

# Complete Characterization of Fairness in Secure Two-Party Computation of Boolean Functions<sup>\*</sup>

Gilad Asharov<sup>1</sup>, Amos Beimel<sup>2</sup>, Nikolaos Makriyannis<sup>3</sup>, and Eran Omri<sup>4</sup>

<sup>1</sup> The Hebrew University of Jerusalem, Jerusalem, Israel

<sup>2</sup> Ben Gurion University of the Negev, Be'er Sheva, Israel

<sup>3</sup> Universitat Pompeu Fabra, Barcelona, Spain

<sup>4</sup> Ariel University, Ariel, Israel

**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 Boolean 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.; 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; however, the outputs are not necessarily the same. We provide both a sufficient condition and 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:** Secure computation, fairness, foundations, malicious adversaries.

---

<sup>\*</sup> The first author is supported by the Israeli Centers of Research Excellence (I-CORE) Program (Center No. 4/11). The second author is partially supported by ISF grant 544/13 and by the Frankel Center for Computer Science. The fourth author is 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 a 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 [15] for the two-party setting and Goldreich, Micali, and Wigderson [10] 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 the 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 [8].

To elaborate further, Cleve [8] 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. [11] showed that this interpretation is inaccurate as there exist interesting functions that can 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. [11] 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 until recently.

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 [8] 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. [11] re-opened the question of characterizing fairness in secure two-party and multiparty computation. This question was studied in a sequence of works [2,4,14]. Asharov, Lindell, and Rabin [4] focused on the work of Cleve [8] and fully identified the 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 [14] 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 [2,14] 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 [14].

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 [12] 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 functions that can be computed with full security in the two-party setting. This solves an open problem of [11]. We focus on the third class of functions that was left unresolved by [2, 14], 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 Theorem 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 [14], we obtain a quite simple characterization for fairness:

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

We recall that an affine-combination is a linear-combination where the sum of coefficients is 1 (see Theorem 1.2).

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.

*Example 1.2.* The following function is a concrete example of a function that was left as unresolved in [2, 14].

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

We show that the GHKL protocol is susceptible to an attack for this particular function, however, we show that it can be computed using our modified protocol. In this example, the all-one vector is an affine combination of the rows (taking the first row with coefficient -1, the second and the fourth 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.

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 an  $\ell \times k$  real matrix. A vector  $\mathbf{w} \in \mathbb{R}^\ell$  is in the image of  $M$ , denoted  $\mathbf{w} \in \text{im}(M)$ , 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  $\mathbf{v} \in \text{ker}(M)$ , if  $M\mathbf{v} = \mathbf{0}_\ell$ .

The affine hull of a set  $\mathcal{V} \subseteq \mathbb{R}^\ell$ , denoted  $\text{affine-hull}(\mathcal{V})$ , 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,  $\text{affine-hull}(\mathcal{V}) = \{\sum_{\mathbf{v} \in \mathcal{V}} a_{\mathbf{v}} \cdot \mathbf{v} : \sum_{\mathbf{v} \in \mathcal{V}} a_{\mathbf{v}} = 1\}$ .

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

*Proof.*  $\mathbf{1}_\ell$  is not a linear combination of the columns of  $M$  iff  $\text{rank}((M|1)) = \text{rank}(M) + 1$  (where  $(M|1)$  is the matrix  $M$  with the extra all-one column) iff  $\dim(\text{ker}((M|1)^T)) = \dim(\text{ker}(M^T)) - 1$  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  $A$  and  $B$ , we say that  $C = A * B$  if  $C$  is the entry-wise (Hadamard) product of the matrices, that is,  $C_{i,j} = A_{i,j} \cdot B_{i,j}$ .*

*Secure Multiparty Computation.* We assume that the reader is familiar with the definitions of secure computation, and with the ideal-real paradigm. We refer to [7, 9] and to the full version of this paper [3] for formal definitions. We distinguish between security-with-abort, for which the adversary may receive outputs while the honest parties might not (security without fairness), and full security (security with fairness), where all parties receive outputs (this is similar to security with respect to honest majority as in [7, 9], although we do not have a honest majority).

### 3 A Fully-Secure Protocol for Boolean Two-Party Functions

In this section we present our protocol that securely computes any function for which the all-one or all-zero vector is an affine combination of either the rows or the columns of its associated matrix. In Section 3.1, we show that our protocol can compute functions that cannot be computed using the original GHKL protocol.

Let  $f : X \times Y \rightarrow \{0, 1\}$  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 1 we describe Protocol FAIRTWOPARTY $_{\sigma}$ , a modification of the GHKL protocol [11]. Protocol FAIRTWOPARTY $_{\sigma}$  is described using an on-line dealer.

*Remark 3.1.* We show that our protocol computes the function with full security in the presence of an on-line dealer, even though each party can abort this on-line dealer at any point of the execution. We remark that this implies full security in the plain model. In order to see this, we first remark that one can use direct transformation from the on-line dealer to the plain model, as discussed in [2]. An alternative transformation is the following, which is by now standard and is based on [11].

