays the role of A in Π_I, and let B_I be the other party in this protocol. Denote by $a_i^{(I)}$ (resp. $b_i^{(I)}$) the backup output that part A_I (resp. B_I) receives in round i of Π_I. In addition, let $S_I^{(1)}$ the set of parties whose inputs correspond to the input of A_I, that is, $S_{J_I}^{(1)} = \{P_j : j \in J\}$ such that X_J is the domain from which the input of A is taken; let $S_I^{(2)}$ be the remaining parties (i.e., whose inputs correspond to that of party B). Let r_f be the maximum over all r_I for I as above.*

The formal definition of Protocol FairMultiParty is presented in Figure 6. Let us now give a less formal description of the protocol. As mentioned above, the protocol will simultaneously emulate an execution of the protocol Π_I for each function f_I (where the sets I are as described above).

For each I, the parties in $S_I^{(1)}$ will play the role of A_I and the parties in $S_I^{(2)}$ will play the role of B_I. Specifically, in each round i of the multiparty protocol, the dealer first shares the value that would be given to party A_I (in the two-party protocol) among the parties in $S_I^{(1)}$ in a k-out-of-k Shamir secret-sharing scheme. Then (in the second part of round i), the dealer shares the value that would be given to party B_I among the parties in $S_I^{(2)}$ in a k-out-of-k Shamir secret-sharing scheme. Jumping ahead, the intuition behind using this secret-sharing scheme is the key property that if all parties in a subset $S_I^{(1)}$ (resp. $S_I^{(2)}$) work together to reconstruct the i'th value, then they are able to, however, even if one party does not participate, then they can learn nothing about the secret. This will ensure that if there is at least one honest party in a set, then the secret will not be prematurely revealed to the adversary. Furthermore, if all parties are honest, then the adversary cannot prevent the reconstruction of the secret.

A key point in our protocol is that dealer will continue with running the protocol as prescribed as long as less than k parties have aborted the execution. This ensures that, if the normal execution has prematurely halted, then all the remaining active parties are honest. We emphasize that our goal is to construct a protocol without any dealer for computing f, and hence, we need to claim that such a dealer can be replaced by a distributed computation. In Section 5.5, we explain how this is done, using similar ideas to those of [5]. Furthermore, we assume that parties are fail-stop, meaning that the only way to deviate from the prescribed protocol is by prematurely aborting the computation. This is later enforced by adding a digital signature to any possible message in the protocol, where the verification key is public and the signing key is secret.

The following claim concludes the proof of Theorem 5.5.

Claim 5.8. *Protocol* FairMultiParty *is a fully-secure protocol for f, tolerating coalitions of size at most k.*

Proof. We describe a simulator $\mathcal{S}$, with access to a real-world adversary $\mathcal{A}$, considering an execution of $\mathcal{S}$ with auxiliary input z. The simulator $\mathcal{S}$ invokes the adversary $\mathcal{A}$ on the auxiliary input z. Let T be the set of parties that $\mathcal{A}$ chooses to corrupt. For simplicity of presentation, assume that $|T| = k$ and that $1 \in T$; these two assumptions are without loss of generality, since if the former is not true, the simulation becomes easy, and the latter simply allows using T rather than $\bar{T}$ when applying Notation 5.7.

A key point in the simulation is that all $\mathcal{A}$ learns during the computation (besides shares of secrets, which it is unable to reconstruct) are the backup outputs that a single corrupted party would see in the analogous execution of of the two-party protocol Π_T for computing (the two-party function) f_T. Furthermore, if the protocol prematurely terminates, (namely, if all the parties in T abort), then the remaining k parties (which are all honest) output what the honest party would output in the analogous two-party protocol execution. Hence, for the most part, the simulation works by $\mathcal{S}$ applying the appropriate simulator, denoted $\mathcal{S}_{\text{two}}$, of the two-party protocol Π_T. In the following we describe how $\mathcal{S}$ interacts with $\mathcal{S}_{\text{two}}$.

