Towards Characterizing Complete Fairness in Secure Two-Party Computation[*]

Gilad Asharov

Department of Computer Science
Bar-Ilan University, Israel
asharog@cs.biu.ac.il

February 10, 2014

Abstract

The well known impossibility result of Cleve (STOC 1986) implies that in general it is impossible to securely compute a function with *complete fairness* without an honest majority. Since then, the accepted belief has been that *nothing* non-trivial can be computed with complete fairness in the two party setting. The surprising work of Gordon, Hazay, Katz and Lindell (STOC 2008) shows that this belief is false, and that there exist *some* non-trivial (deterministic, finite-domain) boolean functions that can be computed fairly. This raises the fundamental question of characterizing complete fairness in secure two-party computation.

In this work we show that not only that some or few functions can be computed fairly, but rather an *enormous number* of functions can be computed fairly. In fact, *almost all* boolean functions with distinct domain sizes can be computed with complete fairness (for instance, more than 99.999% of the boolean functions with domain sizes 31×30). The class of functions that is shown to be possible includes also rather involved and highly non-trivial tasks, such as set-membership, evaluation of a private (boolean) function, private matchmaking and set-disjointness.

In addition, we demonstrate that fairness is not restricted to the class of symmetric boolean functions where both parties get the same output, which is the only known feasibility result. Specifically, we show that fairness is also possible for asymmetric boolean functions where the output of the parties is not necessarily the same. Moreover, we consider the class of functions with *non-binary* output, and show that fairness is possible *for any finite range.*

The constructions are based on the protocol of Gordon et. al, and its analysis uses tools from convex geometry.

Keywords: Complete fairness, secure two-party computation, foundations, malicious adversaries.

[*]This research was supported by the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 239868 and by the israel science foundation (grant No. 189/11).

Contents

1 Introduction

In the setting of secure multiparty computation, some mutually distrusting parties wish to compute some function of their inputs in the presence of adversarial behavior. The security requirements of such a computation are that nothing is learned from the protocol other than the output (privacy), that the outputs are distributed according to the prescribed functionality (correctness) and that the parties cannot choose their inputs as a function of the others' inputs (independence of inputs). Another important security property is that of *fairness*, which intuitively means that the adversary learns the output if and only if, the honest parties learn their output.

In the multiparty case, where a majority of the parties are honest, it is possible to compute any functionality while guaranteeing all the security properties mentioned above [GMW87, BGW88, CCD88, RB89, Gol04]. In the multiparty case when honest majority is not guaranteed, including the important case of the two-party settings where one may be corrupted, it is possible to compute any function while satisfying *all* security properties mentioned above *except* for fairness [Yao86, GMW87, Gol04]. The deficiency of fairness is not just an imperfection of theses constructions, but rather a result of inherent limitation. The well-known impossibility result of Cleve [Cle86] shows that there exist functions that cannot be computed by two parties with complete fairness, and thus, fairness cannot be achieved *in general*. Specifically, Cleve showed that the coin-tossing functionality, where two parties toss an unbiased fair coin, cannot be computed with complete fairness. This implies that any function that can be used to toss a fair coin (like, for instance, the boolean XOR function) cannot be computed fairly as well.

Since Cleve's result, the accepted belief has been that *only trivial functions*[1] can be computed with complete fairness. This belief is based on a solid and substantiate intuition: In any protocol computing any interesting function, the parties move from a state of no knowledge about the output to full knowledge about it. Protocols proceed in rounds and the parties cannot exchange information simultaneously, therefore, apparently, there must be a point in the execution where one party knows more about the output than the other party. Aborting at that round yields the unfair situation where one party can guess better the output, and learn the output alone.

Our understanding regarding fairness has been changed recently by the surprising work of Gordon, Hazay, Katz and Lindell [GHKL08]. This work shows that there *exist* some non-trivial (deterministic, finite-domain) boolean functions that can be computed in the malicious settings with *complete fairness*, and re-opens the research on this subject. The fact that *some* functions can be computed fairly, while some other were proven to be impossible to compute fairly, raises the following fundamental question:

Which functions can be computed with complete fairness?

Recently, [ALR13] provided a full characterization for the class of functions that imply fair coin-tossing and thus are ruled out by Cleve's impossibility. This extends our knowledge on what functions cannot be computed with complete fairness. However, there have been no other works that further our understanding regarding which (boolean) functions can be computed fairly, and the class of functions for which [GHKL08] shows possibility are the only known possible functions. There is therefore a large class of functions for which we have no idea as to whether or not they can be securely computed with complete fairness.

[1]In our context, the term "trivial functions" refers to constant functions, functions that depend on only one party's input and functions where only one party receives output. It is easy to see that these functions can be computed fairly.

To elaborate further, the work of [GHKL08] show that any function that does not contain an embedded XOR (i.e., inputs x_1, x_2, y_1, y_2 such that $f(x_1, y_1) = f(x_2, y_2) \neq f(x_1, y_2) = f(x_2, y_1)$) can be computed fairly. Examples of functions without an embedded XOR include the boolean OR / AND functions and the greater-than function. Given the fact that Cleve's impossibility result rules out completely fair computation of boolean XOR, a natural conjuncture is that any function that does contain an embedded XOR is impossible to compute fairly. However, the work shows that this conclusion is incorrect. Namely, it considers a *specific* function that does contain an embedded XOR, and constructs a protocol that securely computes this function with complete fairness.

In a more detail, [GHKL08] considered a specific simple (but non-trivial) 3×2 function, and showed that this function can be computed fairly by constructing a protocol for this function and a simulator for this protocol. Then, [GHKL08] showed that this protocol can be generalized, by identifying certain values and constants that can be parameterized, and parameterized the simulator also in a similar way. In addition, [GHKL08] showed how to examine whether the (parameterized) simulator succeed to simulate the protocol, by constructing a set of equations, where its solution is in fact, the actual parameters that the simulator has to use. The set of equations that [GHKL08] presented is rather involved, relies heavily on the actual parameters of the real protocol rather than properties of the computed function, and in particular it is hard to decide for a given function whether it can be computed fairly using this protocol, or not.

The protocol of [GHKL08] is ground-breaking and completely changed our perception regarding fairness. The fact that *something* non-trivial can be computed fairly is surprising, and raises many interesting questions. For instance, are there many functions that can be computed fairly, or only a few? Which functions can be computed fairly? Which functions can be computed using this generalized GHKL protocol? What property distinguishes these functions from the functions that are impossible to compute fairly? Furthermore, the protocol of GHKL is especially designed for deterministic symmetric boolean functions with finite domain, where both parties receive the same output. Is fairness possible in any other class of functions, over larger ranges, or for asymmetric functions? Overall, our understanding of what can be computed fairly is very vague.

1.1 Our Work

In this paper, we study the fundamental question of characterizing which functions can be computed with complete fairness. We show that *any* function that defines a full-dimensional geometric object, *can be computed with complete fairness*. That is, we present a simple property on the truth table of the function, and show that every function that satisfies this property, the function can be computed fairly. This extends our knowledge of what can be computed fairly, and is an important step towards a full characterization for fairness.

Our results deepen our understanding of fairness and show that many more functions can be computed fairly than what has been thought previously. Using results of combinatorics, we show that a random function with distinct domain sizes (i.e., functions $f : X \times Y \to \{0, 1\}$ where $|X| \neq |Y|$) defines a full-dimensional geometric object with overwhelming probability. Therefore, surprisingly, not only *some* or *few* functions can be computed fairly, but rather *almost all* functions in this class can be computed with complete fairness.

Although only one bit of information is revealed by output, the class of boolean functions that define full-dimensional geometric object is very rich, and includes fortune of interesting and non-trivial tasks. For instance, the task of *set-membership*, where P_1 holds some set $S \subseteq \Omega$, P_2 holds an element $x \in \Omega$, and the parties wish to find (privately) whether $x \in S$, is a part of this class.

Other examples are tasks like *private matchmaking* and *secure evaluation of a private (boolean) function*, where the latter task is very general and can be applied in many practical situations. Unexpectedly, it turns out that all of these tasks can be computed with complete fairness.

In addition to the above, we provide an additional property that indicates that a function *cannot* be computed using the protocol of GHKL (with the particular simulation strategy described in [GHKL08]). This property is almost always satisfied in the case where $|X| = |Y|$. Thus, at least at the intuitive level, almost all functions with $|X| \neq |Y|$ can be computed fairly, whereas almost all functions with $|X| = |Y|$ cannot be computed using the only known possibility result that we currently have in fairness. We emphasize that this negative result does not rule out the possibility of these functions using some other protocols or even the [GHKL08] protocol itself using some other simulation strategy. Combining this result with [ALR13] (i.e., characterization of coin-tossing), there exists a large class of functions for which the only known possibility result does not apply, the only known impossibility result does not apply either, and so fairness for this set of functions is left as an interesting open problem.

Furthermore, we also consider larger families of functions rather than the symmetric boolean functions with finite domain, and show that fairness is also possible in these classes. We consider the class of asymmetric functions where the parties do not necessarily get the same output, as well as the class of functions with non-binary outputs. This is the first time that fairness is shown to be possible in both families of functions, and it shows that fairness can be achieved in a much larger and wider class of functions than previously known.

Intuition. We present some intuition before proceeding to our results in more detail. The most important and acute point is to understand what distinguishes functions that can be computed fairly from functions that cannot. Towards this goal, let us reconsider the impossibility result of Cleve. This result shows that fair coin-tossing is impossible by constructing concrete adversaries that *bias* and *influence* the output of the honest party in any protocol implementing coin-tossing. We believe that such adversaries can be constructed for any protocol computing any function, and not specific to coin-tossing. In any protocol, one party can better predict the outcome than the other, and abort the execution if it is not satisfied with the result. Consequently, it has a concrete ability to *influence* the output of the honest party by aborting prematurely. Of course, a fair protocol should limit and decrease this ability to the least possible, but in general, this phenomenon cannot be totally eliminated and cannot be prevented.

So if this is the case, how do fair protocols exist? The answer to this question does not lie in the real execution but rather in the ideal process: *the simulator can simulate this influence in the ideal execution.* In some sense, for some functions, the simulator has the ability to significantly influence the output of the honest party in the ideal execution and therefore the bias in the real execution is not considered a breach of security. This is due to the fact that in the malicious setting the simulator has an ability that is crucial in the context of fairness: it can *choose* what input it sends to the trusted party. Indeed, the protocol of GHKL uses this switching-input ability in the simulation, and as pointed out by [ALR13], once we take away this advantage from the simulator – every function that contains an embedded XOR cannot be computed fairly, and fairness is almost always impossible.

Therefore, the structure of the function plays an essential role in the question of whether a function can be computed fairly or not. This is because this structure reflects the "power" and the "freedom" that the simulator has in the ideal world and how it can influence the output of the honest party. The question of whether a function can be computed fairly is related to the amount of "power" the simulator has in the ideal execution. Intuitively, the more freedom that the simulator has, it is more likely that the function can be computed fairly.

A concrete example. We demonstrate this "power of the simulator" on two functions. The first is the XOR function, which is impossible to compute by a simple implication of Cleve's result. The second is the specific function for which GHKL has proved to be possible (which we call "the GHKL function"). The truth tables of the functions are given in Figure 1.

(a)

	y_1	y_2
x_1	0	1
x_2	1	0

(b)

	y_1	y_2
x_1	0	1
x_2	1	0
x_3	1	1

Figure 1: (a) The XOR function – impossible, (b) The GHKL function – possible

What is the freedom of the simulator in each case? Consider the case where P_1 is corrupted (that is, we can assume that P_1 is the first to receive an output, and thus it is "harder" to simulate). In the XOR function, let p be the probability that the simulator sends the input x_1 to the trusted party, and let $(1-p)$ be the probability that it sends x_2. Therefore, the output of P_2 in the ideal execution can be represented as $(q_1, q_2) = p \cdot (0,1) + (1-p) \cdot (1,0) = (1-p, p)$, which means that if P_2 inputs y_1, then it receives 1 with probability $1-p$, and if it uses input y_2, then it receives 1 with probability p. We call this vector *"the output distribution vector"* for P_2, and the set of all possible output distribution vectors reflects the freedom that the simulator has in the ideal execution. In the XOR function, this set is simply $\{(1-p, p) \mid 0 \leq p \leq 1\}$, which gives the simulator one degree of freedom. Any increment of the probability in the first coordinate, must be balanced with an equivalent decrement in the second coordinate, and vice versa.

On the other hand, consider the case of the GHKL function. Assume that the simulator chooses x_1 with probability p_1, x_2 with probability p_2 and x_3 with probability $1-p_1-p_2$. Then, all the output vector distributions are of the form:

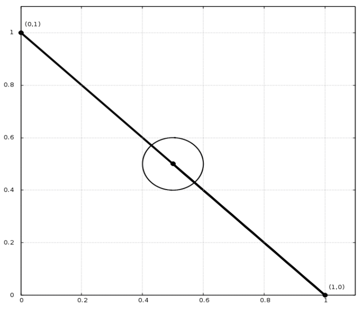
$$(q_1, q_2) = p_1 \cdot (0,1) + p_2 \cdot (1,0) + (1-p_1-p_2) \cdot (1,1) = (1-p_1, 1-p_2) \ .$$

This gives the simulator two degrees of freedom, which is significantly more power.

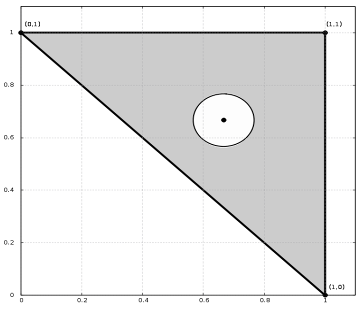
Geometrically, we can refer to the rows of the truth table as points in $\mathbb{R}^2$, and so in the XOR function we have the two points $(0,1)$ and $(1,0)$. All the output distribution vectors are of the form $p \cdot (0,1) + (1-p) \cdot (1,0)$ which is exactly the line segment between these two points (geometric object of dimension 1). In the GHKL function, all the output distribution vectors are the triangle between the points $(0,1), (1,0)$ and $(1,1)$, which is a geometric object of dimension 2 (a full dimensional object in $\mathbb{R}^2$).

The difference between these two geometric objects already gives a perception for the reason why the XOR function is impossible to compute, whereas the GHKL function is possible, as the simulator has significantly more options in the latter case. However, we provide an additional refinement. At least on the intuitive level, fix some output distribution vector of the honest party (q_1, q_2). Assume that there exists a real-world adversary that succeeds in biasing the output and obtain output distribution vector (q_1', q_2') that is at most ϵ-far from (q_1, q_2). In the case of the XOR function, this results in points that are not on the line, and therefore this adversary cannot be simulated. In contrast, in the case of the GHKL function, these points are still in the triangle, and therefore this adversary can be simulated.

In Figure 2, we show the geometric objects defined by the XOR and the GHKL functions. The centers of the circles are the output distribution of honest executions, and the circuits represent the possible biases in the real execution. In (a) there exist small biases that are invalid points, whereas in (b) all small biases are valid points that can be simulated.



(a) The potential output distribution vectors of the XOR function: a line segment between $(0,1)$ and $(1,0)$.



(b) The potential output distribution vectors of the GHKL function: the triangle between $(0,1)$, $(1,0)$ and $(1,1)$.

Figure 2: The geometric objects defined by the XOR function (a) and the GHKL function (b).

1.2 Our Results

For a given function $f : \{x_1, \ldots, x_\ell\} \times \{y_1, \ldots, y_m\} \to \{0, 1\}$, we consider its geometric representation as the convex-hull of ℓ points over $\mathbb{R}^m$, where the jth coordinate of the ith point is simply $f(x_i, y_j)$. We say that this geometric object is of full dimension in $\mathbb{R}^m$, if it cannot be embedded in any subspace of dimension smaller than m (or, any subspace that is isomorphic to $\mathbb{R}^{m-1}$). We then prove that any function that its geometric representation is of full dimension *can be computed with complete fairness.* We prove the following theorem:

Theorem 1.1 (informal) *Let $f : X \times Y \to \{0, 1\}$ be a function. Assuming the existence of an Oblivious Transfer, if the geometric object defined by f is of full-dimension, then the function can be computed with complete fairness.*

For the proof, we use the extended GHKL protocol (with some concrete set of parameters). Moreover, the proof uses tools from convex geometry. We find the connection between the problem of fairness and convex geometry very appealing.

On the other hand, we show that if the function is not full dimensional, and satisfies some additional requirements (that are almost always satisfied in functions with $|X| = |Y|$), then the function *cannot* be computed using the protocol of [GHKL08].

We then proceed to the class of asymmetric functions where the parties do not necessarily get the same output, and the class of non-binary output. Interestingly, the GHKL protocol can be extended to these classes of functions. We show:

Theorem 1.2 (informal) *Assuming the existence of an Oblivious Transfer,*

1. *There exists a large class of asymmetric Boolean functions that can be computed with complete fairness.*
2. *For any finite range Σ, there exists a large class of functions $f : X \times Y \to \Sigma$ that can be computed with complete-fairness.*

For the non-binary case, we provide a general criterion that holds only for functions for which $|X| > (|\Sigma| - 1) \cdot |Y|$, that is, when the ratio between the domain sizes is greater than $|\Sigma| - 1$. This, together with the results in the binary case, may refer to an interesting relationship between the size of the domains and possibility of fairness. This is the first time that a fair protocol is constructed for both non-binary output, and asymmetric Boolean functions. This shows that fairness is not restricted to a very specific and particular type of functions, but rather a property that under certain circumstances can be achieved. Moreover, it shows the power that is concealed in the GHKL protocol alone.