We next sketch the transformation. Protocol FAIRTWOPARTY $_{\sigma}$  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  $i + 1$ . one can transform this protocol to a protocol with an off-line dealer that is only invoked in the initial round of the protocol. This off-line dealer first chooses  $a_1, \dots, a_r$  and  $b_1, \dots, b_r$  in the same way the on-line dealer chooses them, computes authenticated 2-out-of-2 shares of  $a_1, \dots, a_r$  and  $b_1, \dots, b_r$ , and gives one share to  $P_1$  and the other to  $P_2$ . The protocol now proceeds in rounds, where 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  $b_i$ . Assuming that a secure protocol for OT exists, this protocol with an off-line dealer can be transformed into a protocol in the plain model where the parties compute this one-time dealer using security

with abort [9]. Alternatively, we can use Kilian’s protocol [13] that computes the off-line functionality in the OT-hybrid model providing information-theoretic security-with-abort.

**Protocol FAIRTWOPARTY $_{\sigma}$**

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

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

and

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

4. The dealer gives  $b_0$  to  $P_2$ .
5. For  $i = 1, \dots, r$ ,
  - (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.

---

<sup>a</sup> 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.

**Fig. 1.** Protocol FAIRTWOPARTY $_{\sigma}$  for securely computing a function  $f$ , where  $\sigma$  is either 0 or 1

The main difference in Protocol FAIRTWOPARTY $_{\sigma}$  compared to the GHKL protocol is in Step 3, where  $b_{i^*-1} = \sigma$  (compared to  $b_{i^*-1} = f(x, \tilde{y})$  for a random  $\tilde{y}$  in the GHKL protocol). For some functions  $f$  we choose  $\sigma = 0$  and for some we choose  $\sigma = 1$ ; the choice of  $\sigma$  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. See Section 3.1 for more intuition explaining why this change works.

*Remark 3.2.* There are two parameters in Protocol FAIRTWOPARTY $_{\sigma}$  that are unspecified – the parameter  $\alpha$  of the geometric distribution and the number of

rounds  $r$ . We show that there exists a constant  $\alpha_0$  (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 FAIRTWOPARTY<sub>0</sub> 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 [11]. 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 real world adversary  $\mathcal{A}$  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 2. It operates along the same lines as in the proof of [11]. 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.

**The simulator  $\mathcal{S}^{\mathcal{A}}$  for Protocol FAIRTWOPARTY<sub>0</sub>**

- The adversary  $\mathcal{A}$  gives its input  $x$  to the simulator.<sup>a</sup>
- The simulator chooses  $i^* \geq 2$  according to the geometric distribution with probability  $\alpha$ .
- For  $i = 1, \dots, i^* - 1$ :
  - The simulator gives  $a_i = f(x, \tilde{y}^{(i)})$  to the adversary  $\mathcal{A}$ , where  $\tilde{y}^{(i)}$  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  $a_1, \dots, a_i$ , and halts.
- At round  $i = i^*$ , the simulator gives  $x$  to the trusted party and gets the output  $a = f(x, y)$ .
- For  $i = i^*, \dots, r$ : The simulator gives  $a_i = a$  to the adversary  $\mathcal{A}$ , if  $\mathcal{A}$  aborts, then the simulator outputs the bits  $a_1, \dots, a_i$  and halts.
- The simulator outputs the bits  $a_1, \dots, a_r$  and halts.

<sup>a</sup> 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.

**Fig. 2.** The simulator  $\mathcal{S}^{\mathcal{A}}$  for Protocol FAIRTWOPARTY<sub>0</sub>

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 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 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 FAIRTWOPARTY<sub>0</sub>, 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  $a_i = f(x, \tilde{y})$  is chosen with a uniformly random  $\tilde{y}$ . 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 of 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 FAIRTWOPARTY<sub>0</sub> 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  $a_1, \dots, a_i$ . Notice that in both worlds  $a_1, \dots, a_{i-1}$  are equally distributed and are independent of  $(a_i, \text{Out}_{P_2})$ . Thus, we only compare the distribution of  $(a_i, \text{Out}_{P_2})$  in both worlds.

First, we introduce the following notation

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

For example,  $s_y$  is the probability that  $f(\tilde{x}, y) = 1$ , when  $\tilde{x}$  is uniformly distributed. Furthermore, for every  $x \in X$ ,  $y \in Y$ , and  $a \in \{0, 1\}$ , define  $q_x^{(a)}(y) \triangleq \Pr[f(x_0, y) = 1]$ , 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,

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

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  $(a_i, \text{Out}_{P_2})$ .

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

- $i < i^*$ ,  $a_i = f(x, \tilde{y}) = 0$ , and  $\text{Out}_{P_2} = b_{i-1} = f(\tilde{x}, y) = 0$ . The probability of this event is  $(1 - \alpha)(1 - p_x)(1 - s_y)$ .
- $i = i^*$ ,  $a_{i^*} = f(x, y) = 0$ , and  $\text{Out}_{P_2} = b_{i^*-1} = 0$ . Recall that always  $b_{i^*-1} = 0$  in Protocol FAIRTWOPARTY<sub>0</sub>. 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  $\alpha$  otherwise).