- The simulator $\mathcal{S}$ constructs the following adversary $\mathcal{A}_{\text{two}}$ for Π_T, playing the same role as $\mathcal{A}$ (that is, if $\mathcal{A}$ corrupted the set $S_I^{(1)}$, then $\mathcal{A}_{\text{two}}$ will play the role of A_T, and otherwise, $\mathcal{A}_{\text{two}}$ will play the role of B_T):
  - The adversary $\mathcal{A}_{\text{two}}$ invokes $\mathcal{A}$ expecting its inputs x_T (playing the role of the dealer). If the adversary fails to send an input for a party $P_j \in T$, then $\mathcal{A}_{\text{two}}$ selects $x_j \in X_j$ uniformly at random. If $\mathcal{A}$ sends no input, then $\mathcal{A}_{\text{two}}$ aborts the computation without sending any input to its dealer.
  - $\mathcal{A}_{\text{two}}$ sends x_T to the dealer of Π_T.
  - For $i = 1, \ldots, r$, upon receiving backup value d_i

1. If $\mathcal{A}$ controls the parties in $S_T^{(1)}$, then the adversary $\mathcal{A}_{\text{two}}$ shares d_i among the parties of $S_T^{(1)}$ in a k-out-of-k Shamir secret-sharing scheme. In addition, $\mathcal{A}_{\text{two}}$ gives shares of 0 to the *corrupted* parties within $S_I^{(1)}$ for each $I \neq T, \bar{T}$ (in a k-out-of-k Shamir secret-sharing scheme).

   If *all* the parties of some subset $S_T^{(1)}$ abort, then $\mathcal{A}_{\text{two}}$ aborts, and otherwise it continues.
2. If $\mathcal{A}$ controls the parties in $S_T^{(2)}$, then the adversary $\mathcal{A}_{\text{two}}$ shares d_i among the parties of $S_T^{(2)}$ in a k-out-of-k Shamir secret-sharing scheme. In addition, $\mathcal{A}_{\text{two}}$ gives shares of 0 to the *corrupted* parties within $S_I^{(2)}$ for each $I \neq T, \bar{T}$ (in a k-out-of-k Shamir secret-sharing scheme).

   If *all* the parties of some subset $S_T^{(2)}$ abort, then $\mathcal{A}_{\text{two}}$ aborts, and otherwise it continues.

– After round r is completed, seeing d_r, the adversary $\mathcal{A}_{\text{two}}$ selects shares for the honest parties, such that the r backup output of all subsets seen by some corrupted party reconstruct to d_r (this is possible by the properties of the Shamir secret-sharing scheme, since for the corrupted parties could never reconstruct any of the 0 secrets).

- Let $\mathcal{S}_{\text{two}}$ be the simulator for $\mathcal{A}_{\text{two}}$.
- The simulator $\mathcal{S}$ interacts with the two-party protocol simulator $\mathcal{S}_{\text{two}}$, by invoking it on the adversary $\mathcal{A}_{\text{two}}$ (with input x_T). It then receives a simulated view for $\mathcal{A}_{\text{two}}$, containing its random coins and its backup outputs. Having received this view of $\mathcal{A}_{\text{two}}$, the simulator $\mathcal{S}_{\text{two}}$ can extract from it the view of $\mathcal{A}$ in this execution, as it is implied view of $\mathcal{A}_{\text{two}}$. Specifically, the randomness $\mathcal{A}_{\text{two}}$ uses to share different secrets with determines the shares that the corrupted parties see.

To conclude the proof, we analyze the simulation. First note, that by not sending some of the inputs to the dealer, the adversary simply allows the dealer to choose uniform inputs, which is equivalent to the adversary doing so in the first place. Thus, the distribution over the input of $\mathcal{A}_{\text{two}}$ is as implied by the behavior of $\mathcal{A}$.

The key point in the analysis is that the shares of backup values that the corrupted parties see for all sets that are not $\{P_j : j \in T\}$ are identically distributed whether these are shares of real backup values or of 0. Thus, the view of $\mathcal{A}$ in the interaction with the real dealer is distributed identically to its view when interacting with $\mathcal{A}_{\text{two}}$ throughout the execution until the end of round r.

Indeed, the only difference may occur in the opening of the values of all the backup outputs of round r, where in the interaction with $\mathcal{A}_{\text{two}}$, all these values are always equal. In the real protocol, however, it may happen with some negligible probability that not all values are the same. However, this small error probability is allowed and, hence, we can ignore it in the rest of the discussion.