Related work. Several other impossibility results regarding fairness, rather than the result of Cleve, have been published [EY80, AP13]. However, it seems that only Cleve's impossibility can be reduced into the family of boolean functions with finite domain. The work of [ALR13] identifies which function imply fair coin-tossing and are ruled out by the impossibility result of Cleve. Interestingly, the class of functions that imply fair coin-tossing shares a similar (but yet distinct) algebraic structure with the class of functions that we show that cannot be computed using the GHKL protocol. We link between the two criterions in the body of our work.

For decades fairness was believed to be impossible, and so researchers have simply resigned themselves to being unable to achieve this goal. Therefore, a huge amount of works consider several relaxations like gradual release, partial fairness and rational adversaries ([Cle89, GL90, BGMR90, GK10, BLOO11, HT04] to state a few. See [Gor10] for a survey of fairness in secure computation).

Open problems. Our work is an important step towards a full characterization of fairness of finite domain functions. The main open question is to finalize this characterization. In addition, can our results be generalized to functions with infinite domains (domains with sizes that depend on the security parameter)? Finally, in the non-binary case, we have a positive result only when the ratio between the domain sizes is greater than $|\Sigma| - 1$. A natural question is whether fairness be achieved in any other case, or for any other ratio.

2 Definitions and Preliminaries

We present the necessary definitions for secure computation, which are merely the standard stand alone definitions [Can00, Gol04]. We distinguish between security-with-abort (subsection 2.1.2), for which the adversary may receive output while the honest party does not (security without fairness), and security with fairness (subsection 2.1.1), where all parties receive output (this is similar to security with respect to honest majority, although we do not have honest majority). In addition, we cover the mathematical background that is needed for our results.

Notations. In most of the paper, we consider binary deterministic functions over a finite domain; i.e., functions $f : X \times Y \to \{0, 1\}$ where $X, Y \subset \{0, 1\}^*$ are finite sets. Throughout the paper, we denote $X = \{x_1, \ldots, x_\ell\}$ and $Y = \{y_1, \ldots, y_m\}$, for constants $\ell, m \in \mathbb{N}$. Let M_f be the $\ell \times m$ matrix that represents the function, i.e., a matrix whose entry position (i, j) is $f(x_i, y_j)$. For $1 \leq i \leq \ell$, let X_i denote the ith row of M_f, and for $1 \leq j \leq m$ let Y_j denote the jth column of M_f. A vector $\mathbf{p} = (p_1, \ldots, p_\ell)$ is a *probability vector* if $p_i \geq 0$ for every $1 \leq i \leq \ell$ and $\sum_{i=1}^{\ell} p_i = 1$. As a convention, we use **bold**-case letters to represent a vector (e.g., $\mathbf{p}$, $\mathbf{q}$), and sometimes we use upper-case letters (e.g., X_i, as above). We denote by $\mathbf{1}_k$ (resp. $\mathbf{0}_k$) the all one (resp. all zero) vector of size k. We work in the Euclidian space $\mathbb{R}^m$, and use the Euclidian norm $||x|| = \sqrt{\langle x, x \rangle}$ and the distance function as $d(x, y) = ||x - y||$.

2.1 Secure Computation – Definitions

We let κ denote the security parameter. We use standard O notation, and let poly denote a polynomial function. A function $\mu(\cdot)$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large κ it holds that $\mu(n) < 1/p(\kappa)$. A *distribution ensemble* $X = \{X(a,\kappa)\}_{a\in\mathcal{D},\kappa\in\mathbb{N}}$ is an infinite sequence of random variables indexes by $a \in \mathcal{D}$ and $\kappa \in \mathbb{N}$. In the context of secure computation, κ is the security parameter and $\mathcal{D}$ denotes the domain of the parties' input. Two distribution ensembles $X = \{X(a,\kappa)\}_{a\in\mathcal{D},\kappa\in\mathbb{N}}$ and $Y = \{Y(a,\kappa)\}_{a\in\mathcal{D},\kappa\in\mathbb{N}}$ are *computationally indistinguishable*, denoted $X \overset{\rm c}{\equiv} Y$, if for every non-uniform polynomial-time-algorithm D there exists a negligible function $\mu(\cdot)$ such that for every κ and every $a \in \mathcal{D}$:

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

In most of the paper, we consider binary deterministic functions over a finite domain; i.e., functions $f : X \times Y \to \{0,1\}$ where $X, Y \subset \{0,1\}^*$ are finite sets. Throughout, we will denote $X = \{x_1,\ldots,x_\ell\}$ and $Y = \{y_1,\ldots,y_m\}$, for constants $\ell, m \in \mathbb{N}$.

2.1.1 Secure Two-Party Computation with Fairness

We now define what it means for a protocol to be *secure with complete fairness.* Our definition follows the standard definition of [Can00, Gol04], except for the fact that we require complete fairness even though we are in the two-party setting. We consider active adversaries (malicious), who may deviate from the protocol in an arbitrary manner, and static corruptions. Let π be a two party protocol for computing a two party function $f : X \times Y \to (\{0,1\}^*)^2$ be a two party function, and let $f_1(x,y)$ (resp. $f_2(x,y)$) denote the first (resp. second) element of $f(x,y)$ (in most cases, we will consider $f_1(x,y) = f_2(x,y)$, i.e., both parties get the same output). We briefly describe the real execution and the ideal execution.

Execution in the ideal model. An ideal execution involves parties P_1 and P_2, an adversary $\mathcal{S}$ who has corrupted one of the parties, and the trusted party. An ideal execution for the computation of f proceeds as follows:

Inputs: P_1 and P_2 hold inputs $x \in X$, and $y \in Y$, respectively; the adversary $\mathcal{S}$ receives the security parameter 1^κ and an auxiliary input z.

Send inputs to trusted party: The honest party sends its input to the trusted party. The corrupted party controlled by $\mathcal{S}$ may send any value of its choice. Denote the pair of inputs sent to the trusted party by (x', y').

Trusted party sends outputs: If $x' \notin X$, the trusted party sets x' to be the default value x_1; likewise if $y' \notin Y$ the trusted party sets $y' = y_1$. Next, the trusted party computes $f(x',y') = (f_1(x',y'), f_2(x',y'))$ and sends the result $f_1(x',y')$ to P_1 and $f_2(x',y')$ to P_2.

Outputs: The honest party outputs whatever it was sent by the trusted party, the corrupted party outputs nothing and $\mathcal{S}$ outputs an arbitrary function of its view.

We denote by $\textsc{ideal}_{f,\mathcal{S}(z)}(x,y,\kappa)$ the random variable consisting of the output of the adversary and the output of the honest party following an execution in the ideal model as described above.

Execution in the real model. In the real execution, the parties P_1 and P_2 interact, where one is corrupted and therefore controlled by the real-world adversary $\mathcal{A}$. In this case, the adversary $\mathcal{A}$ gets the inputs of the corrupted party and sends all messages on behalf of this party, using an arbitrary strategy. The honest party follows the instructions of π. Let $\text{REAL}_{\pi,\mathcal{A}(z)}(x, y, \kappa)$ be the random variable consisting of the view of the adversary and the output of the honest party, following an execution of π where P_1 begins with input x, P_2 with input y, the adversary has auxiliary input z, and all parties have security parameter 1^κ.

Security by emulation. Informally, a real protocol π is secure if any "attack" carried out by a real adversary $\mathcal{A}$ on π can be carried out by the ideal adversary $\mathcal{S}$ in the ideal model. This is formalized by requiring that $\mathcal{S}$ can simulate the real-model outputs while running in the ideal model.

Definition 2.1 *Protocol* π securely computes f with complete fairness in the presence of malicious adversaries *if for every non-uniform probabilistic polynomial-time adversary* $\mathcal{A}$ *in the real model, there exists a non-uniform probabilistic polynomial-time adversary* $\mathcal{S}$ *in the ideal model such that:*

$$\left\{\text{IDEAL}_{f,\mathcal{S}(z)}(x, y, \kappa)\right\}_{x\in X, y\in Y, z\in\{0,1\}^*, \kappa\in\mathbb{N}} \stackrel{c}{\equiv} \left\{\text{REAL}_{\pi,\mathcal{A}(z)}(x, y, \kappa)\right\}_{x\in X, y\in Y, z\in\{0,1\}^*, \kappa\in\mathbb{N}} .$$

2.1.2 Secure Two-Party Computation without Fairness (Security with Abort)

The following is the standard definition of secure computation *without fairness.* This is formalized by changing the ideal world and allowing the adversary to learn the output alone. This implies that the case where the adversary learns the output without the honest party is not considered as a breach of security since this is allowed in the ideal world. We modify the ideal world as follows:

Inputs: As previously.

Send inputs to trusted party: As previously.

Trusted party sends output to corrupted party: If $x' \notin X$, the trusted party sets x' to be the default value x_1; likewise if $y' \notin Y$ the trusted party sets $y' = y_1$. Next, the trusted party computes $f(x', y') = (f_1(x', y'), f_2(x', y')) = (w_1, w_2)$ and sends w_i to the the corrupted party P_i (i.e., to the adversary $\mathcal{A}$).

Adversary decides whether to abort: After receiving its output, the adversary sends either proceed or abort message to the trusted party. If its sends proceed to the trusted party, the trusted party sends w_j to the honest party P_j. If it sends abort, then it sends the honest party P_j the special symbol $\perp$.

Outputs: As previously.

We denote by $\text{IDEAL}^{\mathsf{abort}}_{f,\mathcal{S}(z)}(x, y, \kappa)$ the random variable consisting of the output of the adversary and the output of the honest party following an execution in the ideal model with abort as described above. The following definition is the analogue to Definition 2.1 and formalizes security with abort:

Definition 2.2 *Protocol π* securely computes f with abort in the presence of malicious adversaries *if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ in the real model, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ in the ideal model such that:*

$$\left\{\text{IDEAL}^{\mathsf{abort}}_{f,\mathcal{S}(z)}(x,y,\kappa)\right\}_{x\in X,y\in Y,z\in\{0,1\}^*,\kappa\in\mathbb{N}} \overset{c}{\equiv} \left\{\text{REAL}_{\pi,\mathcal{A}(z)}(x,y,\kappa)\right\}_{x\in X,y\in Y,z\in\{0,1\}^*,\kappa\in\mathbb{N}} .$$

We remark that in the case of two-party computation, for any functionality f there exists a protocol π that securely computes it with security-with-abort [GMW87, Gol04]. This is true also for reactive functionalities, that is, functionalities that consist of several stages, where in each stage the parties can give inputs and get outputs from the current stage of f, and the function f stores some state between its stages.

2.1.3 Hybrid Model and Composition

The hybrid model combines both the real and ideal models. Specifically, an execution of a protocol π in the g-hybrid model, for some functionality g, involves the parties sending normal messages to each other (as in the real model) and, in addition, having access to a trusted party computing g. The composition theorems of [Can00] imply that if π securely computes some functionality f in the g-hybrid model, and a protocol ρ securely computes g, then the protocol π^ρ (where every ideal call of g is replaced with an execution of ρ) securely-computes f in the real world.

In our work, we consider the case where g is a reactive functionality, and assume that the protocol π^g that uses the functionality g does not contain any messages between the parties (i.e., all the communication between them is performed via the functionality g). Recall that any functionality (even reactive one) can be computed with security-with-abort, and therefore there exists a protocol ρ (or, in case of reactive functionalities, several protocols $\rho_1,\ldots,\rho_m$) that securely computes g. Since there are no messages in π^g we can apply the sequential composition theorem of [Can00] and obtain the following Proposition:

Proposition 2.3 *Let g be a reactive functionality, let π be a protocol that securely computes f with complete fairness in the g-hybrid model (where g is computed according to the ideal world with abort), and assumes that π contains no communication between the parties rather than queries of g and that a single execution of g occurs at the same time. Then, there exists a protocol Π that securely computes f with complete fairness in the plain model.*

2.2 Mathematical Background

Our characterization is based on the geometric representation of the function f. In the following, we provide the necessary mathematical background, and link it to the context of cryptography whenever possible. Most of the following mathematical definitions are taken from [Rom08, Grü03].

Output vector distribution and convex combination. We now analyze the "power of the simulator" in the ideal execution. The following is an inherent property of the concrete function and the ideal execution, and is true for any protocol computing the function. Let $\mathcal{A}$ be an adversary that corrupts the party P_1, and assume that the simulator $\mathcal{S}$ chooses its input according to some distribution $\mathbf{p} = (p_1,\ldots,p_\ell)$. That is, the simulator sends an input x_i with probability p_i, for $1 \le i \le \ell$. Then, the length m vector $\mathbf{q} = (q_{y_1},\ldots,q_{y_m}) \overset{\text{def}}{=} \mathbf{p}\cdot M_f$ represents the *output distribution vector* of the honest party P_2. That is, in case the input of P_2 is y_j for some $1 \le j \le m$, then it gets 1 with probability q_{y_j}.

Convex combination. The output distribution vector is in fact a convex combination of the rows $\{X_1, \ldots, X_\ell\}$ of the matrix M_f. That is, when the simulator uses $\mathbf{p}$, the output vector distribution of P_2 is:

$$\mathbf{p} \cdot M_f = (p_1, \ldots, p_\ell) \cdot M_f = p_1 \cdot X_1 + \ldots + p_\ell \cdot X_\ell \ .$$

A convex combination of points $X_1, \ldots, X_\ell$ in $\mathbb{R}^m$ is a linear combination of the points, where all the coefficients (i.e., $(p_1, \ldots, p_\ell)$) are non-negative and sum up to 1.

Convex hull. The set of all possible output distributions vectors that the simulator can produce in the ideal execution is:

$$\{\mathbf{p} \cdot M_f \mid \mathbf{p} \text{ is a probability vector}\} \ .$$

In particular, this set reflects the "freedom" that the simulator has in the ideal execution. This set is in fact, the *convex hull* of the row vectors $X_1, \ldots, X_\ell$, and is denoted as $\mathbf{conv}(\{X_1, \ldots, X_\ell\})$. That is, for a set $S = \{X_1, \ldots, X_\ell\}$, $\mathbf{conv}(S) = \left\{\sum_{i=1}^{\ell} p_i \cdot X_i \mid 0 \leq p_i \leq 1, \sum_{i=1}^{m} p_i = 1\right\}$. The convex-hull of a set of points is a convex set, which means that for every $X, Y \in \mathbf{conv}(S)$, the line segment between X and Y also lies in $\mathbf{conv}(S)$, that is, for every $X, Y \in \mathbf{conv}(S)$ and for every $0 \leq \lambda \leq 1$, it holds that $\lambda \cdot X + (1 - \lambda) \cdot Y \in \mathbf{conv}(S)$.

Geometrically, the convex-hull of two (distinct) points in $\mathbb{R}^2$, is the line-segment that connects them. The convex-hull of three points in $\mathbb{R}^2$ may be a line (in case all the points lie on a single line), or a triangle (in case where all the points are collinear). The convex-hull of 4 points may be a line, a triangle, or a parallelogram. In general, the convex-hull of k points in $\mathbb{R}^2$ may define a convex polygon of at most k vertices. In $\mathbb{R}^3$, the convex-hull of k points can be either a line, a triangle, a tetrahedron, a parallelepiped, etc.

Affine-hull and affine independence. A subset B of $\mathbb{R}^m$ is an affine subspace if $\lambda \cdot \mathbf{a} + \mu \cdot \mathbf{b} \in B$ for every $\mathbf{a}, \mathbf{b} \in B$ and $\lambda, \mu \in \mathbb{R}$ such that $\lambda + \mu = 1$. For a set of points $S = \{X_1, \ldots, X_\ell\}$, its affine hull is defined as: $\mathbf{aff}(S) = \left\{\sum_{i=1}^{\ell} \lambda_i \cdot X_i \mid \sum_{i=1}^{\ell} \lambda_i = 1\right\}$, which is similar to convex hull, but without the additional requirement for non-negative coefficients. The set of points $X_1, \ldots, X_\ell$ in $\mathbb{R}^m$ is affinely independent if $\sum_{i=1}^{\ell} \lambda_i X_i = \mathbf{0}_m$ holds with $\sum_{i=1}^{\ell} \lambda_i = 0$ only if $\lambda_1 = \ldots = \lambda_\ell = 0$. In particular, it means that one of the points is in the affine hull of the other points. It is easy to see that the set of points $\{X_1, \ldots, X_\ell\}$ is affinely independent if and only if the set $\{X_2 - X_1, \ldots, X_\ell - X_1\}$ is a linearly independent set. As a result, any $m + 2$ points in $\mathbb{R}^m$ are affine dependent, since any $m+1$ points in $\mathbb{R}^m$ are linearly dependent. In addition, it is easy to see that the points $\{X_1, \ldots, X_\ell\}$ over $\mathbb{R}^m$ is affinely independent if and only if the set of points $\{(X_1, 1), \ldots, (X_\ell, 1)\}$ over $\mathbb{R}^{m+1}$ is linearly independent.