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

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

Therefore, in the ideal world  $\Pr[(a_i, \text{Out}_{P_2}) = (0, 0)] = (1 - \alpha)(1 - p_x)(1 - q_x^{(0)}(y)) + \alpha(1 - f(x, y))$ . 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

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

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

*Third Case:*  $(a_i, \text{Out}_{P_2}) = (1, 0)$ . In the real world  $\Pr[(a_i, \text{Out}_{P_2}) = (1, 0)] = (1 - \alpha)p_x(1 - s_y) + \alpha \cdot f(x, y) \cdot 1$ . On the other hand, in the ideal world we have that  $\Pr[(a_i, \text{Out}_{P_2}) = (1, 0)] = (1 - \alpha)p_x(1 - q_x^{(1)}(y))$ . 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

$$\begin{aligned} M^T \mathbf{x}_x^{(1)} &= \mathbf{q}_x^{(1)} = M^T (1/\ell, \dots, 1/\ell)^T - \frac{\alpha}{(1-\alpha)p_x} (\text{row}_x)^T \\ &= M^T \left( (1/\ell, \dots, 1/\ell)^T - \frac{\alpha}{(1-\alpha)p_x} \mathbf{e}_x \right), \end{aligned} \tag{7}$$

where  $\text{row}_x$  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 (6), we remark that if the equalities hold for the first 3 cases of the values for  $(a_i, \text{Out}_{P_2})$ , the equality of the probabilities must also hold for the case  $(a_i, \text{Out}_{P_2}) = (1, 1)$ . In the rest of the proof we show that there exist probability vectors  $\mathbf{x}_x^{(1)}$  for every  $x \in X$  solving Equation (6).

**Claim 3.4.** *Fix  $x \in X$  and let  $\alpha$  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 (6).*

*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  $(1, \dots, 1) \mathbf{u} = 1$ . Consider the vector  $\mathbf{y} = (1/\ell, \dots, 1/\ell)^T + \frac{\alpha}{(1-\alpha)p_x} (\mathbf{u} - \mathbf{e}_x)$ . The vector  $\mathbf{y}$  is a solution to Equation (6) since  $M^T \mathbf{u} = \mathbf{0}_k$ . We need to show that it is a probability vector. First,

$$\begin{aligned} (1, \dots, 1) \mathbf{y} &= (1, \dots, 1) (1/\ell, \dots, 1/\ell)^T + (1, \dots, 1) \frac{\alpha}{(1-\alpha)p_x} (\mathbf{u} - \mathbf{e}_x) \\ &= 1 + \frac{\alpha}{(1-\alpha)p_x} (1 - 1) = 1. \end{aligned}$$

Second,  $y_i \geq 1/\ell - \frac{\alpha}{(1-\alpha)p_x} (1 + |u_i|) \geq 1/\ell - 2k\alpha(1 + \max\{|u_j|\})$  (recall that  $p_x \geq 1/k$  as  $p_x > 0$  and  $p_x$  is a multiple of  $1/k$ ). Thus, by taking  $\alpha \leq 1/(2k\ell(1 + \max\{|u_j|\}))$ , the vector  $\mathbf{y}$  is non-negative, and, therefore, it is a probability vector solving Equation (6).<sup>1</sup>  $\square$

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 (6). Furthermore, these probability vectors can be efficiently found. Using these probability vectors as distributions in the simulator we constructed proves that the protocol FAIRTWOPARTY<sub>0</sub> is fully secure for  $f$ .  $\square$

We provide an alternative proof of Theorem 3.4 in the full version of this paper. Furthermore, in the full version we prove the converse of Theorem 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 FAIRTWOPARTY<sub>0</sub>.

---

<sup>1</sup> As the size of the domain of  $f$  is considered as a constant, the vector  $\mathbf{u}$  is a fixed vector and  $\alpha$  is a constant.

*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 FAIRTWOPARTY<sub>1</sub>), 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 FAIRTWOPARTY<sub>1</sub> with  $\alpha \leq \alpha_0$  is a fully-secure protocol for  $f$ .*

*Proof.* Let  $\overline{M} = (1) - M$ . Executing Protocol FAIRTWOPARTY<sub>1</sub> on  $f$  is equivalent to executing Protocol FAIRTWOPARTY<sub>0</sub> with the function  $\overline{f}(x, y) = 1 - f(x, y)$  (whose associated matrix is  $\overline{M}$ ), and flipping the output. Thus, Protocol FAIRTWOPARTY<sub>1</sub> 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  $\overline{M}$  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 FAIRTWOPARTY<sub>0</sub> is a fully secure protocol for  $\overline{f}$ .  $\square$

**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 FAIRTWOPARTY<sub>0</sub> if  $(\mathbf{0}_k)^T$  is an affine combination of the rows of  $M$ . Likewise, by Theorem 3.5, a function can be computed with protocol FAIRTWOPARTY<sub>1</sub> 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 FAIRTWOPARTY <sub>$\sigma$</sub>  to a protocol without a dealer providing full security.  $\square$

### 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 influence 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 if the 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. If  $a_1 = 0$ , 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:

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

*Protocol FAIRTWO PARTY<sub>1</sub>*. Since  $\mathbf{1}_k^T$  is an affine combination of the rows of  $M$ , we use the protocol FAIRTWO PARTY<sub>1</sub>, 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.