Finally, note that, by definition, the honest parties in an execution of Protocol FairMultiParty with $\mathcal{A}$ output the same output as does the honest party in an execution of Π_T in Π_T. Hence, the distribution over the view of $\mathcal{A}_{\text{two}}$ jointly with the output of the honest party in an execution of Π_{two} is almost identically distributed as the view of $\mathcal{A}$ together with the output of the honest parties. □

5.3 Proving Theorem 5.6

The proof follows by defining a protocol (with an on-line dealer), very similar to Protocol FairMultiParty. However, we cannot simply apply the same construction for randomized functionalities, since the last step of the protocol (in which all subsets reconstruct their last backup output) may reveal many outputs of the function. We next prove that the construction does work for randomized *Boolean* functions, by utilizing a simple property of Protocol FairTwoParty.

Proof sketch. Recall that in Protocol FairMultiParty, the dealer simultaneously runs a two-party protocol Π_I for computing the function f_I for each partitioning implied by a subset I of size k. Here, we use the fact that in all executions of Protocol FairTwoParty, we can use the same value for the backup outputs given to both parties starting from round $i^\star$ in each protocol.

The simulation of the protocol follow the same lines as the simulation of Protocol FairMultiParty, with the only difference, that here it also be true in the for the real execution of the protocol that all the parties agree on the same output if the execution of round r goes through. $\square$

5.4 Two-Party Protocols with a Dealer

Definition 5.9 (Two-party protocols with a dealer)**.** *Given an r-round protocol Π between two parties A and B with inputs x and y, respectively. Let Π_{OL} denote the following on-line dealer protocol with parties A_{OL} and B_{OL}. Assume, without loss of generality, that in each round i, party A first sends its i'th message to B and then party B sends its i'th message to A.*

1. *The parties A_{OL} and B_{OL} send their inputs x and y (respectively) to the dealer.*
2. *The dealer locally computes a full execution of Π on inputs x, y. Let v_{A} (resp. v_{B}) be the view of party A (resp. B) in this execution. For each $i = 0, \ldots, r$, let b_i be the output of party A according to v_{A} in the case that party party A does not send its message (or sends an illegal message) in round $i+1$ let a_i be the output of party A according to v_{A} in the case that party party B does not send its message (or sends an illegal message) in round i (specifically, if both parties play the whole execution, then a_r and b_r are the outputs of A and B, respectively). We call a_i, b_i the* backup outputs*.*
3. *The dealer sends b_0 to party B.*
4. *For $i = 1, \ldots, r$,*
   (a) *The dealer sends a_i to A. If A aborts, then B outputs b_{i-1} and halts.*
   (b) *The dealer sends b_i to B. If B aborts, then A outputs a_i and halts.*

We note that for our on-line two-party protocols to coincide with the notation of the above transformation, we simply let A send a $\perp$ message in the first round of the no dealer protocol.

Lemma 5.10. *Let Π be an r-round, fully secure, two-party protocol without a dealer for computing a function $f \colon X \times Y \mapsto R$, then Π_{OL} is an r-round, fully secure, two-party, on-line dealer protocol for f.*

Proof. To construct a simulator $\mathcal{S}_{\mathrm{OL}}$ for Π_{OL}, we use the simulator $\mathcal{S}$ of Π. To simulate an adversary $\mathcal{A}$, we define an intermediate adversary $\mathcal{A}_\diamond$ for the original protocol Π. On input x, $\mathcal{A}_\diamond$ selects uniform random coins (as an honest party) and acts as follows: before sending each message in the protocol, it computes its current backup output and sends it to $\mathcal{A}$ (this is possible, since the backup output is a function of the current view of the party). If $\mathcal{A}$ aborts after seeing this backup output, then so will $\mathcal{A}_\diamond$, and otherwise $\mathcal{A}_\diamond$ sends to the other party the message that the appropriate honest party would, given its view so far.

Now, to simulate $\mathcal{A}$, the simulator $\mathcal{S}_{\mathrm{OL}}$ accepts an input $\hat{x}$ from $\mathcal{A}$ and invokes simulator $\mathcal{S}$ of Π on the adversary $\mathcal{A}_\diamond$ with input $\hat{x}$ to obtain a j-round view $\hat{v}$. Then, the simulator $\mathcal{S}_{\mathrm{OL}}$ outputs the j backup outputs implied by $\hat{v}$ as the view of $\mathcal{A}$. If $\mathcal{A}$ never sends an input to $\mathcal{S}_{\mathrm{OL}}$, then $\mathcal{S}_{\mathrm{OL}}$ sends a uniformly chosen input for $\mathcal{A}$ to the trusted party and aborts.