Affine-dimension and affine-basis. If the set $S = \{X_1, \ldots, X_\ell\}$ over $\mathbb{R}^m$ is affinely independent, then $\mathbf{aff}(S)$ has dimension $\ell - 1$, and we write $\dim(\mathbf{aff}(S)) = \ell - 1$. In this case, S is the affine basis for $\mathbf{aff}(S)$. Note that an affine basis for an m-dimensional affine space has $m + 1$ elements. As we will see, the affine dimension of ℓ row vectors $X_1, \ldots, X_\ell$ play an essential role in our characterization.

Linear hyperplane. A linear hyperplane in $\mathbb{R}^m$ is a $(m-1)$-dimensional affine-subspace of $\mathbb{R}^m$. The linear hyperplane can be defined as all the points $X = (x_1, \ldots, x_m)$ which are the solutions of a linear equation:

$$a_1x_1 + \ldots a_mx_m = b \ ,$$

for some constants $\mathbf{a} = (a_1, \ldots, a_m) \in \mathbb{R}^m$ and $b \in \mathbb{R}$. We denote this hyperplane by:

$$\mathcal{H}(\mathbf{a}, b) \stackrel{\text{def}}{=} \{X \in \mathbb{R}^m \mid \langle X, \mathbf{a} \rangle = b\} \ .$$

Throughout the chapter, for short, we will use the term hyperplane instead of linear hyperplane. It is easy to see that indeed this is an affine-subspace. In $\mathbb{R}^1$, an hyperplane is a single point, in $\mathbb{R}^2$ it is a line, in $\mathbb{R}^3$ it is a plane and so on. We remark that for any m affinely independent points in $\mathbb{R}^m$ there exists a *unique* hyperplane that contains all of them (and infinitely many in case they are not affinely independent). This is a simple generalization of the fact that for any distinct 2 points there exists a single line that passes through them, for any 3 (collinear) points there exists a single plane that contains all of them and etc.

Convex polytopes. Geometrically, a full dimensional convex polytope in $\mathbb{R}^m$ is the convex-hull of a finite set S where $\dim(\mathbf{aff}(S)) = m$. Polytopes are familiar objects: in $\mathbb{R}^2$ we get *convex polygons* (a triangle, a parallelogram etc.). In $\mathbb{R}^3$ we get *convex polyhedra* (a tetrahedron, a parallelepiped etc.). Convex polytopes play an important role in solutions of linear programming.

In addition, a special case of polytope is simplex. If the set S is affinely independent of cardinality $m + 1$, then $\mathbf{conv}(S)$ is an m-dimensional *simplex* (or, m-simplex). For $m = 2$, this is simply a triangle, whereas in $m = 3$ we get a tetrahedron. A simplex in $\mathbb{R}^m$ consists of $m + 1$ *facets*, which are themselves simplices of lower dimensions. For instance, a tetrahedron (which is a 3-simplex) consists of 4 facets, which are themselves triangles (2-simplex).

3 The Protocol of Gordon, Hazay, Katz and Lindell [GHKL08]

In the following, we give a high level overview of [GHKL08]. We also present its simulation strategy, and the set of equations that indicates whether a given function can be computed with this protocol, which is the important part for our discussion. We also generalize a bit the protocol by adding additional parameters, which adds some flexibility to the protocol and may potentially compute more functions, and we also represent it a bit differently than the original construction, which is merely a matter of taste.

3.1 The Protocol

Assume the existence of an online dealer (a reactive functionality that can be replaced using standard secure computation that is secure-with-abort). The parties invoke this online-dealer and send it their respective inputs $(x, y) \in X \times Y$. The online dealer computes values $a_1, \ldots, a_R$ and $b_1, \ldots, b_R$ (we will see later how they are defined). In round i the dealer sends party P_1 the value a_i and afterward it sends b_i to P_2. At each point of the execution, each party can halt the online-dealer, preventing the other party from receiving its value at that round. In such a case, the other party is instructed to halt and output the last value it has received from the dealer. For instance, if P_1 aborts at round i after it learns a_i and prevents from P_2 to learn b_i, P_2 halts and outputs b_{i-1}.

The values $(a_1, \ldots, a_R), (b_1, \ldots, b_R)$ are generated by the dealer in the following way: The dealer first chooses a round i^* according to geometric distribution with parameter α. In each round $i < i^*$, the parties receive bits (a_i, b_i), that depend on their respective inputs solely and uncorrelated to the input of the other party. In particular, for party P_1 the dealer computes $a_i = f(x_i, \hat{y})$ for some random $\hat{y}$, and for P_2 it computes $b_i = f(\hat{x}, y_j)$ for some random $\hat{x}$. As we will see, we will choose $\hat{y}$ uniformly from Y, whereas $\hat{x}$ will be chosen according to some distribution X_{real}, which is a parameter for the protocol[2]. For every round $i \geq i^*$, the parties receive the correct output $a_i = b_i = f(x_i, y_j)$. Note that if we set $R = \alpha^{-1} \cdot \omega(\ln \kappa)$, then $i^* < R$ with overwhelming probability, and so correctness holds.

The full specification consists of the definition of the online-dealer, which is the F_{dealer} functionality and is formally defined in Functionality 3.2. In addition, we present the protocol in the F_{dealer}-hybrid model in Protocol 3.3. Algorithm 3.1 is needed for both the F_{dealer} functionality and the protocol.

$\mathsf{RandOut}_1(x)$ – Algorithm for P_1:	**$\mathsf{RandOut}_2(y)$ – Algorithm for P_2:**
Input: An input $x \in X$. **Output:** Choose $\hat{y} \leftarrow Y$ uniformly and output $f(x, \hat{y})$.	**Input:** An input $y \in Y$. **Parameter:** Distribution X_{real}. **Output:** Choose $\hat{x} \leftarrow X$ according to distribution X_{real} and output $f(\hat{x}, y)$.

ALGORITHM 3.1 (Default output algorithms – $\mathsf{RandOut}_1(x)$, $\mathsf{RandOut}_2(y)$)

FUNCTIONALITY 3.2 (The (reactive) online-dealer functionality – F_{dealer})

- **Inputs:** P_1 sends x as its input, Y sends y.
- **Parameters:** (α, R), where $0 < \alpha < 1$ and R is the number of rounds.
- **The functionality:**
  1. **Check inputs:** If $x \notin X$ or $y \notin Y$ then send abort to both parties and halt.
  2. **Preliminary phase:**
     1. Choose i^* according to a geometric distribution with parameter α.
     2. Define values $(a_1, \ldots, a_R)$, $(b_1, \ldots, b_R)$, where:
        (a) For $i = 1$ to $i^* - 1$: set $(a_i, b_i) = (\mathsf{RandOut}_1(x), \mathsf{RandOut}_2(y))$.
        (b) For $i = i^*$ to R: set $(a_i, b_i) = (f(x, y), f(x, y))$.
  3. **The online phase:**
     1. In every round $i = 1$ to R:
        (a) Upon receiving proceed from P_2, send a_i to P_1.
        (b) Upon receiving proceed from P_1, send b_i to P_2.
        Upon receiving abort from any one of the parties at any point of the execution, send abort to both parties and halt.

[2]This is the generalization of the protocol of GHKL that we have mentioned, which adds some flexibility, and will make the proof of security a bit simpler. We note that a similar distribution Y_{real} could have been added as well, but as we will see, this modification is not helpful and will just add complication.

PROTOCOL 3.3 (Computing f in the F_{dealer}-hybrid model)

- **Inputs:** P_1 holds x, P_2 holds y.
- **Parameters:** Some value $0 < \alpha < 1$, and security parameter κ.
- **The protocol:**
  1. Set $R = \alpha^{-1} \cdot \omega(\log \kappa)$.
  2. P_1 computes $a_0 = \mathsf{RandOut}_1(x)$, P_2 computes $b_0 = \mathsf{RandOut}_2(y)$.
  3. The parties invoke the F_{dealer} functionality. P_1 sends as input x, P_2 sends y.
  4. For every round $i = 1$ to R:
     (a) P_2 sends proceed to F_{dealer}, and P_1 receives a_i from F_{dealer}. If P_1 receives abort from F_{dealer}, it halts and outputs a_{i-1}.
     (b) P_1 sends proceed to F_{dealer}, and P_2 receives b_i from F_{dealer}. If P_2 receives abort from F_{dealer}, it halts and outputs b_{i-1}.
- **Output:** If all R iterations have been run, party P_1 outputs a_R and party P_2 outputs b_R.

3.2 Security

Since P_2 is the second to receive an output, it is easy to simulate an adversary that corrupts P_2. If the adversary aborts before i^*, then it has not obtained any information about the input of P_1. If the adversary aborts at or after i^*, then in the real execution the honest party P_1 already receives the correct output $f(x_i, y_j)$, and fairness is obtained. Therefore, the protocol is secure with respect to corrupted P_2, *for any function f*.

The case of corrupted P_1 is more delicate, and defines some requirements from f. Intuitively, if the adversary aborts before i^*, then the outputs of both parties are uncorrelated, and no one gets any advantage. If the adversary aborts after i^*, then both parties receive the correct output and fairness is obtained. The worst case, then, occurs when P_1 aborts exactly in iteration i^*, as P_1 has then learned the correct value of $f(x_i, y_j)$ while P_2 has not. Since the simulator has to give P_1 the true output if it aborts at i^*, it sends the trusted party the *true* input x_i in round i^*. As a result, P_2 in the ideal execution learns the correct output $f(x_i, y_j)$ at round i^*, unlike the real execution where it outputs a random value $f(\hat{x}, y_j)$. [GHKL08] overcomes this problem in a very elegant way: in order to balance this advantage of the honest party in the ideal execution in case the adversary aborts at i^*, the simulator chooses a random value $\hat{x}$ *different* from the way it is chosen in the real execution in case the adversary abort *before* i^* (that is, according to a different distribution than the one the dealer uses in the real execution). The calculations show that overall, the output distribution of the honest party is distributed identically in the real and ideal executions. This balancing is possible only sometimes, and depends on the actual function f that is being evaluated.

In more detail, in the real execution the dealer before i^* chooses b_i as $f(\hat{x}, y_j)$, where $\hat{x}$ is chosen according to some distribution X_{real}. In the ideal execution, in case the adversary sends x to the simulated online-dealer, aborts at round $i < i^*$ upon viewing some a_i, the simulator chooses the input $\tilde{x}$ it sends to the trusted party according to distribution X_{ideal}^{x,a_i}. Then, define $Q^{x,a_i} = X_{ideal}^{x,a_i} \cdot M_f$, the output distribution vector of the honest party P_2 in this case. In fact, the protocol and the simulation define the output distribution vectors Q^{x,a_i}, and simulation is possible only if the corresponding X_{ideal}^{x,a_i} exists, which depends on the function f being computed.

We now present the exact requirements from the output distribution vectors Q^{x,a_i}. In the proof sketch below, we do not present the simulator nor a full proof of security; We just give a perception for the reason why the vectors Q^{x,a_i} are defined like that. The full proof can be found in Appendix A and [GHKL08].

The output distributions vectors $Q^{x,a}$. Let $f : \{x_1, \ldots, x_\ell\} \times \{y_1, \ldots, y_m\} \to \{0,1\}$. Fix X_{real}, and let U_Y denote the uniform distribution over Y. For every $x \in X$, denote by p_x the probability that $a_i = 1$ before i^*. Similarly, for every $y_j \in Y$, let p_{y_j} denote the probability $b_i = 1$ before i^*. That is: $p_x \stackrel{\text{def}}{=} \Pr_{\hat{y} \leftarrow U_Y}[f(x,\hat{y}) = 1]$, and $p_{y_j} \stackrel{\text{def}}{=} \Pr_{\hat{x} \leftarrow X_{real}}[f(\hat{x}, y_j) = 1]$. For every $x \in X$, $a \in \{0,1\}$, define the row vectors $Q^{x,a} = (q^{x,a}_{y_1}, \ldots, q^{x,a}_{y_m})$ indexed by $y_j \in Y$ as follows:

$$q^{x,0}_{y_j} \stackrel{\text{def}}{=} \begin{cases} p_{y_j} & \text{if } f(x,y_j) = 1 \\ p_{y_j} + \frac{\alpha \cdot p_{y_j}}{(1-\alpha)\cdot(1-p_x)} & \text{if } f(x,y_j) = 0 \end{cases} \quad q^{x,1}_{y_j} \stackrel{\text{def}}{=} \begin{cases} p_{y_j} + \frac{\alpha\cdot(p_{y_j}-1)}{(1-\alpha)\cdot p_x} & \text{if } f(x,y_j) = 1 \\ p_{y_j} & \text{if } f(x,y_j) = 0 \end{cases} \tag{1}$$

For every $x \in X, a \in \{0,1\}$ there exists a probability vector $X^{x,a}_{ideal}$ such that $X^{x,a}_{ideal} \cdot M_f = Q^{x,a}$, then the simulator succeeds to simulate the protocol. We therefore have the following theorem:

Theorem 3.4 *Let $f : \{x_1, \ldots, x_\ell\} \times \{y_1, \ldots, y_m\} \to \{0,1\}$ and let M_f be as above. If there exist probability vector X_{real} and a parameter $0 < \alpha < 1$ (where $\alpha^{-1} \in O(\mathsf{poly}(\kappa))$), such that for every $x \in X$, $a \in \{0,1\}$, there exists a probability vector $X^{x,a}_{ideal}$ for which:*

$$X^{x,a}_{ideal} \cdot M_f = Q^{x,a} ,$$

then the protocol securely computes f with complete fairness.

Proof Sketch: The full proof, including the case where P_2 is corrupted, appears in Appendix A. Here, we consider the case where P_1 is corrupted. Let $\mathcal{A}$ be the adversary that corrupts it, and let $\mathcal{S}$ be the simulator mentioned above. Let x be the input the adversary sends to the simulated F_{dealer} functionality. We recall that in case the adversary aborts exactly at i^*, the simulator $\mathcal{S}$ sends the input x to the trusted party, and so both parties receive $f(x,y)$, unlike the real execution. Moreover, in case the adversary has aborted at round $i < i^*$, upon viewing a at round i, the simulator $\mathcal{S}$ chooses input $\hat{x}$ according to distribution $X^{x,a}_{ideal}$. Full specification of the simulator appears in the full proof.

We show that the joint distribution of the view of the adversary and the output of the honest party is distribution identically in the hybrid and the ideal executions. This is done easily in the case where the adversary aborts some round $i > i^*$ (and thus, both parties receive the correct output $f(x,y)$), and is given in the full proof of this theorem. Now, we consider the case where $i \leq i^*$. In the full proof we show that the view of the adversary until round i (i.e., of the first $i-1$ rounds) is distributed identically in both executions. Thus, all is left to show is that the view of the adversary in the last round and the output of the honest party are distributed identically in both executions, and this is what we show here in this proof sketch. That is, we show that for every $(a,b) \in \{0,1\}^2$,

$$\begin{aligned} &\Pr\left[(\text{VIEW}^i_{\mathsf{hyb}}, \text{OUT}_{\mathsf{hyb}}) = (a,b) \;\mid\; i \leq i^*\right] \\ &\quad = \Pr\left[(\text{VIEW}^i_{\mathsf{ideal}}, \text{OUT}_{\mathsf{ideal}}) = (a,b) \;\mid\; i \leq i^*\right] \end{aligned} \tag{2}$$

where $(\text{VIEW}^i_{\mathsf{hyb}}, \text{OUT}_{\mathsf{hyb}})$ denote the view of the adversary at round i (i.e., the round where it has aborted), and the output of the honest party in the hybrid execution. $\text{VIEW}^i_{\mathsf{ideal}}, \text{OUT}_{\mathsf{ideal}})$ denote the outputs in the ideal execution.

We now show that these probabilities are equal. First, observe that:

$$\Pr[i = i^* \mid i \leq i^*] = \alpha \qquad \text{and} \qquad \Pr[i < i^* \mid i \leq i^*] = 1 - \alpha$$

We now show that Eq. (2) holds, by considering all possible values for (a,b).