We next give some intuition why the GHKL protocol is susceptible to this attack whereas our protocol is immune to it. 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.

## 4 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. To prove this characterization, recall the definition of *semi-balanced* functions given in [14]:

**Definition 4.1.** *A function  $f : X \times Y \rightarrow \{0, 1\}$  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-balanced.*

Makriyannis [14] 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 FAIRTWOPARTY $_\sigma$ .

**Lemma 4.2.** *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 Theorem 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  $M$ .  $\square$

**Theorem 4.3.** *Assume 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 [14] it cannot be computed with complete fairness, hence, it cannot be computed with full security.

If  $f$  is not semi-balanced, then, by Lemma 4.2, 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.  $\square$

### 4.1 Extensions

First we consider the OT-hybrid model. As explained in Theorem 3.1, our protocol can be executed in the OT-hybrid model providing information-theoretic security (without any dealer). Furthermore, the impossibility result of [1] holds in the OT-hybrid model and the reduction of [14] is information-theoretic secure. Thus, our characterization remains valid in the OT-hybrid model.

**Corollary 4.4.** *Let  $f$  be a Boolean two-party function  $f$  with an associated matrix  $M$ . The function  $f$  can be computed with full-security in the OT-hybrid model with information-theoretic 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  $M$ .

Second, consider randomized functionalities. Let  $f : X \times Y \rightarrow \Delta(\{0, 1\})$  be a randomized finite Boolean function and define associated  $\ell \times k$  matrix  $M$  such that for all  $i, j$ ,  $M_{i,j} = \Pr[f(x_i, y_j) = 1]$ . One can modify Protocol 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 4.5.** *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.*

### 4.2 A Geometric Interpretation of the Characterization

We review the geometric representation of our characterization, which may shed some light 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  $\mathbf{x} = (x_1, \dots, x_m) \in \mathbb{R}^m$  that are a solution of some linear equation  $a_1x_1 + \dots + a_mx_m = b$ , 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 4.6.** *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.  $\mathbf{0}_k, \mathbf{1}_k \notin \text{affine-hull}\{X_1, \dots, X_\ell\}$  and  $\mathbf{0}_\ell, \mathbf{1}_\ell \notin \text{affine-hull}\{Y_1, \dots, Y_k\}$ .
- 3. *There exists an hyperplane  $\mathcal{H}(\mathbf{q}, 1)$  that contains all the rows  $X_1, \dots, X_\ell$ , and there exists yet another hyperplane  $\mathcal{H}(\mathbf{p}, 1)$  that contains all the columns  $Y_1, \dots, Y_k$ . In addition,  $\mathbf{1}_k, \mathbf{0}_k \notin \mathcal{H}(\mathbf{q}, 1)$  and  $\mathbf{1}_\ell, \mathbf{0}_\ell \notin \mathcal{H}(\mathbf{p}, 1)$ .*

The proof can be found in the full version of this paper [3]. The following theorem divides the Boolean functions to three categories. Each category has some different properties which we will discuss in the following.

**Theorem 4.7.** *Let  $f$  be a Boolean function. Then:*

**1. The function is balanced:**

*There exists an hyperplane  $\mathcal{H}(\mathbf{q}, \delta_1)$  that contains all the rows  $X_1, \dots, X_\ell$ , and there exists yet another hyperplane  $\mathcal{H}(\mathbf{p}, \delta_2)$  that contains all the columns  $Y_1, \dots, Y_k$ . Then:*

- (a) *If  $\mathbf{0}_\ell, \mathbf{1}_\ell \notin \mathcal{H}(\mathbf{p}, \delta_1)$  and  $\mathbf{0}_k, \mathbf{1}_k \notin \mathcal{H}(\mathbf{q}, \delta_2)$ , then the function is semi-balanced and cannot be computed fairly.*
- (b) *If either  $\mathcal{H}(\mathbf{p}, \delta_1)$  or  $\mathcal{H}(\mathbf{q}, \delta_2)$  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 not lie on a single hyperplane, then the function is unbalanced and can be computed with full-security.*

We remark that case 1b was left unresolved in [2, 14], and is finally proven to be possible here.

*Intuition.* Consider a single (ideal, fair) execution of some function  $f$  where  $P_1$  chooses its input according to some distribution  $\mathbf{a} = (a_1, \dots, a_\ell)$ , i.e., chooses input  $x_i$  with probability  $a_i$ , and  $P_2$  chooses its input according to distribution  $\mathbf{b} = (b_1, \dots, b_k)$ . Then, the parties invoke the function and receive the same output simultaneously. The vector  $\alpha^T \cdot M_f = (w_1, \dots, w_k)$  is the output vector of  $P_2$ , where  $w_j$  represents the probability that the output of  $P_2$  is 1 when it uses input  $y_j$  in this execution. Similarly, we consider also the output vector  $M_f \cdot \mathbf{b}$ .

Intuitively, fairness is achievable in functions which are more “unbalanced” and is impossible to achieve in more “balanced” functions. In our context, the term “balanced” relates to the question of whether there exists a party that can influence and bias the output of the other party in a single (and fair) invocation of the function, as mentioned above.