To see that this simulation is good, note that by the security of Π, the simulator $\mathcal{S}$ must be able to simulate $\mathcal{A}_\diamond$. Furthermore, if $\mathcal{A}$ does send its input to $\mathcal{S}_{\mathrm{OL}}$, then the distribution over the backup outputs of $\mathcal{A}_\diamond$ together with the output of the honest party (in Π) is exactly the same as the distribution over the view of $\mathcal{A}$ together with the output of the honest party (in Π_{OL}). $\square$

5.5 Eliminating the Dealer of the Protocol

Following [4, 5], we eliminate the trusted on-line dealer of our multiparty protocols in a few steps using a few layers of secret-sharing schemes. In the first step, we convert the on-line dealer to an off-line dealer. That is, we construct a protocol in which the dealer sends only one message to each party in an initialization stage; the parties then interact in rounds using a broadcast channel (without the dealer) and in each sub-round of round i each party learns its shares of the i'th round. Specifically, in round i, party P_j learns a share in a k-out-of-k secret-sharing scheme, for every set S of size k to which p_j belongs (we call these shares, P_j's shares of the inner secret-sharing scheme).

For this purpose, the dealer computes, in a preprocessing phase, the appropriate shares for the inner secret-sharing scheme. For each round, the shares of each party P_j are then shared in a *special* 2-out-of-2 secret-sharing scheme, where P_j gets one of the two shares (this share is a mask, enabling P_j to privately reconstruct its shares of although messages are sent on a broadcast channel). All other parties receive shares in a k-out-of-$(m-1)$ Shamir secret-sharing scheme of the other share of the 2-out-of-2 secret sharing. We call the resulting secret-sharing scheme the *outer* $(k+1)$-out-of-m scheme (since k parties and the holder of the mask are needed to reconstruct the secret).

The use of the outer secret-sharing scheme with threshold $k+1$ plays a crucial role in eliminating the on-line dealer. On the one hand, it guarantees that an adversary, corrupting at most k parties, cannot reconstruct the shares of round i before round i. On the other hand, at least k parties must abort to prevent the reconstruction of the outer secret-sharing scheme. Furthermore, when k corrupted parties abort, there are only honest parties remaining. In this case we can execute a protocol with full security for the reconstruction. Finally, we replace the off-line dealer by using a secure-with-abort and cheat-detection protocol computing the functionality computed by the dealer.

To prevent corrupted parties from cheating, by say, sending false shares and causing reconstruction of wrong secrets, every message that a party should send during (any possible flow of) the execution of the protocol is signed in the preprocessing phase (together with the appropriate round number and with the party's index). In addition, the dealer sends a verification key to each of the parties. To conclude, the off-line dealer gives each party the signed shares for the outer secret-sharing scheme together with the verification key.

6 Open Problems

As a conclusion, we provide a short list of questions that are left unanswered by our paper. First, regarding asymmetric functionalities, it would be interesting to know if a generalized version of Protocol
FairTwoPartySpecial offers any significant improvement toward bridging the gap in the asymmetric case. In Protocol FairTwoPartySpecial, for certain inputs P_1 gets the output first, and for others P_2 gets the output first. In our protocol this partition depends only on the input of P_1. The question is which asymmetric functions can be computed with full security when the protocol uses an arbitrary partition of the parties' inputs.

On the other hand, some functions that lie in the gap might be outright impossible to compute with full security. If true, then these functions do not imply non-trivial sampling by the construction[3] of [12], and thus, any impossibility result would require a new argument. We provide an example of a function that lies in the gap, and whose fairness does not seem to derive from the work of the present paper.

$$M_1 = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \qquad M_2 = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}.$$

Another direction of inquiry would be the computation of Boolean functions in the multi-party setting, where the corrupted parties form a strict majority. In particular, there may be room to generalize protocol FairMultiParty to handle strict majorities of corrupted parties. However, several difficulties arise in view of the fact that the adversary would have access to the backup outputs of multiple partial functions, and not just one.

Furthermore, the characterization of functions with arbitrary output domains, a subject already touched upon by [2], seems like a hard problem to tackle.