In case $\boldsymbol{f(x,y)=0}$. We compute the probability that the outputs of the parties are $(0,0)$ in the real execution, when the adversary $\mathcal{A}$ aborts at round $i \leq i^*$. with probability α, we get that $i = i^*$. In this case, the output of the adversary P_1 is always the correct output 0. On the other hand, the output of the honest party P_2 is chosen independently according to $\mathsf{RandOut}_2(y)$. With probability $1-\alpha$, we have that $i < i^*$. In this case *both* parties get uncorrelated values, chosen according to $\mathsf{RandOut}_1(x), \mathsf{RandOut}_2(y)$. Thus, the probability that we get $(0,0)$ is:

$$\Pr\left[(\text{VIEW}^i_{\mathsf{hyb}}, \text{OUT}_{\mathsf{hyb}}) = (0,0) \ \mid \ i \leq i^*\right] = \alpha \cdot 1 \cdot (1-p_y) + (1-\alpha)\cdot(1-p_x)\cdot(1-p_y)$$

On the other hand, in the ideal execution, with probability α *both* parties receive the true output $f(x,y)$. With probability $1-\alpha$, we have that $i < i^*$. Thus, the simulator sends the adversary the value 0 with probability $(1-p_x)$, and in case it aborts, it chooses the input $\hat{x}$ according to distribution $X^{x,0}_{ideal}$. Since $Q^{x,0} = X^{x,0}_{ideal} \cdot M_f$, the probability that the honest party receives 0 is exactly $1-q^{x,0}_y$. We therefore have that:

$$\Pr\left[(\text{VIEW}^i_{\mathsf{ideal}}, \text{OUT}_{\mathsf{ideal}}) = (a,b) \ \mid \ i \leq i^*\right] = \alpha + (1-\alpha)\cdot(1-p_x)\cdot(1-q^{x,0}_y) \ .$$

Similarly, we compute the above for all possible outputs $(a,b) \in \{0,1\}^2$ and obtain:

output (a,b)	real	ideal
$(0,0)$	$(1-\alpha)\cdot(1-p_x)\cdot(1-p_y) + \alpha\cdot(1-p_y)$	$(1-\alpha)\cdot(1-p_x)\cdot(1-q^{x,0}_y) + \alpha$
$(0,1)$	$(1-\alpha)\cdot(1-p_x)\cdot p_y + \alpha\cdot p_y$	$(1-\alpha)\cdot(1-p_x)\cdot q^{x,0}_y$
$(1,0)$	$(1-\alpha)\cdot p_x\cdot(1-p_y)$	$(1-\alpha)\cdot p_x\cdot(1-q^{x,1}_y)$
$(1,1)$	$(1-\alpha)\cdot p_x\cdot p_y$	$(1-\alpha)\cdot p_x\cdot q^{x,1}_y$

Therefore, we get the following constraints:

$$q^{x,0}_y = p_y + \frac{\alpha\cdot p_y}{(1-\alpha)\cdot(1-p_x)} \qquad \text{and} \qquad q^{x,1}_y = p_y \ ,$$

which are satisfied according to our assumption in the theorem.

In case $\boldsymbol{f(x,y)=1}$. This is similar for the previous case. We have:

output (a,b)	real	ideal
$(0,0)$	$(1-\alpha)\cdot(1-p_x)\cdot(1-p_y)$	$(1-\alpha)\cdot(1-p_x)\cdot(1-q^{x,0}_y)$
$(0,1)$	$(1-\alpha)\cdot(1-p_x)\cdot p_y$	$(1-\alpha)\cdot(1-p_x)\cdot q^{x,0}_y$
$(1,0)$	$(1-\alpha)\cdot p_x\cdot(1-p_y) + \alpha\cdot(1-p_y)$	$(1-\alpha)\cdot p_x\cdot(1-q^{x,1}_y)$
$(1,1)$	$(1-\alpha)\cdot p_x\cdot p_y + \alpha\cdot p_y$	$(1-\alpha)\cdot p_x\cdot q^{x,1}_y + \alpha$

we again get the following constraints:

$$q^{x,0}_y = p_y \qquad \text{and} \qquad q^{x,1}_y = p_y + \frac{\alpha\cdot(p_y-1)}{(1-\alpha)\cdot p_x} \ .$$

However, since $X^{x,a}_{ideal}\cdot M_f = Q^{x,a}$, the above constraints are satisfied. ■

An alternative formulation for Theorem 3.4, is to require that for every x, a, the points $Q^{x,a}$ is in $\mathbf{conv}(\{X_1, \ldots, X_\ell\})$, where X_i is the ith row of M_f. Moreover, observe that in order to decide whether a function can be computed using the protocol, there are 2ℓ linear systems that should be satisfied, with m constraints each, and with $2\ell^2$ variables overall. This criterion depends heavily on some parameters of the protocols (like p_{x_i}, p_{y_j}) rather than properties of the function. We are interested in a simpler and easier way to validate criterion.

4 Our Criteria

4.1 Possibility of Full-Dimensional Functions

In this section, we show that any function that defines a full-dimensional geometric object, can be computed using the protocol of [GHKL08]. The formal definition for this notion is as follows:

Definition 4.1 (full-dimensional function) *Let $f : \{x_1, \dots, x_\ell\} \times \{y_1, \dots, y_m\} \to \{0,1\}$ be a function, and let $X_1, \dots, X_\ell$ be the ℓ rows of M_f over $\mathbb{R}^m$. If $\dim(\mathbf{aff}(\{X_1, \dots, X_\ell\})) = m$ then we say that f is* a full-dimensional function.

Recall that for a set of points $S = \{X_1, \dots, X_\ell\} \in \mathbb{R}^m$, if $\dim(\mathbf{aff}(S)) = m$ then the convex-hull of the points defines a full-dimensional convex polytope. Thus, intuitively, the simulator has enough power to simulate the protocol. Recall that a basis for an affine space of dimension m has cardinality $m+1$, and thus we must have that $\ell > m$. Thus, we assume without loss of generality that this holds (and consider the transposed function $f^T : \{y_1, \dots, y_m\} \times \{x_1, \dots, x_\ell\} \to \{0,1\}$, defined as $f^T(y,x) = f(x,y)$, otherwise). Overall, our property inherently holds only if $\ell \neq m$.

4.1.1 Alternative Representation

Before we prove that any full-dimensional function can be computed fairly, we give a different representation for this definition. This strengthens our understanding of this property, and is also related to the balanced property defined in [ALR13] (see more in Section 4.3). We have:

Claim 4.2 *Let $f : \{x_1, \dots, x_\ell\} \times \{y_1, \dots, y_m\} \to \{0,1\}$ be a function, let M_f be as above and let $S = \{X_1, \dots, X_\ell\}$ be the rows of M_f (ℓ points in $\mathbb{R}^m$). The following are equivalent:*

1. The function is right-unbalanced with respect to arbitrary vectors.
   That is, for every non-zero $\mathbf{q} \in \mathbb{R}^m$ and any $\delta \in \mathbb{R}$ it holds that: $M_f \cdot \mathbf{q}^T \neq \delta \cdot \mathbf{1}_\ell$.
2. The rows of the matrix do not lie on the same hyperplane.
   That is, for every non-zero $\mathbf{q} \in \mathbb{R}^m$ and any $\delta \in \mathbb{R}$, there exists a point X_i such that $X_i \notin \mathcal{H}(\mathbf{q}, \delta)$. Alteratively, $\mathbf{conv}(\{X_1, \dots, X_\ell\}) \not\subseteq \mathcal{H}(\mathbf{q}, \delta)$.
3. The function is full-dimensional.
   There exists a subset of $\{X_1, \dots, X_\ell\}$ of cardinality $m+1$, that is affinely independent. *Thus, $\dim(\mathbf{aff}(\{X_1, \dots, X_\ell\})) = m$.*

Proof:
$\underline{\neg 1 \Leftrightarrow \neg 2}$: By contradiction, if there exists $\mathbf{q}, \delta$ such that $M_f \cdot \mathbf{q}^T = \delta \cdot \mathbf{1}_\ell$, then for every row of the matrix M_f it holds that $\langle X_i, \mathbf{q} \rangle = \delta$ and so $X_i \in \mathcal{H}(\mathbf{q}, \delta)$. This also implies that for any point in $\mathbf{conv}(\{X_1, \dots, X_\ell\})$, the point is in $\mathcal{H}(\mathbf{q}, \delta)$. This is because any point in $\mathbf{conv}(\{X_1, \dots, X_\ell\})$ can be represented as $\mathbf{a} \cdot M_f$ where $\mathbf{a}$ is a probability vector, and thus we have that

$$\langle \mathbf{a} \cdot M_f, \mathbf{q} \rangle = \mathbf{a} \cdot M_f \cdot \mathbf{q}^T = \mathbf{a} \cdot \delta \cdot \mathbf{1}_\ell = \delta \ ,$$

since $\mathbf{a}$ is a probability vector and sum-up to 1. We therefore have that $\mathbf{conv}(\{X_1, \dots, X_\ell\}) \subseteq \mathcal{H}(\mathbf{q}, \delta)$. For the reverse direction, if such $\mathcal{H}(\mathbf{q}, \delta)$ exists that contains all the rows of the matrix, then clearly $M_f \cdot \mathbf{q}^T = \delta \cdot \mathbf{1}_\ell^T$ in contradiction.

$\underline{3 \Rightarrow 2}$: Since the affine dimension of $\{X_1, \ldots, X_\ell\}$ is m, then it cannot lie in a single hyperplane (an affine subspace of dimension $m-1$). Thus, this implication trivially holds.

$\underline{2 \Rightarrow 3}$: We first claim that there exists a subset of $m+1$ points $S' = \{\hat{X}_1, \ldots, \hat{X}_{m+1}\}$ that does not lie on the same hyperplane, for any hyperplane. This set can be found iteratively, where we start with a single point, and add some other distinct point (such must exist otherwise we found an hyperplane that contains all the points). Then, we look for another point that do not lie on the same line that is defined by the 2 points that we have (again, such must exist from the same reason as above). We then look for a forth points that do not lie on the same plane that contains all the three points that we have, and so on. At the end of this process, we get the set $S' = \{\hat{X}_1, \ldots, \hat{X}_{m+1}\}$, which are $m+1$ points that do not lie on the same hyperplane, for any hyperplane. We now claim that S' is affinely independent. Take the first m points, which define some hyperplane $\mathcal{H}(\mathbf{q}, \delta)$. Since $X_{m+1} \notin \mathcal{H}(\mathbf{q}, \delta)$ and since $\mathcal{H}(\mathbf{q}, \delta) = \mathbf{aff}(\{X_1, \ldots, X_m\})$ then $X_{m+1} \notin \mathbf{aff}(\{X_1, \ldots, X_m\})$, and therefore the points $X_1, \ldots, X_{m+1}$ are affine independent.

■

From Alternative 1, checking whether a function is full-dimensional can be done very efficiently. Giving that $\ell > m$, all we have to do is to verify that the only possible solution $\mathbf{q}$ to the linear system $M_f \cdot \mathbf{q}^T = \mathbf{0}_\ell^T$ is the trivial one, and the there is no solution $\mathbf{q}$ to the linear system $M_f \cdot \mathbf{q}^T = \mathbf{1}_\ell^T$. This implies that the function is unbalanced for every $\delta \in \mathbb{R}$.

4.1.2 The Proof of Possibility

We now show that any function that is full-dimensional can be computed with complete fairness, using the protocol of [GHKL08]. The proof for this Theorem is geometrical. Recall that by Theorem 3.4, we need to show that there exists a solution for some set of equations. In our proof here, we show that such a solution exists without solving the equations explicitly. We show that all the points $Q^{x,a}$ that the simulator needs (by Theorem 3.4) are in the convex-hull of the rows $\{X_1, \ldots, X_\ell\}$, and therefore there exist probability vectors $X_{ideal}^{x,a}$ as required. We show this in two steps. First, we show that all the points are very "close" to some point $\mathbf{c}$, and therefore, all the points are inside the Euclidian ball centered at $\mathbf{c}$ for some small radius ϵ (defined as $B(\mathbf{c}, \epsilon) \stackrel{\text{def}}{=} \{Z \in \mathbb{R}^m \mid d(Z, \mathbf{c}) \leq \epsilon\}$). Second, we show that this whole ball is embedded inside the convex-polytope that is defined by the rows of the function, which implies that all the points $Q^{x,a}$ are in the convex-hull and simulation is possible.

In more detail, fix some distribution X_{real} for which the point $\mathbf{c} = (p_{y_1}, \ldots, p_{y_m}) = X_{real} \cdot M_f$ is inside the convex-hull of the matrix. Then, we observe that by adjusting α, all the points $Q^{x,a}$ that we need are very "close" to this point $\mathbf{c}$. This is because each coordinate $q_{y_j}^{x,a}$ is exactly p_{y_j} plus some term that is multiplied by $\alpha/(1-\alpha)$, and therefore we can control its distance from p_{y_j} (see Eq. (1)). In particular, if we choose $\alpha = 1/\ln\kappa$, then for all sufficiently large κ's the distance between $Q^{x,a}$ and $\mathbf{c}$ is smaller than any constant. Still, for $\alpha = 1/\ln\kappa$, the number of rounds of the protocol is $R = \alpha^{-1} \cdot \omega(\ln\kappa) = \ln\kappa \cdot \omega(\ln\kappa)$, and thus asymptotically remains unchanged.

All the points $Q^{x,a}$ are close to the point $\mathbf{c}$. This implies that they all lie in the m-dimensional Euclidian ball of some constant radius $\epsilon > 0$ centered at $\mathbf{c}$. Moreover, since the function is of full-dimension, the convex-hull of the function defines a full-dimensional convex polytope, and therefore this ball is embedded in this polytope. We prove this by showing that the center of the ball $\mathbf{c}$ is "far" from each facet of the polytope, using the separation theorems of closed convex sets. As a result, all the points that are "close" to $\mathbf{c}$ (i.e., our ball) are still "far" from each facet of the polytope, and thus they are inside it. As an illustration, consider again the case of the GHKL function in Figure 2, in the Introduction (where here, the circle represents the ball and the center of the circle is the point $\mathbf{c}$). We conclude that all the points that the simulator needs are in the convex-hull of the function, and therefore the protocol can be simulated.

Before we proceed to the full proof formally, we give an additional definition and an important Claim that will be helpful for our proof. For a set $F \subseteq \mathbb{R}^m$ and a point $\mathbf{p} \in \mathbb{R}^m$, we define the distance between $\mathbf{p}$ and F to be the minimal distance between $\mathbf{p}$ and a point in F, that is: $d(\mathbf{p}, F) = \min\{d(\mathbf{p}, \mathbf{f}) \mid \mathbf{f} \in F\}$. The following claim states that if a point is not on a closed convex set, then there exists a constant distance between the point and the convex set. We use this claim to show that the point $\mathbf{c}$ is far enough from each one of the facets of the polytope (and therefore the ball centered in $\mathbf{c}$ is in the convex). This Claim is a simple implication of the separation theorems for convex sets, see [Rom08]. We have:

Claim 4.3 *Let $\mathcal{C}$ be a closed convex subset of $\mathbb{R}^m$, and let $\mathbf{a} \in \mathbb{R}^m$ such that $\mathbf{a} \notin \mathcal{C}$. Then, there exists a constant $\epsilon > 0$ such that $d(\mathbf{a}, \mathcal{C}) > \epsilon$ (that is, for every $Z \in \mathcal{C}$ it holds that $d(\mathbf{a}, Z) > \epsilon$).*

We now ready for our main theorem of this section:

Theorem 4.4 *Let $f : \{x_1, \ldots, x_\ell\} \times \{y_1, \ldots, y_m\} \to \{0,1\}$ be a Boolean function. If f is of full-dimension, then f can be computed with complete fairness.*

Proof: Since f is full-dimensional, there exists a subset of $m+1$ rows that are affinely independent. Let $S' = \{X_1, \ldots, X_{m+1}\}$ be this subset of rows. We now choose parameters for the GHKL protocol, such that $\mathbf{c}$ will be inside the simplex that is defined by S'. For this, we can simply define X_{real} to be the uniform distribution over S' (the ith position of X_{real} is 0 if $X_i \notin S'$, and $1/(m+1)$ if $X_i \in S'$). We also set $\alpha = 1/\ln \kappa$, and this all the parameters that are needed for the real protocol.

Let $\mathbf{c} = (p_{y_1}, \ldots, p_{y_m}) = X_{real} \cdot M_f$, the output distribution vector that is correspond to X_{real}. Consider the set of points $\{Q^{x,a}\}_{x \in X, a \in \{0,1\}}$ that are needed for the simulation as in Eq. (1). The next claim shows that all these points are close to $\mathbf{c}$, and in the m-dimensional ball $B(\mathbf{c}, \epsilon)$ for some small $\epsilon > 0$. That is:

Claim 4.5 *For every constant $\epsilon > 0$, for every $x \in X, a \in \{0,1\}$, and for all sufficiently large κ's it holds that:*

$$Q^{x,a} \in B(\mathbf{c}, \epsilon)$$

Proof: Fix ϵ. Since $\alpha = 1/\ln \kappa$, for every constant $\delta > 0$ and for all sufficiently large κ's it holds that: $\alpha/(1-\alpha) < \delta$. We show that for every x, a, it holds that $d(Q^{x,a}, \mathbf{c}) \leq \epsilon$, and thus $Q^{x,a} \in B(\mathbf{c}, \epsilon)$.