More concretely, for balanced functions (class 1a), we use the fact that in a single execution no party can bias the output of the other in the reduction to sampling [14]. Specifically, the reduction works by considering a single (fair) execution of the function, where each party chooses its input randomly according to some cleverly chosen distributions  $\mathbf{a}'$  and  $\mathbf{b}'$ . The geometric representation of the functions in this class may explain why these functions are reducible to sampling. The fact that all the rows lie on a single hyperplane  $\mathcal{H}(\mathbf{q}, \delta_1)$  implies that all their convex combinations also lie on that hyperplane. Therefore, no matter what input distribution  $\mathbf{a}'$  a malicious  $P_1$  may choose (i.e., what convex combination of the rows), the resulting vector  $\mathbf{a}' \cdot M_f = (w_1, \dots, w_k)$  lies on the hyperplane  $\mathcal{H}(\mathbf{q}, \delta_1)$  as well, and therefore satisfies the relation  $\langle \mathbf{a}', \mathbf{q} \rangle = \delta_1$ . This guarantees that there is the exact *same* correlation between the possible outputs of  $P_2$  no matter what input  $P_1$  may use, and this enables  $P_2$  to deduce a coin



from this function. We also have a similar guarantee for the output of  $P_1$  and malicious  $P_2$ , and thus this function is reducible to sampling.

The other extreme case is class 2, the class of unbalanced functions. In this class of functions, one party has significant more power over the other, and can manipulate its output. In this case, the single invocation of the function that we consider is the process of the ideal execution: the honest party and the simulator send their inputs to the trusted party, who computes the function correctly and sends the outputs back simultaneously. What enables successful simulation is the fact that the simulator can actually manipulate the output of the honest party. In particular, in the proof of Theorem 3.3 the simulator chooses the input distribution  $\mathbf{x}_x^{(b)}$  cleverly in order to succeed in the simulation. Geometrically, since the inputs of the honest party do not lie on any hyperplane, its outputs are not correlated and the simulator has enough freedom to manipulate them.

Additional interesting geometric properties that show the differences between the two class of functions (classes 1a and 2) are the following. First, one can show that for each function in class 1a, the two hyperplanes that contain the rows and resp. the columns are *unique*. This implies that the *affine dimensions* of the affine hulls of the rows and the affine-hull of the columns are equal. On the other hand, in class 2, the affine dimensions of these two affine-hulls are always distinct. Moreover, almost all functions that satisfy  $|X| = |Y|$  are in class 1a, whereas almost all functions that satisfy  $|X| \neq |Y|$  are in class 2.

*Class 1b.* The third class of functions is where the things become less clearer, and may even contradict the intuition mentioned above. This class contains functions that are totally symmetric and satisfy  $M_f^T = M_f$  (see, for instance, Example 1.2), and thus both parties have the exact same influence on the output of the other party in a single invocation of the function, the affine dimensions of the rows and the columns are the same, and also in most cases  $|X| = |Y|$ . Yet, somewhat surprisingly, fairness is possible. Overall, since all the rows and columns lie on hyperplanes, all the functions in this class are reducible to sampling; however, the sampling is a trivial one (where the resulting coins of the parties are uncorrelated). In addition, although the influence 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. This case is much more delicate, and the GHKL protocol fails to work for some functions in this class. Nevertheless, as we show, the limited power that the simulator has in the ideal execution suffices.

## 5 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  $f_i : X \times Y \rightarrow \{0, 1\}$  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 (the 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.

### 5.1 Sufficient Condition for Fairness of Two-Party Asymmetric Functionalities

We analyze when Protocol  $\text{FAIRTWOPARTY}_\sigma$ , described in Figure 1, provides full security for Boolean asymmetric functionalities  $f = (f_1, f_2)$ . We modify Protocol  $\text{FAIRTWOPARTY}_\sigma$  such that the backup values of  $P_i$  are computed with  $f_i$ ; we call the resulting protocol  $\text{FAIRTWOPARTYASYMM}_\sigma$ . For the next theorem recall that  $M_1 * M_2$  is the entry-wise product of the matrices.

**Theorem 5.1.** *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  $\text{FAIRTWOPARTYASYMM}_0$  with  $\alpha \leq \alpha_0$  is a fully-secure protocol for  $(f_1, f_2)$ .*

In Theorem 5.1, 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 5.1).* We only discuss the required changes in the proof of Protocol  $\text{FAIRTWOPARTYASYMM}_0$  compared to the proof of Protocol  $\text{FAIRTWOPARTY}_0$ . 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  $(a_i, \text{Out}_{P_2})$  in both worlds given that the adversary aborts in round  $i \leq i^*$ .

First, we introduce the following notation

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

For example,  $s_y$  is the probability that  $f_2(\tilde{x}, y) = 1$ , when  $\tilde{x}$  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  $(a_i, \text{Out}_{P_2})$ .