Our protocols assume that the size of the input domain is a constant independent of the security parameter. It can be shown that in our protocols for two parties, if the size of the input domains is $\log n$ (where n is the security parameter), then the number of rounds and the computation are still polynomial in n. It would be interesting to construct protocols for families of functions $\{f_n\}_{n \in \mathbb{N}}$, where the size of the domain of f_n is polynomial or even exponential in n.

References

[1] S. Agrawal and M. Prabhakaran. On fair exchange, fair coins and fair sampling. In *Advances in Cryptology – CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 259–276. Springer-Verlag, 2013.

[2] G. Asharov. Towards characterizing complete fairness in secure two-party computation. In *Proc. of the Eleventh Theory of Cryptography Conference – TCC 2014*, volume 8349, pages 291–316. Springer-Verlag, 2014.

[3] G. Asharov, Y. Lindell, and T. Rabin. A full characterization of functions that imply fair coin tossing and ramifications to fairness. In *Proc. of the Tenth Theory of Cryptography Conference*

[3]The construction is also presented in the proof of Theorem 4.4 in Section 4.2.

– *TCC 2013*, volume 7785 of *Lecture Notes in Computer Science*, pages 243–262. Springer-Verlag, 2013.

[4] A. Beimel, Y. Lindell, E. Omri, and I. Orlov. $1/p$-secure multiparty computation without honest majority and the best of both worlds. In *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 277–296. Springer-Verlag, 2011.

[5] A. Beimel, E. Omri, and I. Orlov. Protocols for multiparty coin toss with dishonest majority. *J. of Cryptology*, 2013. To appear. Conference version in: T. Rabin, editor, Advances in Cryptology – CRYPTO 2010, volume 6223 of Lecture Notes in Computer Science, pages 538-557. Springer-Verlag, 2010.

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

[7] R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proc. of the 18th ACM Symp. on the Theory of Computing*, pages 364–369, 1986.

[8] O. Goldreich. *Foundations of Cryptography, Voume II Basic Applications.* Cambridge University Press, 2004.

[9] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. of the 19th ACM Symp. on the Theory of Computing*, pages 218–229, 1987.

[10] S. D. Gordon, C. Hazay, J. Katz, and Y. Lindell. Complete fairness in secure two-party computation. *J. of the ACM*, 58(6):Article No. 24, 2011.

[11] S. D. Gordon and J. Katz. Complete fairness in multi-party computation without an honest majority. In *Proc. of the Sixth Theory of Cryptography Conference – TCC 2009*, pages 19–35, Berlin, Heidelberg, 2009. Springer-Verlag.

[12] N. Makriyannis. On the classification of finite boolean functions up to fairness. In *Security and Cryptography for Networks – 9th International Conference, SCN 2014*, volume 8642 of *Lecture Notes in Computer Science*, pages 135–154. Springer-Verlag, 2014.

[13] A. C. Yao. How to generate and exchange secrets. In *Proc. of the 27th IEEE Symp. on Foundations of Computer Science*, pages 162–167, 1986.

A An Alternative Proof of Claim 3.4

We start with a proving converse of Claim 3.4.

Claim A.1. *Fix $x \in X$ such that $p_x > 0$. If there exists a probability vector $\mathbf{x}_x^{(1)}$ solving Equation (7), then $(\mathbf{0}_k)^T$ is an affine combinations of the rows of M.*

Proof. Define

$$\mathbf{u} = \frac{(1-\alpha)p_x}{\alpha}\left(\mathbf{x}_x^{(1)} - \begin{pmatrix} 1/\ell \\ \vdots \\ 1/\ell \end{pmatrix}\right) + \mathbf{e_x}.$$

Now,

$$M^T\mathbf{u} \quad = \quad M^T\left(\frac{(1-\alpha)p_x}{\alpha}\mathbf{x}_x^{(1)} - \frac{(1-\alpha)p_x}{\alpha}\begin{pmatrix}1/\ell\\ \vdots \\ 1/\ell\end{pmatrix} - \mathbf{e_x}\right) \quad = \quad \mathbf{0}_k.$$

Furthermore, $\begin{pmatrix}1, \cdots, 1\end{pmatrix}\mathbf{u} = \frac{(1-\alpha)p_x}{\alpha}\begin{pmatrix}1, \cdots, 1\end{pmatrix}(\mathbf{x}_x^{(1)} - \begin{pmatrix}1/\ell\\ \vdots \\ 1/\ell\end{pmatrix}) + \begin{pmatrix}1, \cdots, 1\end{pmatrix}\mathbf{e_x} = 1$. Thus, $\mathbf{0}_k$ is an affine combination of the rows of M. □

We next provide an alternative Proof of Claim 3.4. This proof gives more intuition to how we got to the condition of the claim.