Recall the definition of $Q^{x,0}$ as in Eq. (1): If $f(x, y_j) = 1$ then $q^0_{y_j} = p_{y_j}$ and thus $|p_{y_j} - q^0_{y_j}| = 0$. In case $f(x, y_j) = 1$, for $\delta = \epsilon(1-p_x)/\sqrt{m}$ and for all sufficiently large κ's it holds that:

$$\left|p_{y_j} - q^{x,0}_{y_j}\right| = \left|p_{y_j} - p_{y_j} - \frac{\alpha}{1-\alpha} \cdot \frac{p_{y_j}}{(1-p_x)}\right| \leq \frac{\alpha}{1-\alpha} \cdot \frac{1}{(1-p_x)} \leq \frac{\delta}{(1-p_x)} = \frac{\epsilon}{\sqrt{m}} .$$

Therefore, for all sufficiently large κ's, $\left|p_{y_j} - q^{x,0}_{y_j}\right| \leq \epsilon/\sqrt{m}$ irrespectively to whether $f(x, y_j)$ is 1 or 0. Similarly, for all sufficiently large κ's it holds that: $\left|p_{y_j} - q^{x,1}_{y_j}\right| \leq \epsilon/\sqrt{m}$. Overall, for every

$x \in X$, $a \in \{0,1\}$ we have that the distance between the points $Q^{x,a}$ and $\mathbf{c}$ is:

$$d(Q^{x,a}, \mathbf{c}) = \sqrt{\sum_{j=1}^{m} \left(q_{y_j}^{x,b} - p_{y_j} \right)^2} \leq \sqrt{\sum_{j=1}^{m} \left(\frac{\epsilon}{\sqrt{m}} \right)^2} \leq \epsilon$$

and therefore $Q^{x,a} \in B(\mathbf{c}, \epsilon)$. ■

We now show that this whole ball is embedded inside the simplex of S'. That is:

Claim 4.6 *There exists a constant $\epsilon > 0$ for which $B(\mathbf{c}, \epsilon) \subset \mathbf{conv}(S')$.*

Proof: Since $S' = \{X_1, \ldots, X_{m+1}\}$ is affinely independent set of cardinality $m+1$, $\mathbf{conv}(S')$ is a simplex. Recall that $\mathbf{c}$ is a point in the simplex (since it assigns 0 to any row that is not in S'), and so $\mathbf{c} \in \mathbf{conv}(S')$. We now show that for every *facet* of the simplex, there exists a constant distance between the point $\mathbf{c}$ and the facet. Therefore, there exists a small ball around $\mathbf{c}$ that is "far" from each facet of the simplex, and inside the simplex.

For every $1 \leq i \leq m+1$, the ith facet of the simplex is the set $F_i = \mathbf{conv}(S' \setminus \{X_i\})$, i.e., the convex set of the vertices of the simplex without the vertex X_i. We now show that $\mathbf{c} \notin F_i$, and therefore, using Claim 4.3, $\mathbf{c}$ is ϵ-far from F_i, for some small $\epsilon > 0$.

In order to show that $\mathbf{c} \notin F_i$, we show that $\mathbf{c} \notin \mathcal{H}(\mathbf{q}, \delta)$, where $\mathcal{H}(\mathbf{q}, \delta)$ is an hyperplane that contains F_i. That is, let $\mathcal{H}(\mathbf{q}, \delta)$ be the unique hyperplane that contains all the points $S' \setminus \{X_i\}$ (these are m affinely independent points and therefore there is a unique hyperplane that contains all of them). Recall that $X_i \notin \mathcal{H}(\mathbf{q}, \delta)$ (otherwise, S' is affinely dependent). Observe that $F_i = \mathbf{conv}(S' \setminus \{X_i\}) \subset \mathcal{H}(\mathbf{q}, \delta)$, since each point X_i is in the hyperplane, and the hyperplane is an affine set. We now show that since $X_i \notin \mathcal{H}(\mathbf{q}, \delta)$, then $\mathbf{c} \notin \mathcal{H}(\mathbf{q}, \delta)$ and therefore $\mathbf{c} \notin F_i$.

Assume by contradiction that $\mathbf{c} \in \mathcal{H}(\mathbf{q}, \delta)$. We can write:

$$\delta = \langle \mathbf{c}, \mathbf{q} \rangle = \left\langle \sum_{j=1}^{m+1} \frac{1}{m+1} \cdot X_j, \mathbf{q} \right\rangle = \frac{1}{m+1} \langle X_i, \mathbf{q} \rangle + \frac{1}{m+1} \sum_{j \neq i} \langle X_j, \mathbf{q} \rangle = \frac{1}{m+1} \langle X_i, \mathbf{q} \rangle + \frac{m}{m+1} \cdot \delta$$

and so, $\langle X_i, \mathbf{q} \rangle = \delta$, which implies that $X_i \in \mathcal{H}(\mathbf{q}, \delta)$ in contradiction.

Since $\mathbf{c} \notin F_i$, and since F_i is a closed[3] convex, we can apply Claim 4.3 to get the existence of a constant $\epsilon_i > 0$ such that $d(\mathbf{c}, F_i) > \epsilon_i$.

Now, let $F_1, \ldots, F_{m+1}$ be the facets of the simplex. We get the existence of $\epsilon_1, \ldots, \epsilon_{m+1}$ for each facet as above. Let $\epsilon = \min\{\epsilon_1, \ldots, \epsilon_{m+1}\}/2$, and so for every i, we have: $d(\mathbf{c}, F_i) > 2\epsilon$.

Consider the ball $B(\mathbf{c}, \epsilon)$. We show that any point in this ball is of distance at least ϵ from each facet F_i. Formally, for every $\mathbf{b} \in B(\mathbf{c}, \epsilon)$, for every facet F_i it holds that: $d(\mathbf{b}, F_i) > \epsilon$. This can be easily derived from the triangle inequality, where for every $\mathbf{b} \in B(\mathbf{c}, \epsilon/2)$:

$$d(\mathbf{c}, \mathbf{b}) + d(\mathbf{b}, F_i) \geq d(\mathbf{c}, F_i) > 2\epsilon \ ,$$

and so $d(\mathbf{b}, F_i) > \epsilon$ since $d(\mathbf{b}, \mathbf{c}) \leq \epsilon$.

Overall, all the points $\mathbf{b} \in B(\mathbf{c}, \epsilon)$ are of distance at least ϵ from each facet of the simplex, and inside the simplex. This shows that $B(\mathbf{c}, \epsilon) \subset \mathbf{conv}(S')$. ■

[3]The convex-hull of a finite set S of vectors in $\mathbb{R}^m$ is a compact set, and therefore is closed (See [Rom08, Theorem 15.4]).

For conclusion, there exists a constant $\epsilon > 0$ for which $B(\mathbf{c}, \epsilon) \subset \mathbf{conv}(S') \subseteq \mathbf{conv}(\{X_1, \ldots, X_\ell\})$. Moreover, for all $x \in X, a \in \{0,1\}$ and for all sufficiently large κ's, it holds that $Q^{x,a} \in B(\mathbf{c}, \epsilon)$. Therefore, the requirements of Theorem 3.4 are satisfied, and the protocol securely computes f with complete fairness. ■

4.1.3 On the Number of Full-Dimensional Functions

We count the number of functions that are full dimensional. Recall that a function with $|X| = |Y|$ cannot be full-dimensional, and we consider only functions where $|X| \neq |Y|$. Interestingly, the probability that a random function with distinct domain sizes is full-dimensional tends to 1 when $|X|, |Y|$ grow. Thus, almost always, a random function with distinct domain sizes can be computed with complete fairness(!). The answer for the frequency of full-dimensional functions within the class of Boolean functions with distinct sizes relates to a beautiful problem in combinatorics and linear algebra, that has received careful attention: Estimating the probability that a random Boolean matrix of size $m \times m$ is singular. Denote this probability by P_m. The answer for our question is $1 - P_m$, and is even larger when the difference between $|X|$ and $|Y|$ increases (see Claim 4.7 below).

The value of P_m is conjectured to be $(1/2 + o(1))^m$. Recent results [Kom67, JKS95, Woo09] are getting closer to this conjuncture by showing that $P_m \leq (1/\sqrt{2} + o(1))^m$, which is roughly the probability to have two identical or compliments rows or columns. Observe that this is a negligible function. Since our results hold only for the case of *finite* domain, it is remarkable to address that P_m is small already for very small dimensions m. For instance, $P_{10} < 0.29$, $P_{15} < 0.047$ and $P_{30} < 1.6 \cdot 10^{-6}$ (and so $> 99.999\%$ of the 31×30 functions can be computed fairly). See more experimental results in [VZ06]. The following Claim is based on [Zie00, Corollary 14]:

Claim 4.7 *With a probability that tends to* 1 *when* $|X|, |Y|$ *grow, a random function with* $|X| \neq |Y|$ *is full-dimensional.*

Proof: Our question is equivalent to the following: What is the probability that the convex-hull of $m + 1$ (or even more) random 0/1-points in $\mathbb{R}^m$ is an m-dimensional simplex?

Recall that P_m denotes the probability that a random m vectors of size m are linearly dependent. Then, the probability for our question is simply $1 - P_m$. This is because with very high probability our $m + 1$ points will be distinct, we can choose the first point X_1 arbitrarily, and the rest of the points $S = \{X_2, \ldots, X_{m+1}\}$ uniformly at random. With probability $1 - P_m$, the set S is linearly independent, and so it linearly spans X_1. It is easy to see that this implies that $\{X_2 - X_1, \ldots, X_{m+1} - X_1\}$ is a linearly independent set, and thus $\{X_1, \ldots, X_{m+1}\}$ is affinely-independent set. Overall, a random set $\{X_1, \ldots, X_{m+1}\}$ is affinely independent with probability $1 - P_m$. ■

4.2 Functions that Are Not Full-Dimensional

4.2.1 A Negative Result

We now consider the case where the functions are not full-dimensional. This includes the limited number of functions for which $|X| \neq |Y|$, and *all* functions with $|X| = |Y|$. In particular, for a function that is not full-dimensional, all the rows of the function lie in some hyperplane (a $(m-1)$-dimensional subspace of $\mathbb{R}^m$), and all the columns of the matrix lie in a different hyperplane (in $\mathbb{R}^\ell$). We show that under additional requirements, the protocol of [GHKL08] cannot be simulated for any choice of parameters, with respect to the specific simulation strategy defined in the proof of Theorem 3.4.

We have the following Theorem:

Theorem 4.8 *Let $f, M_f, \{X_1, \ldots, X_\ell\}$ be as above, and let $\{Y_1, \ldots, Y_m\}$ be the columns of M_f. Assume that the function is not full-dimensional, that is, there exist non-zero $\mathbf{p} \in \mathbb{R}^\ell$, $\mathbf{q} \in \mathbb{R}^m$ and some $\delta_1, \delta_2 \in \mathbb{R}$ such that:*

$$X_1, \ldots, X_\ell \in \mathcal{H}(\mathbf{q}, \delta_2) \qquad \text{and} \qquad Y_1, \ldots, Y_m \in \mathcal{H}(\mathbf{p}, \delta_1) \ .$$

Assume that in addition, $\mathbf{0}_\ell, \mathbf{1}_\ell \notin \mathcal{H}(\mathbf{p}, \delta_1)$ and $\mathbf{0}_m, \mathbf{1}_m \notin H(\mathbf{q}, \delta_2)$. Then, the function f cannot be computed using the GHKL protocol, for any choice of parameters (α, X_{real}), with respect to the specific simulation strategy used in Theorem 3.4.

Proof: We first consider the protocol where P_1 plays the party that inputs $x \in X$ and P_2 inputs $y \in Y$ (that is, P_2 is the second to receive output, exactly as GHKL protocol is described in Section 3). Fix any X_{real}, α, and let $\mathbf{c} = (p_{y_1}, \ldots, p_{y_m}) = X_{real} \cdot M_f$. First, observe that $\mathbf{conv}(\{X_1, \ldots, X_\ell\}) \subseteq \mathcal{H}(\mathbf{q}, \delta_2)$, since for any point $Z \in \mathbf{conv}(\{X_1, \ldots, X_\ell\})$, we can represent Z as $\mathbf{a} \cdot M_f$ for some probability vector $\mathbf{a}$. Then, we have that $\langle Z, \mathbf{q}\rangle = \langle \mathbf{a} \cdot M_f, \mathbf{q}\rangle = \mathbf{a} \cdot \delta_2 \cdot \mathbf{1}_\ell = \delta_2$ and so $Z \in \mathcal{H}(\mathbf{q}, \delta_2)$. Now, assume by contradiction that the set of equations is satisfied. This implies that $Q^{x,a} \in \mathcal{H}(\mathbf{q}, \delta_2)$ for every $x \in X$, $a \in \{0, 1\}$, since $Q^{x,a} \in \mathbf{conv}(\{X_1, \ldots, X_\ell\}) \subseteq \mathcal{H}(\mathbf{q}, \delta_2)$.

Let $\circ$ denote the entrywise product over $\mathbb{R}^m$, that is for $Z = (z_1, \ldots, z_m)$, $W = (w_1, \ldots, w_m)$, the point $Z \circ W$ is defined as $(z_1 \cdot w_1, \ldots, z_m \cdot w_m)$. Recall that $\mathbf{c} = (p_{y_1}, \ldots, p_{y_m})$. We claim that for every X_i, the point $\mathbf{c} \circ X_i$ is also in the hyperplane $\mathcal{H}(\mathbf{q}, \delta_2)$. This trivially holds if $X_i = \mathbf{1}_m$. Otherwise, recall the definition of $Q^{x_i,0}$ (Eq. (1)):

$$q^{x_i,0}_{y_j} \overset{\text{def}}{=} \begin{cases} p_{y_j} & \text{if } f(x_i, y_j) = 1 \\ p_{y_j} + \frac{\alpha \cdot p_{y_j}}{(1-\alpha)\cdot(1-p_{x_i})} & \text{if } f(x_i, y_j) = 0 \end{cases} ,$$

Since $X_i \neq \mathbf{1}_m$, it holds that $p_{x_i} \neq 1$. Let $\gamma = \frac{\alpha}{(1-\alpha)\cdot(1-p_{x_i})}$. We can write $Q^{x,0}$ as follows:

$$Q^{x,0} = (1 + \gamma) \cdot \mathbf{c} - \gamma \cdot (\mathbf{c} \circ X_i) \ .$$

Since for every i, the point $Q^{x_i,0}$ is in the hyperplane $\mathcal{H}(\mathbf{q}, \delta_2)$, we have:

$$\delta_2 = \langle Q^{x,0}, \mathbf{q}\rangle = \langle (1+\gamma)\cdot\mathbf{c} - \gamma\cdot(\mathbf{c}\circ X_i), \mathbf{q}\rangle = (1+\gamma)\cdot\langle \mathbf{c}, \mathbf{q}\rangle - \gamma\cdot\langle \mathbf{c}\circ X_i, \mathbf{q}\rangle = (1+\gamma)\cdot\delta_2 - \gamma\cdot\langle \mathbf{c}\circ X_i, \mathbf{q}\rangle$$

and thus, $\langle \mathbf{c} \circ X_i, \mathbf{q}\rangle = \delta_2$ which implies that $\mathbf{c} \circ X_i \in \mathcal{H}(\mathbf{q}, \delta_2)$.

We conclude that all the points $(\mathbf{c} \circ X_1), \ldots, (\mathbf{c} \circ X_\ell)$ are in the hyperplane $\mathcal{H}(\mathbf{q}, \delta_2)$. Since all the points $Y_1, \ldots, Y_m$ are in $\mathcal{H}(\mathbf{p}, \delta_1)$, it holds that $\mathbf{p} \cdot M_f = \delta_1 \cdot \mathbf{1}_m$. Thus, $\sum_{i=1}^{\ell} p_i \cdot X_i = \delta_1 \cdot \mathbf{1}_m$, which implies that:

$$\begin{aligned}\sum_{i=1}^{\ell} p_i \cdot \delta_2 &= \sum_{i=1}^{\ell} p_i \cdot \Big\langle \mathbf{c} \circ X_i, \mathbf{q} \Big\rangle = \Big\langle \sum_{i=1}^{\ell} p_i \cdot (\mathbf{c} \circ X_i), \mathbf{q} \Big\rangle = \Big\langle \mathbf{c} \circ (\sum_{i=1}^{\ell} p_i \cdot X_i), \mathbf{q} \Big\rangle = \langle \mathbf{c} \circ (\delta_1 \cdot \mathbf{1}_m), \mathbf{q}\rangle \\ &= \delta_1 \cdot \langle \mathbf{c}, \mathbf{q}\rangle = \delta_1 \cdot \delta_2\end{aligned}$$

and thus it must hold that either $\sum_{i=1}^{\ell} p_i = \delta_1$ or $\delta_2 = 0$, which implies that $\mathbf{1} \in \mathcal{H}(\mathbf{p}, \delta_1)$ or $\mathbf{0} \in \mathcal{H}(\mathbf{q}, \delta_2)$, in contradiction to the additional requirements.

The above shows that the protocol does not hold when the P_1 party is the first to receive output. We can change the roles and let P_2 to be the first to receive an output (that is, we can use the protocol to compute f^T). In such a case, we will get that it must hold that $\sum_{i=1}^{m} q_i = \delta_2$ or $\delta_1 = 0$, again, in contradiction to the assumptions that $\mathbf{1} \notin \mathcal{H}(\mathbf{q}, \delta_2)$ and $\mathbf{0} \notin \mathcal{H}(\mathbf{p}, \delta_1)$. ■

This negative result does not rule out the possibility of these functions using some other protocol. However, it rules out the only known possibility result that we have in fairness. Moreover, incorporating this with the characterization of coin-tossing [ALR13], there exists a large set of functions for which the only possibility result does not hold, and the only impossibility result does not hold either. Moreover, this class of functions shares similar (but yet distinct) algebraic structure with the class of functions that imply fair coin-tossing. See more in Subsection 4.3.