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

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

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

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

**Claim 5.2.** *Fix  $x \in X$  and let  $\alpha$  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  $\mathbf{1}$  denote the  $\ell \times k$  all-one matrix, and let  $\lambda_\alpha = \frac{\alpha}{(1 - \alpha)(1 - p_x)}$ . We get that  $(\overline{M_1} * M_2)^T \mathbf{e}_x = ((\mathbf{1} - M_1) * M_2)^T \mathbf{e}_x = (M_2 - M_1 * M_2)^T \mathbf{e}_x = M_2^T \mathbf{e}_x - (M_1 * M_2)^T \mathbf{e}_x$ . 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  $M_2^T \cdot \mathbf{v}$  for some vector  $\mathbf{v} \in \mathbb{R}^\ell$ . We therefore are looking for a probability vector  $\mathbf{x}_x^{(0)}$  solving:

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

Let  $\beta = \langle \mathbf{1}, \mathbf{v} \rangle$ , and recall that  $\mathbf{0}_k$  is an affine combination 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  $\mathbf{y} = (1/\ell, \dots, 1/\ell)^T - \lambda_\alpha \mathbf{v} + \lambda_\alpha \beta \mathbf{u}$ . 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:

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

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

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

*Third Case:*  $(a_i, \text{Out}_{P_2}) = (1, 0)$ . In the real world  $\Pr[(a_i, \text{Out}_{P_2}) = (1, 0)] = (1 - \alpha)p_x(1 - s_y) + \alpha \cdot f_1(x, y) \cdot 1$ . On the other hand, in the ideal world  $\Pr[(a_i, \text{Out}_{P_2}) = (1, 0)] = (1 - \alpha)p_x(1 - q_x^{(1)}(y)) + \alpha \cdot f_1(x, y)(1 - f_2(x, y))$ . 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 deduce, using Equations (9) and (8), that

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

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  $(a_i, \text{Out}_{P_2})$ , the equality of the probabilities must also hold for the case  $(a_i, \text{Out}_{P_2}) = (1, 1)$ .

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<sub>0</sub> is fully secure for  $f = (f_1, f_2)$ .  $\square$

*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  $\overline{M} = (\mathbf{1}) - M$ ) 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<sub>0</sub>

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 5.3.** *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 there is a constant  $\alpha_0 > 0$  such that the Protocol FAIRTWOPARTYASYMM' with  $\alpha \leq \alpha_0$  is a fully-secure protocol for  $f = (f_1, f_2)$ .*

## 5.2 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 5.4.** *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  $\Pi$  in the hybrid model with ideal access to  $f$  (with full security). This protocol achieves non-trivial sampling.

- **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 5.5.** *Let  $\text{Out}_1, \text{Out}_2$  for the outputs of  $P_1$  and  $P_2$  in the above protocol. In an honest execution of  $\Pi$ , the parties’ outputs satisfy  $\Pr[\text{Out}_1 = 1] = \delta_1 + p^-$ ,  $\Pr[\text{Out}_2 = 1] = \delta_2 + q^-$  and  $\Pr[\text{Out}_1 = 1 \wedge \text{Out}_2 = 1] = \delta_{1,2} + p^- \delta_2 + q^- \delta_1 + p^- q^-$ , 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  $\text{row}_{1,i}$ ,  $\text{row}_{2,i}$  denote the  $i$ -th row of  $M_1$  and  $M_2$  respectively, and let  $\text{col}_{1,j}$ ,  $\text{col}_{2,j}$  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  $\text{row}_{1,i}$  if  $p_i \geq 0$  and  $\mathbf{1}_k^T - \text{row}_{1,i}$  otherwise,
- the  $j$ -th column of  $\hat{M}_2$  is equal to  $\text{col}_{2,j}$  if  $q_j \geq 0$  and  $\mathbf{1}_\ell - \text{col}_{2,j}$  otherwise.

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

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

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

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

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

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

□

It remains to show that protocol  $\Pi$  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  $\Pr[\text{Out}_1 = 1] \cdot \Pr[\text{Out}_2 = 1] = \Pr[\text{Out}_1 = 1 \wedge \text{Out}_2 = 1]$ . 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 does not send one at all), by (17), the probability that the output of  $P_1$  is 1 is equal to  $\delta_1 + p^-$ . Hence, by [1], we get a contradiction. □

### 5.3 Special Round Protocol with a Twist

Define an asymmetric functionality  $f_{\text{sp}}(x, y) = (f_1(x, y), f_2(x, y))$ , 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  $f_{\text{sp}}$  does not satisfy the hypothesis of Theorem 5.4. 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.

**On the limits of the known protocols for the function  $f_{\text{sp}}$ .** Suppose that the parties execute protocol FAIRTWOPARTYASYMM $_0$  for computing  $f_{\text{sp}}$ . In addition, suppose that  $P_2$  chooses 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 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  $y \in_U \{y_1, y_2\}$ :

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

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  $b_{i^*-1} = f_2(\tilde{x}^{(i^*-1)}, y)$  where  $\tilde{x}^{(i^*-1)}$  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  $y \in_U \{y_1, y_2\}$ :

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

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 FAIRTWOPARTYSPECIAL (described in Figure 3), which foils the attacks described above and provides full security for  $f_{\text{sp}}$ . Our protocol is the same as 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  $f_{\text{sp}}$ : if  $x \in \{x_1, x_2\}$ , then  $a_{i^*} = f_1(x, y)$ , otherwise  $a_{i^*} = f_1(x, \tilde{y}^{(i^*)})$ , where  $\tilde{y}^{(i^*)}$  is chosen uniformly at random. However, always  $b_{i^*} = f_2(x, y)$ . Thus, for certain inputs, the second player  $P_2$  effectively obtains the output first. We claim that this modification suffices to compute  $f_{\text{sp}}$  with full security. and we dedicate the rest of the section to the proof this claim.