Proof. We consider the following change of variable

$$\widehat{\mathbf{x}}_x = \frac{(1-\alpha)p_x}{\alpha}\cdot((1/\ell, \ldots, 1/\ell)^T - \mathbf{x}_x^{(1)})$$

By orthogonality, we deduce that $(\mathsf{row}_x)^T$ belongs to the image of M and admits a suitable pre-image if

$$\forall \mathbf{w} \in \ker(M'), \qquad (\mathsf{row}_x)\mathbf{w} = 0,$$

which, by letting x vary yields

$$\forall \mathbf{w} \in \ker(M'), \qquad M\mathbf{w} = \mathbf{0}_k. \tag{22}$$

Finally,

$$\ker(M') = \left\{ \mathbf{w} \in \mathbb{R}^\ell \;\middle|\; M\mathbf{w} = \begin{pmatrix} \delta \\ \vdots \\ \delta \end{pmatrix} \right\}.$$

Thus, the condition in (22) is equivalent to

$$\text{There is no vector } w \in \mathbb{R}^k \text{ such that } M\mathbf{w} = \mathbf{1}_\ell. \tag{23}$$

□

We next prove that the conditions of Theorem 4.1 are necessary for our simulator to work. This implies that Protocol FairTwoPartyAsymm is a fully secure protocol for $f = (f_1, f_2)$ with our simulator if and only if $(\mathbf{0}_k)^T$ is an affine combinations of the rows of M_2 and each row of $M_1 * M_2$ is a linear combination of the rows of M_2.

Claim A.2. *If there exists probability vectors $(\mathbf{x}_x^{(a)})_{a\in\{0,1\},x\in X}$ solving Equations (12) and (15), then $(\mathbf{0}_k)^T$ is an affine combinations of the rows of M_2 and each row of $M_1 * M_2$ is a linear combination of the rows of M_2.*

Proof. Let $M_1 * M_2 = \begin{pmatrix} \mathsf{row}_1 \\ \vdots \\ \mathsf{row}_\ell \end{pmatrix}$. We first prove that each row of $M_1 * M_2$ is a linear combination of the rows of M_2. We only have to consider non-zero rows of $M_1 * M_2$; in this case we can rearrange Equation (15), and obtain the desired property,

$$(\mathsf{row}_x)^T = (M_1 * M_2)^T \mathbf{e_x} = \frac{(1-\alpha)p_x}{\alpha} M^T(\mathbf{x}_x^{(1)} - \big(1/\ell, \cdots, 1/\ell\big)^T). \tag{24}$$

Now take any $x \in X$ such that row_x is non-zero and let $\mathbf{z}$ be such that

$$(\mathsf{row}_x)^T = M_2^T \mathbf{z}.$$

We can rearrange Equation (15) and obtain the following equality.

$$M_2^T\left(\mathbf{x}_x^{(1)} - \big(1/\ell, \cdots, 1/\ell\big)^T + \frac{\alpha}{(1-\alpha)p_x}\mathbf{z}\right) = \mathbf{0}_k, \tag{25}$$

Thus, if $\sum_{i=1}^{\ell} z_i \neq 0$, then $(\mathbf{0}_k)^T$ is an affine combination of the rows of M_2. Otherwise, we can rearrange Equation (12) (remembering that $M_1 * M_2 + \overline{M_1} * M_2 = M_2$) and obtain the following equality.

$$M_2^T\left(\mathbf{x}_x^{(0)} - \big(1/\ell, \cdots, 1/\ell\big)^T + \frac{\alpha}{(1-\alpha)(1-p_x)}(\mathbf{e_x} - \mathbf{z})\right) = \mathbf{0}_k. \tag{26}$$

Since $\sum_{i=1}^{\ell} z_i = 0$, we have shown that $(\mathbf{0}_k)^T$ is an affine combination of the rows of M_2. □