A possible generalization of [GHKL08]. In Section 3, we generalized the protocol such that the algorithm $\mathsf{RandOut}_2(y)$ chooses $\hat{x}$ according to some distribution X_{real} and not just uniformly at random as in [GHKL08]. Similarly, we could have generalized the protocol for the $\mathsf{RandOut}_1(x)$ algorithm as well, while defining some distribution Y_{real}, and parameterized the protocol with this additional distribution (in the same sense as we parameterized X_{real}). By using this modification, the protocol may potentially compute more functions. However, by a simple adjustment of the proof of Theorem 4.8, as long as Y_{real} is "valid" (in a sense that $p_{x_i} = 1$ if and only if $X_i = \mathbf{1}_m$ and $p_{x_i} = 0$ if and only if $X_i = \mathbf{0}_m$, otherwise the output distribution vectors $Q^{x,a}$ are not well-defined), this generalization does not help, and the negative result holds for this generalization also.

Our theorem does not hold in cases where either $\mathbf{0}_\ell \in \mathcal{H}(\mathbf{p}, \delta_1)$ or $\mathbf{1}_\ell \in \mathcal{H}(\mathbf{p}, \delta_1)$ (likewise, for $\mathcal{H}(\mathbf{q}, \delta_2)$). These two requirements are in some sense equivalent. This is because the alphabet is not significant, and we can switch between the two symbols 0 and 1. Thus, if for some function f the hyperplane $\mathcal{H}(\mathbf{p}, \delta_1)$ passes through the origin $\mathbf{0}$, the corresponding hyperplane for the function $\bar{f}(x,y) = 1 - f(x,y)$ passes through $\mathbf{1}$ and vice versa. Feasibility of fairness for f and $\bar{f}$ is equivalent.

4.2.2 On the Number of Functions that Satisfy the Additional Requirements

We now count on the number of functions with $|X| = |Y|$ that satisfy these additional requirements, that is, define hyperplanes that do not pass through the origin $\mathbf{0}$ and the point $\mathbf{1}$. As we have seen in Theorem 4.8, these functions cannot be computed with complete fairness using the protocol of [GHKL08]. As we will see, only negligible amount of functions with $|X| = |Y|$ do not satisfy these additional requirements. Thus, our characterization of [GHKL08] is almost tight: Almost all functions with $|X| \neq |Y|$ can be computed fairly, whereas almost all functions with $|X| = |Y|$ cannot be computed using the protocol of [GHKL08]. We have the following Claim:

Claim 4.9 *With a probability that tends to* 0 *when* $|X|, |Y|$ *grow, a random function with* $|X| = |Y|$ *define hyperplanes that pass through the points* $\mathbf{0}$ *or* $\mathbf{1}$*.*

Proof: Let $m = |X| = |Y|$. Recall that P_m denotes the probability that a random m vectors of size m are linearly dependent. Moreover, by Claim 4.7, the probability that a random set $\{X_1, \ldots, X_{m+1}\}$ is affinely independent with probability $1 - P_m$, even when one of the points is chosen arbitrarily.

Thus, with probability P_m, the set $\{X_1, \ldots, X_m, \mathbf{1}\}$ where $X_1, \ldots, X_m$ are chosen at random is affinely dependent. In this case, the hyperplane defined by $\{X_1, \ldots, X_m\}$ contains the point $\mathbf{1}$. Similarly, the set $\{X_1, \ldots, X_m, \mathbf{0}\}$ is affienely dependent with the same probability P_m. Overall, using union-bound, the probability that the hyperplane of random points $X_1, \ldots, X_m$ contains the points $\mathbf{1}$ or $\mathbf{0}$ is less than equal to $2 \cdot P_m$. From similar arguments, the probability that the hyperplane that is defined by the columns of the matrix contains either $\mathbf{1}$ or $\mathbf{0}$ is $2 \cdot P_m$, and therefore the overall probability is $4 \cdot P_m$.

■

4.2.3 Functions with Monochromatic Input

We consider a limited case where the above requirements do not satisfy, that is, functions that are not full-dimensional but define hyperplanes that pass through $\mathbf{0}$ or $\mathbf{1}$. For this set of functions, the negative result does not apply. We now show that for some subset in this class, fairness is possible. Our result here does not cover all functions in this subclass.

Assume that a function contains a "monochromatic input", that is, one party has an input that causes the same output irrespectively to the input of the other party. For instance, P_2 has input y_j such that for every $x \in X$: $f(x, y_j) = 1$. In this case, the point $\mathbf{1}_\ell$ is one of the columns of the matrix, and therefore, the hyperplane $\mathcal{H}(\mathbf{p}, \delta_1)$ must pass through it. We show that in this case we can ignore this input and consider the "projected" $m \times (m-1)$ function f' where we remove the input y_j. This latter function may now be full-dimensional, and the existence of a protocol for f' implies the existence of a protocol for f. Intuitively, this is because when P_2 uses y_j, the real-world adversary P_1 cannot bias its output since it is always 1. We have:

Claim 4.10 *Let $f : X \times Y \to \{0,1\}$, and assume that M_f contains the all-one (resp. all-zero) column. That is, there exists $y \in Y$ such that for every $\hat{x} \in X$, $f(\hat{x}, y) = 1$ (resp. $f(\hat{x}, y) = 0$).*

If the function $f' : X \times Y' \to \{0,1\}$, where $Y' = Y \setminus \{y\}$ is full-dimensional, then f can be computed with complete-fairness.

Proof: Assume that the function contains the all one column, and that it is obtained by input y_m (i.e., the mth column is the all-one column). Let $X_1, \ldots, X_m$ be the rows of M_f, and let X'_i be the rows over $\mathbb{R}^{m-1}$ without the last coordinate, that is, $X_i = (X'_i, 1)$. Consider the "projected" function $f' : \{x_1, \ldots, x_m\} \times \{y_1, \ldots, y_{m-1}\} \to \{0,1\}$ be defined as $f'(x, y) = f(x, y)$, for every x, y in the range (we just remove y_m from the possible inputs of P_2). The rows of $M_{f'}$ are $X'_1, \ldots, X'_m$.

Now, since f' is of full-dimensional, the function f' can be computed using the GHKL protocol. Let $X^{x,a}_{ideal}$ be the solutions for equations of Theorem 3.4 for the function f'. It can be easily verified that $X^{x,a}_{ideal}$ are the solutions for equations for the f function as well, since for every x, a, the first $m-1$ coordinates of $Q^{x,a}$ are the same as f', and the last coordinate of $Q^{x,a}$ is always 1. For $Q^{x,0}$ it holds immediately, for $Q^{x,1}$ observe that $p_{y_m} = 1$ no matter what X_{real} is, and thus $p_{y_j} + \frac{\alpha \cdot (p_{y_j} - 1)}{(1-\alpha) \cdot p_x} = 1 + 0 = 1$). Therefore, $X^{x,a}_{ideal}$ are the solutions for f as well, and Theorem 3.4 follows for f as well. ■

The above implies an interesting an easy to verify criterion:

Proposition 4.11 *Let $f : \{x_1, \ldots, x_m\} \times \{y_1, \ldots, y_m\} \to \{0,1\}$ be a function. Assume that f contains the all-one column, and that M_f is of full rank. Then, the function f can be computed with complete fairness.*

Proof: Let $X_1, \ldots, X_m$ be the rows of M_f, and assume that the all-one column is the last one (i.e., input y_m). Consider the points $X'_1, \ldots, X'_m$ in $\mathbb{R}^{m-1}$, where for every i, $X_i = (X'_i, 1)$ (i.e., X'_i is the first $m-1$ coordinates of X_i). Since M_f is of full-rank, the rows $X_1, \ldots, X_m$ are linearly independent, which implies that m points $X'_1, \ldots, X'_m$ in $\mathbb{R}^{m-1}$ are affinely independent. We therefore can apply Claim 4.10 and fairness in f is possible. ■

Finally, from simple symmetric properties, almost *always* a random matrix that contains the all one row / vector is of full rank, in the sense that we have seen in Claims 4.7 and 4.9. Therefore, almost always a random function that contains a monochromatic input can be computed with complete fairness.

Functions with no embedded XOR. [GHKL08] presents a totally different and simpler protocol for handling functions that do not contain an embedded XOR (i.e., the Boolean AND/OR functions, and the greater-than function). An immediate Corollary from Proposition 4.11 is that *all* the functions that can be computed using this simpler protocol, can also be computed using the second generalized protocol. This is because all functions that do not contain an embedded XOR (or their complement function) have both full-rank and monochromatic input, and therefore can be computed using the generalized protocol by Proposition 4.11. This shows that indeed, the only known possibility result that we have in fairness is Protocol 3.3.

However, this first totally different protocol gives an answer for an interesting question regarding the round-complexity of fair protocols. In particular, [GHKL08] gives a lower bound of $\omega(\log \kappa)$ for the round complexity of protocol for computing functions that contain embedded XOR. The simpler protocol for handling functions with no embedded XOR has round complexity that is linear in $|X|, |Y|$ and is *independent* of the security parameter κ.

4.3 Conclusion: Symmetric Boolean Functions with Finite Domain

We overview all the known results in complete fairness for symmetric Boolean functions with finite domain, and we link our results to the balanced property of [ALR13]. Moreover, we give few examples for functions that can be computed with complete fairness, that are part of this class of functions.

Characterization of coin-tossing [ALR13]. The work of Asharov, Lindell and Rabin [ALR13] considers the task of coin-tossing, which was shown to be impossible to compute fairly [Cle86]. The work provides a simple property that indicates whether a function implies fair coin-tossing or not. If the function satisfies the property, then the function implies fair coin tossing, in the sense that a fair protocol for the function implies the existence of a fair protocol for coin-tossing, and therefore it cannot be computed fairly by Cleve's impossibility. On the other hand, if a function f does not satisfy the property, then for any protocol for coin-tossing in the f-hybrid model there exists an (inefficient) adversary that biases the output of the honest party. Thus, the function does not imply fair coin-tossing, and may potentially be computed with complete fairness. The results hold also for the case where the parties have an ideal access to Oblivious Transfer [Rab81, EGL82]. The property that [ALR13] has defined is as follows:

Definition 4.12 (strictly-balanced property [ALR13]) *Let $f : \{x_1, \ldots, x_\ell\} \times \{y_1, \ldots, y_m\} \to \{0, 1\}$ be a function. We say that the function is* balanced with respect to probability vectors *if there exist probability vectors $\mathbf{p} = (p_1, \ldots, p_\ell)$, $\mathbf{q} = (q_1, \ldots, q_m)$ and a constant $0 < \delta < 1$ such that:*

$$\mathbf{p} \cdot M_f = \delta \cdot \mathbf{1}_m \qquad \text{and} \qquad M_f \cdot \mathbf{q}^T = \delta \cdot \mathbf{1}_\ell^T \ .$$

Intuitively, if such probability vectors exist, then in a single execution of the function f, party P_1 can choose its input according to distribution $\mathbf{p}$ which fixes the output distribution vector of P_2 to be $\delta \cdot \mathbf{1}_m$. This means that no matter what input (malicious) P_2 uses, the output is 1 with probability δ. Likewise, honest P_2 can choose its input according to distribution $\mathbf{q}$, and malicious P_1 cannot bias the result. We therefore obtain a fair coin-tossing protocol. On the other hand, [ALR13] shows that if the function does not satisfy the condition above, then there always exists a party that can bias the result of any coin-tossing protocol that can be constructed using f.

4.3.1 The Characterization

A full-dimensional function is an important special case of this unbalanced property, as was pointed out in Claim 4.2. Combining the above characterization of [ALR13], we get the following Theorem:

Theorem 4.13 *Let $f : \{x_1, \ldots, x_\ell\} \times \{y_1, \ldots, y_m\} \to \{0, 1\}$, and let M_f be the corresponding matrix representing f as above. Then:*

1. **Balanced with respect to probability vectors [ALR13]:**
   If there exist probability vectors $\mathbf{p} = (p_1, \ldots, p_\ell), \mathbf{q} = (q_1, \ldots, q_m)$ and a constant $0 < \delta < 1$ such that: $\mathbf{p} \cdot M_f = \delta \cdot \mathbf{1}_m$ and $M_f \cdot \mathbf{q}^T = \delta \cdot \mathbf{1}_\ell^T$.

   Then, the function f implies fair coin-tossing, and is impossible to compute fairly.

2. **Balanced with respect to arbitrary vectors, but not balanced with respect to probability vectors:**
   If there exist two non-zero vectors $\mathbf{p} = (p_1, \ldots, p_\ell) \in \mathbb{R}^\ell$, $\mathbf{q} = (q_1, \ldots, q_m) \in \mathbb{R}^m$, $\delta_1, \delta_2 \in \mathbb{R}$, such that: $\mathbf{p} \cdot M_f = \delta_1 \cdot \mathbf{1}_m$ and $M_f \cdot \mathbf{q}^T = \delta_2 \cdot \mathbf{1}_\ell^T$
   then we say that the function is balanced with respect to arbitrary vectors. Then, the function does not (information-theoretically) imply fair-coin tossing [ALR13]. Moreover:
   (a) *If δ_1 and δ_2 are non-zero, $\sum_{i=1}^{\ell} p_i \neq \delta_1$ and $\sum_{i=1}^{m} q_i \neq \delta_2$, then the function f cannot be computed using the GHKL protocol (Theorem 4.8).*
   (b) *Otherwise: this case is left not characterized. For a subset of this subclass, we show possibility (Proposition 4.10).*

3. **Unbalanced with respect to arbitrary vectors:**
   If for every non-zero $\mathbf{p} = (p_1, \ldots, p_\ell) \in \mathbb{R}^\ell$ and any $\delta_1 \in \mathbb{R}$ it holds that: $\mathbf{p} \cdot M_f \neq \delta_1 \cdot \mathbf{1}_m$, **OR** *for every non-zero $\mathbf{q} = (q_1, \ldots, q_m) \in \mathbb{R}^m$ and any $\delta_2 \in \mathbb{R}$ it holds that: $M_f \cdot \mathbf{q}^T \neq \delta_2 \cdot \mathbf{1}_\ell^T$, then f can be computed with complete fairness (Theorem 4.4).*

We remark that in general, if $|X| \neq |Y|$ then almost always a random function is in subclass 3. Moreover, if $|X| = |Y|$, only negligible amount of functions are in subclass 2b, and thus only negligible amount of functions are left not characterized.

If a function is balanced with respect to arbitrary vectors (i.e., the vector may contain negative values), then all the rows of the function lie in the hyperplane $\mathcal{H}(\mathbf{q}, \delta_2)$, and all the columns lie in the hyperplane $\mathcal{H}(\mathbf{p}, \delta_1)$. Observe that $\delta_1 = 0$ if and only if $\mathcal{H}(\mathbf{p}, \delta_1)$ passes through the origin, and $\sum_{i=1}^{\ell} p_i = \delta_1$ if and only if $\mathcal{H}(\mathbf{p}, \delta_1)$ passes through the all one point $\mathbf{1}$. Thus, the requirements of subclass 2a are a different formalization of the requirements of Theorem 4.8. Likewise, the requirements of subclass 3 are a different formalization of Theorem 4.4, as was proven in Claim 4.2.

4.3.2 Examples

We give few examples for interesting functions and practical tasks that can be computed with complete-fairness.