**Theorem 5.6.** *For every  $\alpha \leq 1/9$ , protocol FAIRTWOPARTYSPECIAL is fully secure for  $f_{\text{sp}}$ .*

*Proof.* The proof follows the ideas of the symmetric case with two significant modifications. First, we need two distinct simulations for  $P_1$  depending on the choice of the input. 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 the equivalent asymmetric protocol FAIRTWOPARTYASYMM<sub>0</sub>). 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<sub>0</sub>), 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 proof of Protocol FAIRTWOPARTYASYMM<sub>0</sub>. As in the proofs of Theorem 3.3 and Theorem 5.1, we only need to compare the distribution of  $(a_i, \text{Out}_{P_2})$  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 5.1, 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



**Protocol FAIRTWOPARTYSPECIAL**

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

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

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

4. The dealer gives  $b_0$  to  $P_2$ .
5. For  $i = 1, \dots, r$ ,
  - (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.

---

<sup>a</sup> 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.

**Fig. 3.** Protocol FAIRTWOPARTYSPECIAL for securely computing a function  $f_{sp}$

can choose any input from the entire domain to send to the trusted party. By considering the four possible values of  $(a_i, \text{Out}_{P_2})$ , 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,

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

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

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

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

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  $\alpha$  and,

- for  $i = 0, \dots, i^* - 1$ , the simulator hands  $b_i = f(\tilde{x}^{(i)}, y)$  to  $\mathcal{A}$ , where  $\tilde{x}^{(i)} \in U_X$ . 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  $(b_0, \dots, b_i)$  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  $(b_0, \dots, b_{i^*})$  and halts.
- for  $i = i^* + 1, \dots, r$ , the simulator hands  $b_i = b$  to  $\mathcal{A}$ . If  $\mathcal{A}$  decides to quit,  $\mathcal{S}^{\mathcal{A}}$  outputs  $(b_0, \dots, b_i)$  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 distribution of  $(b_i, \text{Out}_{P_1})$  in both worlds assuming that  $i \leq i^*$ . 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  $(b_i, \text{Out}_{P_1})$ . 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:*  $(b_i, \text{Out}_{P_1}) = (0, 0)$ . In the real world, if  $x \in \{x_1, x_2\}$ , then we have  $\Pr[(b_i, \text{Out}_{P_1}) = (0, 0)] = (1 - \alpha)(1 - p_x)(1 - s_y) + \alpha(1 - f_1(x, y))(1 - f_2(x, y))$ . Otherwise,  $\Pr[(b_i, \text{Out}_{P_1}) = (0, 0)] = (1 - \alpha)(1 - p_x)(1 - s_y) + \alpha(1 - p_x)$

$(1 - f_2(x, y))$ . On the other hand, in the ideal world  $\Pr[(b_i, \text{Out}_{P_1}) = (0, 0)] = (1 - \alpha)(1 - s_y)(1 - q_y^{(0)}(x)) + \alpha(1 - f_1(x, y))(1 - f_2(x, y))$ . To get full security we need that these two probabilities in the two worlds are the same, that is,

$$(1 - \alpha)(1 - s_y)(1 - q_y^{(0)}(x)) + \alpha(1 - f_1(x, y))(1 - f_2(x, y)) \tag{19}$$

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

Eq. (19) is equivalent to

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

Let  $\lambda = \alpha/(1 - \alpha)$ ; we now compute the vectors  $\mathbf{q}_y^{(0)}$ :

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

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

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

$$\mathbf{y}_{y_1}^{(0)} = 1/4 \cdot (1, 1, 1, 1)^T + \frac{\lambda}{1 - s_{y_1}} 1/2 \cdot (0, 1, -1, 0)^T$$

$$\mathbf{y}_{y_2}^{(0)} = 1/4 \cdot (1, 1, 1, 1)^T + \frac{\lambda}{1 - s_{y_2}} 1/2 \cdot (1, 0, -1, 0)^T$$

$$\mathbf{y}_{y_3}^{(0)} = 1/4 \cdot (1, 1, 1, 1)^T + \frac{\lambda}{1 - s_{y_3}} 1/2 \cdot (1, 0, -1, 0)^T$$

$$\mathbf{y}_{y_4}^{(0)} = 1/4 \cdot (1, 1, 1, 1)^T + \frac{\lambda}{1 - s_{y_4}} 1/2 \cdot (-1, 0, 1, 0)^T .$$

Note that, to obtain probability vectors, any  $\alpha \leq 1/9$  will do.

We refer to the full proof in [3] for cases  $(b_i, \text{Out}_{P_1}) = (0, 1)$  and  $(b_i, \text{Out}_{P_1}) = (1, 0)$ . □

## 6 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 4, we fully characterize

the 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 stated in Theorem 6.2 below.