- **Set membership.** Assume some finite possible elements Ω, and consider the task of set-membership: P_1 holds $S \subseteq \Omega$, P_2 holds some elements $\omega \in \Omega$, and the parties with to find out (privately) whether $\omega \in S$. The number of possible inputs for P_1 is $|\Omega|$, whereas the number of possible inputs for P_2 is $|P(\Omega)| = 2^{|\Omega|}$, and the truth table for this function contains all possible Boolean vectors of length-$|\Omega|$ (See Figure 3 for the case of $\Omega = \{a, b\}$).
- **Private evaluation of Boolean function.** Let $\mathcal{F} = \{g \mid g : \Omega \to \{0,1\}\}$ be the family of all Boolean functions with $|\Omega|$ inputs. Assume that P_1 holds some function $g \in \mathcal{F}$ and P_2 holds some input $y \in \Omega$. The parties wish to privately learn $g(y)$. This task has the exact same truth-table as the set-membership functionality, i.e., it contains all possible Boolean vectors of length-$|\Omega|$ (in this case, each vector represents a possible function g).
- **Private matchmaking.** Assume that P_1 holds some set of preferences, while P_2 holds a profile. The parties wish to learn (privately) whether there is a matching between them. In fact, this task is a special case of the *subset-equal* functionality. that is, P_1 holds some $A \subseteq \Omega$, P_2 holds $B \subseteq \Omega$, and the parties wish to learn whether $A \subseteq B$. Although the possible inputs for both parties is $2^{|\Omega|}$ (i.e., $|X| = |Y|$), the truth-table for this functionality satisfies Proposition 4.11, contains monochromatic row and can be computed using the protocol. See Figure 3.
- **Set disjointness.** The set-disjointness is another functionality that is feasible although $|X| = |Y|$, and is defined as follows: P_1 holds $A \subseteq \Omega$, P_2 holds $B \subseteq \Omega$, and the parties learn whether $A \cap B = \emptyset$. In fact, the possibility of this function is easily derived from the possibility of $A \subseteq B$, using the following observation:

$$A \cap B = \emptyset \iff A \subseteq \overline{B}$$

	a	b
$\emptyset$	0	0
$\{a\}$	1	0
$\{b\}$	0	1
$\{a,b\}$	1	1

(a) The set-membership functionality

	$\emptyset$	$\{a\}$	$\{b\}$	$\{a,b\}$
$\emptyset$	1	1	1	1
$\{a\}$	0	1	0	1
$\{b\}$	0	0	1	1
$\{a,b\}$	0	0	0	1

(b) The $A \subseteq B$ functionality

	$\emptyset$	$\{a\}$	$\{b\}$	$\{a,b\}$
$\emptyset$	1	1	1	1
$\{a\}$	1	0	1	0
$\{b\}$	1	1	0	0
$\{a,b\}$	1	0	0	0

(c) The set-disjointness functionality

Figure 3: (1) The functionalities (a), (b) and (c) with $\Omega = \{a, b\}$

5 Extensions: Asymmetric Functions and Non-Binary Outputs

5.1 Asymmetric Functions

We now move to a richer class of functions, and consider asymmetric Boolean functions where the parties do not necessarily get the same output. We consider functions $f(x, y) = (f_1(x, y), f_2(x, y))$, where each f_i, $i \in \{1, 2\}$ is defined as: $f_i : \{x_1, \ldots, x_\ell\} \times \{y_1, \ldots, y_m\} \to \{0, 1\}$. Interestingly, our result here shows that if the function f_2 is of full-dimensional, then f can be computed fairly, irrespectively to the function f_1. This is because simulating P_1 is more challenging (because it is the first to receive an output) and the simulator needs to assume the rich description of f_2 in order to be able to bias the output of the honest party P_2. On the other hand, since P_2 is the second to receive an output, simulating P_2 is easy and the simulator does not need to bias the output of P_1.

The protocol. We first revise the protocol of [GHKL08]. The $\mathsf{RandOut}_1(x)$ algorithm is changed such the function that is being evaluated is f_1, and algorithm $\mathsf{RandOut}_2(y)$ is modified such that the function that is being evaluated is f_2. Step 2.2.b in the F_{dealer} functionality (Functionality 3.2) is modified such that each party receives an output according to its respective output, that is:

Step 2.2.b: *For $i = i^*$ to R: set $(a_i, b_i) = (f_1(x,y), f_2(x,y))$.*

Analysis. The proof for the case where P_2 follows exactly along the lines of the proof of Claim A.1. We now turn to corrupted P_1. Here, we again define the probability p_x for every $x \in X$ as $\Pr_{\hat{y} \leftarrow U_Y}[f_1(x,\hat{y}) = 1]$. Then, for every $y_j \in Y$, we define $p_{y_j} = \Pr_{\hat{x} \leftarrow X_{real}}[f_2(\hat{x}, y_j) = 1]$. This time, $(p_{y_1}, \ldots, p_{y_m}) = X_{real} \cdot M_{f_2}$, where M_{f_2} represents the function f_2.

The case where P_1 is corrupted is very similar to the symmetric case, however, the constraints and the vectors $Q^{x,a}$ are different. Using same calculations as the symmetric case, we get the following vectors $Q^{x,0} = (q_{y_1}^{x,0}, \ldots, q_{y_m}^{x,0})$ and $Q^{x,1} = (q_{y_1}^{x,1}, \ldots, q_{y_m}^{x,1})$:

$$q_{y_j}^{x,0} \stackrel{\text{def}}{=} \begin{cases} p_{y_j} + \frac{\alpha \cdot p_{y_j}}{(1-\alpha)\cdot(1-p_x)} & \text{if } f(x,y_j) = (0,0) \\ p_{y_j} + \frac{\alpha \cdot (p_{y_j}-1)}{(1-\alpha)\cdot(1-p_x)} & \text{if } f(x,y_j) = (0,1) \\ p_{y_j} & \text{if } f(x,y_j) = (1,0) \\ p_{y_j} & \text{if } f(x,y_j) = (1,1) \end{cases} ,$$

$$q_{y_j}^{x,1} \stackrel{\text{def}}{=} \begin{cases} p_{y_j} & \text{if } f(x,y_j) = (0,0) \\ p_{y_j} & \text{if } f(x,y_j) = (0,1) \\ p_{y_j} + \frac{\alpha \cdot p_{y_j}}{(1-\alpha)\cdot p_x} & \text{if } f(x,y_j) = (1,0) \\ p_{y_j} + \frac{\alpha \cdot (p_{y_j}-1)}{(1-\alpha)\cdot p_x} & \text{if } f(x,y_j) = (1,1) \end{cases} .$$

Observe that if $f_1(x_i, y_j) = 0$ it implies that $p_{x_i} \neq 1$, and therefore $Q^{x_i,0}$ is well defined (i.e., $p_{x_i} = 1$ if and only of the row X_i is the all one row). Similarly, if $f_1(x_i, y_j) = 1$ it implies that $p_{x_i} \neq 0$, and therefore $Q^{x,1}$ is well-defined. Overall, we get:

Theorem 5.1 *Let $f : \{x_1, \ldots, x_\ell\} \times \{y_1, \ldots, y_m\} \to \{0,1\} \times \{0,1\}$, where $f = (f_1, f_2)$. Let M_{f_2} be the matrix that represents f_2. If there exist X_{real}, $0 < \alpha < 1$ such that $\alpha^{-1} \in O(\mathsf{poly}(\kappa))$, such that for every $x \in X$, $a \in \{0,1\}$, there exists a probability vector $X_{ideal}^{x,a}$ such that:*

$$X_{ideal}^{x,a} \cdot M_{f_2} = Q^{x,a} ,$$

then Protocol 3.3 securely computes f with complete fairness.

Proof: The case of P_2 is corrupted follows exactly the same as the case in the proof of Theorem A.2. All is left is just to what are the requirements from $f(x, y_j)$ in each one of the possible outputs $(0,0), (0,1), (1,0), (1,1)$. The cases of $(0,0)$ and $(1,1)$ are exactly the same as in the proof of Theorem A.2, where the case of $(0,0)$ corresponds to the symmetric case of $f(x, y_j) = 0$, and the case of $(1,1)$ corresponds to the symmetric case $f(x,y_j)=1$. This defines the following requirements (exactly as Equation 1):

$$q_{y_j}^{x,0} \stackrel{\text{def}}{=} \begin{cases} p_{y_j} & \text{if } f(x,y_j)=(1,1) \\ p_{y_j} + \frac{\alpha \cdot p_{y_j}}{(1-\alpha)\cdot(1-p_x)} & \text{if } f(x,y_j)=(0,0) \end{cases} \qquad q_{y_j}^{x,1} \stackrel{\text{def}}{=} \begin{cases} p_{y_j} + \frac{\alpha\cdot(p_{y_j}-1)}{(1-\alpha)\cdot p_x} & \text{if } f(x,y_j)=(1,1) \\ p_{y_j} & \text{if } f(x,y_j)=(0,0) \end{cases}$$

We now turn to the cases where the outputs are distinct.

The case where $f(x,y_j)=(0,1)$. That is, we consider the case where $f_1(x,y_j)=0$, whereas $f_2(x,y_j)=1$. We get the following four possibilities:

output (a,b)	REAL	IDEAL
$(0,0)$	$(1-\alpha)\cdot(1-p_x)\cdot(1-p_y)+\alpha\cdot(1-p_y)$	$(1-\alpha)\cdot(1-p_x)\cdot(1-q_y^{x,0})$
$(0,1)$	$(1-\alpha)\cdot(1-p_x)\cdot p_y+\alpha\cdot p_y$	$(1-\alpha)\cdot(1-p_x)\cdot q_y^{x,0}+\alpha$
$(1,0)$	$(1-\alpha)\cdot p_x\cdot(1-p_y)$	$(1-\alpha)\cdot p_x\cdot(1-q_y^{x,1})$
$(1,1)$	$(1-\alpha)\cdot p_x\cdot p_y$	$(1-\alpha)\cdot p_x\cdot q_y^{x,1}$

The only difference from the case of $f(x,y_j)=(0,0)$ is that in the ideal execution, the $+\alpha$ is obtained in the second row instead of the first row. The probabilities are equal if and only if, the following hold:

$$q_y^{x,0} = p_y + \frac{\alpha\cdot(p_y-1)}{(1-\alpha)\cdot(1-p_x)} \qquad \text{and} \qquad q_y^{x,1} = p_y \ .$$

The case where $f(x,y)=(1,0)$. Similarly to the above, we obtain:

output (a,b)	REAL	IDEAL
$(0,0)$	$(1-\alpha)\cdot(1-p_x)\cdot(1-p_y)$	$(1-\alpha)\cdot(1-p_x)\cdot(1-q_y^{x,0})$
$(0,1)$	$(1-\alpha)\cdot(1-p_x)\cdot p_y$	$(1-\alpha)\cdot(1-p_x)\cdot q_y^{x,0}$
$(1,0)$	$(1-\alpha)\cdot p_x\cdot(1-p_y)+\alpha\cdot(1-p_y)$	$(1-\alpha)\cdot p_x\cdot(1-q_y^{x,1})+\alpha$
$(1,1)$	$(1-\alpha)\cdot p_x\cdot p_y+\alpha\cdot p_y$	$(1-\alpha)\cdot p_x\cdot q_y^{x,1}$

The only difference from the case of $f(x,y_j)=(1,1)$ is that in the ideal execution, the $+\alpha$ is obtained in the second row instead of the forth row, which result in the following constraints:

$$q_y^{x,0} = p_y \qquad \text{and} \qquad q_y^{x,1} = p_y + \frac{\alpha\cdot p_y}{(1-\alpha)\cdot p_x} \ .$$

■

Similarly to the case of single output, the above implies that:

Corollary 5.2 *Let $f:\{x_1,\ldots,x_\ell\}\times\{y_1,\ldots,y_m\}\to\{0,1\}\times\{0,1\}$, where $f=(f_1,f_2)$. If f_2 is a full-dimensional function, then f can be computed with complete fairness.*

Proof: It is easy to see that for every constant $\epsilon>0$, for every $x\in X$ and $a\in\{0,1\}$, and for all sufficiently large κ's, it holds that $Q^{x,a}\in B(\mathbf{c},\epsilon)$, where $\mathbf{c}=(p_{y_1},\ldots,p_{y_m})$. As we saw, in case f_2 is of full-dimension, we can embed $\mathbf{c}$ inside the convex and there exists a constant $\epsilon>0$ such that $B(\mathbf{c},\epsilon)\subset \mathbf{conv}(\{X_1^{(2)},\ldots,X_\ell^{(2)}\})$, where $X_i^{(2)}$ is the ith row of f_2. The corollary follows. ■

5.2 Functions with Non-Binary Output

Until now, all the known possibility results in fairness deal with the case of binary output. We now extend the results to the case of non-binary output. Let $\Sigma = \{0, \ldots, k-1\}$ be an alphabet for some finite $k > 0$ (the alphabet may be arbitrary, we use $[0, k-1]$ for the sake of convenience), and consider functions $f : \{x_1, \ldots, x_\ell\} \times \{y_1, \ldots, y_m\} \to \Sigma$.

The protocol is exactly the GHKL protocol presented in Section 3, where here a_i, b_i are elements in Σ and not just bits. We just turn to the analysis. The proof of corrupted P_2 again follows along the lines of the proof of Claim A.1. The difficult part is corrupted P_1. Again, all the claims follow exactly the same and we just need to compare the output distributions of the parties in case the adversary aborts before or at i^*. We analyze this in the following.

Fix some X_{real}, and let U_Y denote the uniform distribution over Y. For any symbol $\sigma \in \Sigma$, and for every $x \in X$, denote by $p_x(\sigma)$ the probability that $\mathsf{RandOut}_1(x)$ is σ. Similarly, for every $y_j \in Y$, let $p_{y_j}(\sigma)$ denote the probability that the output of $\mathsf{RandOut}_2(y_j)$ is σ. That is:

$$p_x(\sigma) \stackrel{\text{def}}{=} \Pr_{\hat{y} \leftarrow U_Y}[f(x, \hat{y}) = \sigma] \qquad \text{and} \qquad p_{y_j}(\sigma) \stackrel{\text{def}}{=} \Pr_{\hat{x} \leftarrow X_{real}}[f(\hat{x}, y_j) = \sigma] \ .$$

Observe that $\sum_{\sigma \in \Sigma} p_x(\sigma) = 1$ and $\sum_{\sigma \in \Sigma} p_{y_j}(\sigma) = 1$.

For every $\sigma \in \Sigma$, we want to present the vector $(p_{y_1}(\sigma), \ldots, p_{y_m}(\sigma))$ as a function of X_{real} and M_f, as we did in the binary case (where there we just had: $(p_{y_1}, \ldots, p_{y_m}) = X_{real} \cdot M_f$). However, here it is a bit more complicated then the binary case. Therefore, for any $\sigma \in \Sigma$ we define the binary matrix M_f^σ as follows:

$$M_f^\sigma(i,j) = \begin{cases} 1 & \text{if } f(x_i, y_j) = \sigma \\ 0 & \text{otherwise} \end{cases} .$$

Then, for every $\sigma \in \Sigma$, it holds that: $(p_{y_1}(\sigma), \ldots, p_{y_m}(\sigma)) = X_{real} \cdot M_f^\sigma$. Moreover, it holds that: $\sum_{\sigma \in \Sigma} M_f^\sigma = \mathbf{J}_{\ell \times m}$, where $\mathbf{J}_{\ell \times m}$ is the all one matrix with sizes $\ell \times m$. In the binary case, M_f is in fact M_f^1, and there was no need to consider the matrix M_f^0, for a reason that we will see below (intuitively, since in the binary case, $p_{y_j}(0) + p_{y_j}(1) = 1$).

The output distribution vectors. Similarly as in the binary case, in case the adversary sends x to the simulated F_{dealer} and aborts at round $i < i^*$ upon receiving some symbol $a \in \Sigma$, the simulator chooses its input $\hat{x}$ according to some distribution $X_{ideal}^{x,a}$. For each such a vector $X_{ideal}^{x,a}$, we define $|\Sigma|$ points $Q^{x,a}(\sigma_1), \ldots, Q^{x,a}(\sigma_k)$ in $\mathbb{R}^m$, where each vector $Q^{x,a}(b)$ (for $b \in \Sigma$) is obtained by: $Q^{x,a}(b) = X_{ideal}^{x,a} \cdot M_f^b$.

For a given $x \in X, y_j \in Y$ and $a, b \in \Sigma$, we define the requirements from each vector $Q^{x,a}(b) = (q_{y_1}^{x,a}(b), \ldots, q_{y_m}^{x,a}(b))$. We have:

$$q_{y_j}^{x,a}(b) \stackrel{\text{def}}{=} \begin{cases} p_{y_j}(b) + \frac{\alpha}{(1-\alpha)} \cdot \frac{(p_{y_j}(b)-1)}{p_x(a)} & \text{if } f(x, y_j) = a = b \\ p_{y_j}(b) + \frac{\alpha}{(1-\alpha)} \cdot \frac{p_{y_j}(b)}{p_x(a)} & \text{if } f(x, y_j) = a \neq b \\ p_{y_j}(b) & \text{if } f(x, y_j) \neq a \end{cases} \tag{3}$$

We then require that for the *same* distribution $X_{ideal}^{x,a}$ it holds that $X_{ideal}^{x,a} \cdot M_f^b = Q^{x,a}(b)$ for every $b \in \Sigma$, *simultaneously*. We have the following Theorem:

Theorem 5.3 *Let $f : \{x_1, \ldots, x_\ell\} \times \{y_1, \ldots, y_m\} \to \Sigma$ be a function. If there exists a parameter $0 < \alpha < 1$, $\alpha^{-1} \in O(\mathsf{poly}(\kappa))$ and a distribution X_{real}, such that for every $x \in X$, for every $a \in \Sigma$ and for every $b \in \Sigma$ it holds that:*

$$X_{ideal}^{x,a} \cdot M_f^b = Q^{x,a}(b) \quad ,$$

Then the function f can be computed with complete fairness.

Proof: The simulator for P_1 is exactly the same as in Theorem 3.4. All is left is to show that the outputs are distributed identically in case the adversary aborts before or at i^*.