Let  $X = X_1 \times X_2 \times \dots \times X_m$  and let  $f : X \rightarrow R$  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  $I$ . For an input  $\mathbf{x} \in X$ , let  $\mathbf{x}_I$  be the projection of  $\mathbf{x}$  on  $I$ . We define the function  $f_I : X_I \times X_{\bar{I}} \rightarrow R$ , by  $f_I(\mathbf{w}, \mathbf{z}) = f(\mathbf{x})$ , where  $\mathbf{x}$  is such that  $\mathbf{x}_I = \mathbf{w}$  and  $\mathbf{x}_{\bar{I}} = \mathbf{z}$ .

The full proof of the theorem appears in the full version of this work (see, Section 5 in [3]). The proof of the feasibility part of Theorem 6.2 is proved by describing 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  $\Pi_I$  that securely computes the two-party function  $f_I$ . The description of Protocol FAIRMULTIPARTY, proving the feasibility, appears in Figure 4. The proof of its security as well as the explanation of how the on-line dealer is eliminated are deferred to the full version of this paper. Our proofs draw on ideas from [5, 6].

**Protocol FAIRMULTIPARTY**

1. The parties  $P_1, \dots, P_m$  hand their inputs, denoted  $\mathbf{x} = x_1, \dots, x_m$ , 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  $f(x_1, \dots, x_m)$  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  $\Pi_I$ .
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  $i = 1, \dots, r$ ,
  - (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).

**Fig. 4.** Protocol FAIRMULTIPARTY for securely computing a function  $f$ , if at least half of the parties are honest

We next introduce some of the notation that is necessary for understanding the description of Protocol FAIRMULTIPARTY described in Figure 4. We assume without loss of generality that there exists an integer  $r$  such that each  $\Pi_I$  is an

$r$ -round protocol with an on-line dealer for securely computing the two-party function  $f_I$ . We show in the full version that this is without loss of generality.

**Notation 6.1.** Fix  $I \subset [m]$ , such that  $1 \in I$  and  $|I| = k$ . Let  $A_I$  be the party that plays the role of  $A$  in  $\Pi_I$ , and let  $B_I$  be the other party in this protocol. Denote by  $a_i^{(I)}$  (resp.  $b_i^{(I)}$ ) the backup output that party  $A_I$  (resp.  $B_I$ ) receives in round  $i$  of  $\Pi_I$ . In addition, let  $S_I^{(1)}$  be the set of parties whose inputs correspond to the input of  $A_I$ , that is,  $S_I^{(1)} = \{P_i : i \in I\}$ ; let  $S_I^{(2)}$  be the remaining parties (i.e., whose inputs correspond to that of party  $B$ ).

**Theorem 6.2.** Let  $m = 2k$  be a constant, let  $X = X_1 \times X_2 \times \dots \times X_m$  be a finite domain, and let  $f : X \rightarrow R$  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.

## 7 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<sup>2</sup> of [14], 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.

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

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

---

<sup>2</sup> The construction is also presented in the proof of Theorem 5.4 in Section 5.2.

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. Agrawal, S., Prabhakaran, M.: On fair exchange, fair coins and fair sampling. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 259–276. Springer, Heidelberg (2013)
2. Asharov, G.: Towards characterizing complete fairness in secure two-party computation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 291–316. Springer, Heidelberg (2014)
3. Asharov, G., Beimel, A., Makriyannis, N., Omri, E.: Complete characterization of fairness in secure two-party computation of boolean functions. Cryptology ePrint Archive, Report 2014/1000 (2014), <http://eprint.iacr.org/>
4. Asharov, G., Lindell, Y., Rabin, T.: A full characterization of functions that imply fair coin tossing and ramifications to fairness. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 243–262. Springer, Heidelberg (2013)
5. Beimel, A., Lindell, Y., Omri, E., Orlov, I.:  $1/p$ -secure multiparty computation without honest majority and the best of both worlds. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 277–296. Springer, Heidelberg (2011)
6. Beimel, A., Omri, E., Orlov, I.: Protocols for multiparty coin toss with dishonest majority. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 538–557. Springer, Heidelberg (2010)
7. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. of Cryptology 13(1), 143–202 (2000)
8. Cleve, R.: Limits on the security of coin flips when half the processors are faulty. In: 18th STOC, pp. 364–369 (1986)
9. Goldreich, O.: Foundations of Cryptography, Volume II Basic Applications. Cambridge University Press (2004)
10. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: 19th STOC, pp. 218–229 (1987)
11. Gordon, S.D., Hazay, C., Katz, J., Lindell, Y.: Complete fairness in secure two-party computation. J. of the ACM 58(6), Article No. 24 (2011)
12. Gordon, S.D., Katz, J.: Complete fairness in multi-party computation without an honest majority. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 19–35. Springer, Heidelberg (2009)
13. Kilian, J.: Basing cryptography on oblivious transfer. In: 20th STOC, pp. 20–31 (1988)
14. Makriyannis, N.: On the classification of finite boolean functions up to fairness. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 135–154. Springer, Heidelberg (2014)
15. Yao, A.C.: How to generate and exchange secrets. In: 27th FOCS, pp. 162–167 (1986)