The real execution. We now consider the output distribution where the adversary halts at round i for which $i < i^*$. In such a case, both parties output independent outputs according to $\mathsf{RandOut}_1(x)$ and $\mathsf{RandOut}_2(y_j)$, where (x, y_j) are the inputs sent to the F_{dealer}. As a result, they receive output (a, b) with probability $(1-\alpha) \cdot p_x(a) \cdot p_{y_j}(b)$, where $(1-\alpha)$ is the probability that $i < i^*$ given that $i \leq i^*$. In case $i = i^*$ (which given $i \leq i^*$, it happens with probability α), the adversary P_1 learns the correct output $f(x, y_j)$, whereas P_2 gets an output according to $\mathsf{RandOut}_2(y_j)$. Overall, we have that:

$$\begin{aligned}
&\Pr\left[(\text{VIEW}_{\mathsf{hyb}}^i, \text{OUT}_{\mathsf{hyb}}) = (a,b) \ \mid \ i \leq i^*\right] \\
&= \begin{cases}
(1-\alpha) \cdot p_x(a) \cdot p_{y_j}(b) + \alpha \cdot 1 \cdot p_{y_j}(b) & \text{if } f(x,y_j) = a = b \\
(1-\alpha) \cdot p_x(a) \cdot p_{y_j}(b) + \alpha \cdot 1 \cdot p_{y_j}(b) & \text{if } f(x,y_j) = a \neq b \\
(1-\alpha) \cdot p_x(a) \cdot p_{y_j}(b) & \text{if } f(x,y_j) \neq a
\end{cases}
\end{aligned}$$

(we differentiate between the first two cases although they are equal here, since they are different in the ideal.)

The ideal execution. In case the adversary aborts at i^*, the simulator sends the true input x to the trusted party and *both* parties receive the correct output $f(x, y_j)$. On the other hand, in case the adversary aborts at round $i < i^*$ after sending some $x \in X$ to the simulated F_{dealer} and upon receiving a symbol $a \in \Sigma$, the simulator chooses an input $\hat{x}$ according to distribution $X_{ideal}^{x,a}$, and the output of the honest party is determined accordingly, where we denote by $q_{y_j}^{x,a}(b)$ the probability that the output of the honest party P_2 is b in this case. Therefore, we have that:

$$\begin{aligned}
&\Pr\left[(\text{VIEW}_{\mathsf{ideal}}^i, \text{OUT}_{\mathsf{ideal}}) = (a,b) \ \mid \ i \leq i^*\right] \\
&= \begin{cases}
(1-\alpha) \cdot p_x(a) \cdot q_{y_j}^{x,a}(b) + \alpha & \text{if } f(x,y_j) = a = b \\
(1-\alpha) \cdot p_x(a) \cdot q_{y_j}^{x,a}(b) & \text{if } f(x,y_j) = a \neq b \\
(1-\alpha) \cdot p_x(a) \cdot q_{y_j}^{x,a}(b) & \text{if } f(x,y_j) \neq a
\end{cases}
\end{aligned}$$

Therefore, if Eq. (3) holds, then we have for every $(a, b) \in \Sigma^2$:

$$\Pr\left[(\text{VIEW}_{\mathsf{hyb}}^i, \text{OUT}_{\mathsf{hyb}}) = (a,b) \ \mid \ i \leq i^*\right] = \Pr\left[(\text{VIEW}_{\mathsf{ideal}}^i, \text{OUT}_{\mathsf{ideal}}) = (a,b) \ \mid \ i \leq i^*\right] \ .$$

This, combining with the rest of the proof of Theorem A.2 shows that the protocol can be simulated. ■

Relaxing the requirements. In the above Theorem, for the *same* distribution $X_{ideal}^{x,a}$ it is required that $X_{ideal}^{x,a} \cdot M_f^b = Q^{x,a}(b)$ for every $b \in \Sigma$ simultaneously. This requirement makes the things much harder, since it involves convex combinations with the same coefficients ($X_{ideal}^{x,a}$) of *different* convex-hulls $M_f^{\sigma_1}, \ldots, M_f^{\sigma_k}$. We therefore want to encode all the requirements together for the same $X_{ideal}^{x,a}$. We do it in two steps. First, we show that it it is enough to consider equality with respect to $|\Sigma| - 1$ symbols in the alphabet, and the last one is a simple derivation of the others (this also explains why in the binary case, we did not consider the matrix M_f^0.). Next, we show hot to encode all the requirements together into a single system.

Claim 5.4 *Let f, be as above and let $\sigma \in \Sigma$ be arbitrary. Then, if there exists $0 < \alpha < 1$ (such that $\alpha^{-1} \in O(\mathsf{poly}(\kappa))$, distribution X_{real} such that for every $x \in X$, for every $a \in \Sigma$ and for every $b \in \Sigma \setminus \{\sigma\}$*

$$X_{ideal}^{x,a} \cdot M_f^b = Q^{x,a}(b) \ ,$$

then it holds that $X_{ideal}^{x,a} \cdot M_f^\sigma = Q^{x,a}(\sigma)$ as well and thus f can be computed with complete fairness.

Proof: Fix σ, α, X_{real} and assume that for every $x \in X$, $a \in \Sigma$ there exists $X_{ideal}^{x,a}$ as above. We now show that the $X_{ideal}^{x,a} \cdot M_f^\sigma = Q^{x,a}(\sigma)$. The key observation is that the sum of all matrices $\sum_{b \in \Sigma} M_f^b$ is $\mathbf{J}_{\ell \times m}$ (the all-one matrix). Moreover, for every x, a, it holds that $\sum_b Q^{x,a}(b) = \mathbf{1}_m$. This can is derived by analyzing each coordinate $q_{y_j}^{x,a}$ separately as follows:

- **Case 1 – $f(x, y_j) \neq a$:** In this case, $q_{y_j}^{x,a}(b) = p_{y_j}(b)$ for every $b \in \Sigma$. Therefore,

$$\sum_{b \in \Sigma} q_{y_j}^{x,a}(b) = \sum_{b \in \Sigma} p_{y_j}(b) = 1 \ .$$

- **Case 2 – $f(x, y_j) = a$:** In this case, for every $b \neq a$, we have that:

$$q_{y_j}^{x,a}(b) = p_{y_j}(b) + \frac{\alpha}{(1-\alpha)} \cdot \frac{p_{y_j}(b)}{p_x(a)}$$

  Observe that $\sum_{b \neq a} p_{y_j}(b) = 1 - p_{y_j}(a)$ (since they all sum-up to 1), and therefore we have:

$$\sum_{b \neq a} q_{y_j}^{x,a}(b) = \big(1 - p_{y_j}(a)\big) + \frac{\alpha}{(1-\alpha)} \cdot \frac{1 - p_{y_j}(a)}{p_x(a)} \ .$$

  On the other hand, for $b = a$ we have: $q_{y_j}^{x,a}(a) = p_{y_j}(a) + \frac{\alpha}{(1-\alpha)} \cdot \frac{(p_{y_j}(a)-1)}{p_x(a)}$, and thus:

$$\begin{aligned}\sum_{b \in \Sigma} q_{y_j}^{x,a}(b) &= q_{y_j}^{x,a}(a) + \sum_{b \neq a} q_{y_j}^{x,a}(b) \\ &= \left(p_{y_j}(a) + \frac{\alpha}{(1-\alpha)} \cdot \frac{(p_{y_j}(a)-1)}{p_x(a)}\right) + \left(\big(1 - p_{y_j}(a)\big) + \frac{\alpha}{(1-\alpha)} \cdot \frac{1 - p_{y_j}(a)}{p_x(a)}\right) = 1 \ .\end{aligned}$$

Therefore, we can conclude that for every x, a it holds that: $Q^{x,a}(\sigma) = \mathbf{1}_m - \sum_{b \neq \sigma} Q^{x,a}(b)$. Thus, we conclude:

$$X_{ideal}^{x,a} \cdot M_f^\sigma = X_{ideal}^{x,a} \cdot \Big(\mathbf{J}_{\ell \times m} - \sum_{b \neq \sigma} M_f^b\Big) = \mathbf{1}_m - \sum_{b \neq \sigma} X_{ideal}^{x,a} \cdot M_f^b = \mathbf{1}_m - \sum_{b \neq \sigma} Q^{x,a}(b) = Q^{x,a}(\sigma) \ .$$

and the claim follows. ■

Encoding the requirements together. Let M_f denote the $\ell \times (|\Sigma| - 1) \cdot m$ binary matrix defined as the concatenation of $M_f^{\sigma_1} || \ldots || M_f^{\sigma_{k-1}}$. Let $Q^{x,a}$ be the $(|\Sigma| - 1) \cdot m$ vector which is a concatenation of $Q^{x,a} = (Q^{x,a}(\sigma_1), \ldots, Q^{x,a}(\sigma_{k-1}))$. Then, we have that:

Corollary 5.5 *Let $f : \{x_1, \ldots, x_\ell\} \times \{y_1, \ldots, y_m\} \to \Sigma$, M_f, $Q^{x,a}$ be as above. If there exists a parameter $0 < \alpha < 1$ (for which $\alpha^{-1} \in O(\mathsf{poly}(\kappa))$), and a distribution X_{real}, such that for every $x \in X$, for every $a \in \Sigma$ it holds that:*

$$X_{ideal}^{x,a} \cdot M_f = Q^{x,a} \quad ,$$

then the function f can be computed with complete fairness.

As a result, if the rows of the matrix M_f define a full-dimensional object (this time, in $\mathbb{R}^{\ell \times (|\Sigma|-1) \cdot m}$, then the function can be computed fairly.

Corollary 5.6 *let M_f be an $\ell \times ((|\Sigma|-1) \cdot m)$ matrix as above and let $X_1, \ldots, X_\ell$ be the rows of M_f. If there exists a subset of cardinality $(|\Sigma| - 1) \cdot m + 1$ of $\{X_1, \ldots, X_\ell\}$ that is affinely-independent, then f can be computed with complete fairness.*

Proof: Let $s = (|\Sigma| - 1) \cdot m$. Fix $\alpha = 1/\ln(\kappa)$ and let X_{real} be the uniform distribution over the affinely independent rows in M_f (and assigns 0 to the others, if exist). Similarly to the binary case, we observe that all the points $Q^{x,a}$ (this time - points in $\mathbb{R}^s$ and not $\mathbb{R}^m$) are in the Euclidian ball $B(\mathbf{c}, \epsilon)$, where $\mathbf{c} = X_{real} \cdot M_f$. Moreover, since $\mathbf{conv}(\{X_1, \ldots, X_\ell\})$ defines an s-dimensional simplex in $\mathbb{R}^s$, the ball $B(\mathbf{c}, \epsilon)$ is inside it, and therefore there exist probability vectors $X_{ideal}^{x,a}$ as above. ■

Alternative representation. We now write the above in a different form. Giving a function $f : X \times Y \to \Sigma$, let $\rho \in \Sigma$ be arbitrarily, and define $\Sigma_\rho = \Sigma \setminus \{\rho\}$. Define the *Boolean* function $f' : X \times Y^{\Sigma_\rho} \to \{0, 1\}$, where $Y^{\Sigma_\rho} = \{y_j^\sigma \mid y_j \in Y, \sigma \in \Sigma_\rho\}$, as follows:

$$f'(x, y_j^\sigma) = \begin{cases} 1 & \text{if } f(x, y_j) = \sigma \\ 0 & \text{otherwise} \end{cases}$$

Observe that $|Y^{\Sigma_\rho}| = (|\Sigma| - 1) \cdot |Y|$. It is easy to see that if f is full-dimensional, then the function f can be computed with complete-fairness. This provide a property that may be satisfied only when $|X|/|Y| > |\Sigma| - 1$.

An example. We give an example for a non-binary function that can be computed with complete-fairness. We consider trinary alphabet $\Sigma = \{0, 1, 2\}$, and thus we consider a function of dimensions 5×2. We provide the trinary function f and the function f' that it reduced to. Since the binary function f' is a full-dimensional function in $\mathbb{R}^4$, it can be computed fairly, and thus the trinary function f can be computed fairly as well. We have:

Acknowledgement

The author gratefully thanks Eran Omri, Ran Cohen, Carmit Hazay, Yehuda Lindell and Tal Rabin for very helpful discussions and helpful comments.

f	y_1	y_2
x_1	0	1
x_2	1	0
x_3	1	1
x_4	2	0
x_5	1	2

$\Longrightarrow$

f'	y_1^1	y_2^1	y_1^2	y_2^2
x_1	0	1	0	0
x_2	1	0	0	0
x_3	1	1	0	0
x_4	0	0	1	0
x_5	1	0	0	1

References

[ALR13] Gilad Asharov, Yehuda Lindell, and Tal Rabin. A full characterization of functions that imply fair coin tossing and ramifications to fairness. In *TCC*, pages 243–262, 2013.

[AP13] Shashank Agrawal and Manoj Prabhakaran. On fair exchange, fair coins and fair sampling. In *CRYPTO (1)*, pages 259–276, 2013.

[BGMR90] Michael Ben-Or, Oded Goldreich, Silvio Micali, and Ronald L. Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, 36(1):40–46, 1990.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[BLOO11] Amos Beimel, Yehuda Lindell, Eran Omri, and Ilan Orlov. $1/p$-secure multiparty computation without honest majority and the best of both worlds. In *CRYPTO*, pages 277–296, 2011.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, pages 11–19, 1988.

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

[Cle89] Richard Cleve. Controlled gradual disclosure schemes for random bits and their applications. In *CRYPTO*, pages 573–588, 1989.

[EGL82] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In *CRYPTO*, pages 205–210, 1982.

[EY80] Shimon Even and Yacov Yacobi. Relations among public key signature schemes. *Technical Report #175, Technion Israel Institute of Technology, Computer Science Department*, 1980.

[GHKL08] S. Dov Gordon, Carmit Hazay, Jonathan Katz, and Yehuda Lindell. Complete fairness in secure two-party computation. In *STOC*, pages 413–422., 2008. Extended full version available on: `http://eprint.iacr.org/2008/303`. Journal version: [GHKL11].

[GHKL11] S. Dov Gordon, Carmit Hazay, Jonathan Katz, and Yehuda Lindell. Complete fairness in secure two-party computation. *J. ACM*, 58(6):24, 2011.

[GK10] S. Dov Gordon and Jonathan Katz. Partial fairness in secure two-party computation. In *EUROCRYPT*, pages 157–176, 2010.

[GL90] Shafi Goldwasser and Leonid A. Levin. Fair computation of general functions in presence of immoral majority. In *CRYPTO*, pages 77–93, 1990.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications.* Cambridge University Press, 2004.

[Gor10] S. Dov Gordon. *On fairness in secure computation.* PhD thesis, University of Maryland, 2010.

[Grü03] Branko Grünbaum. *Convex Polytopes : Second Edition Prepared by Volker Kaibel, Victor Klee, and Günter Ziegler (Graduate Texts in Mathematics).* Springer, May 2003.

[HT04] Joseph Y. Halpern and Vanessa Teague. Rational secret sharing and multiparty computation: extended abstract. In *STOC*, pages 623–632, 2004.

[JKS95] J. Komlòs J. Kahn and E. Szemerèdi. On the probability that a random ± 1-matrix is singular. *Journal of Amer. Math. Soc.*, 8:223–240, 1995.

[Kom67] J. Komlòs. On the determinant of $(0,1)$ matrices. *Studia Sci. Math. Hungar*, 2:7–21, 1967.

[Rab81] Michael O. Rabin. How to exchange secrets with oblivious transfer. *Technical Report TR-81, Aiken Computation Lab, Harvard University*, 1981.

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *STOC*, pages 73–85, 1989.

[Rom08] Steven Roman. *Advanced Linear Algebra 3rd ed.* Graduate Texts in Mathematics 135. New York, NY: Springer. xviii, 2008.

[VZ06] Thomas Voigt and Günter M. Ziegler. Singular 0/1-matrices, and the hyperplanes spanned by random 0/1-vectors. *Combinatorics, Probability and Computing*, 15(3):463–471, 2006.

[Woo09] Philip J. Wood. *On the probability that a discrete complex random matrix is singular.* PhD thesis, Rutgers University, New Brunswick, NJ, USA, 2009. AAI3379178.

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

[Zie00] Günter M. Ziegler. Lectures on 0/1-polytopes. *Polytopes: Combinatorics and Computation, Vol. 29 of DMV Seminar, Birkhauser, Basel*, pages 1–40, 2000.

A Security Proofs of [GHKL08]

We now analyze the security of the protocol. All the analysis that we present in this section appears also in [GHKL08], and is given here for completeness.

First, when both parties are honest, they both receive an output except for some negligible probability. Recall that in the first step of the protocol, the number of rounds R is set as $\alpha^{-1} \cdot \omega(\log \kappa)$. Both parties learn the correct output unless $i^* > R$, which happens with probability $(1-\alpha)^R < e^{-\alpha \cdot R} \leq e^{-\omega(\log \kappa)}$, which is negligible in κ. Note that we can set α to be polynomially small (say, $1/\kappa$), and still get number of rounds which is polynomial ($\kappa \cdot \omega(\log \kappa)$). In our results, we set $\alpha = 1/\ln \kappa$.

A.1 Security with Respect to Corrupted P_2

In each round, P_1 is the first to receive a message from F_{dealer}. As a result, in round i^* the party P_2 gets an output *after* P_1 has already received its output. Therefore, simulating corrupted P_2 is easy: the simulator invokes the adversary and receives its input y to the simulated F_{dealer}. It then chooses the round i^* as F_{dealer}, and gives the adversary values according to $\mathsf{RandOut}_2(y)$ for each round until i^*. In case the round i^* is reached, the simulator sends y to the trusted party computing f, receives the output $f(x, y)$, and gives the adversary this value until round R. Clearly, if the adversary aborts after or at i^*, the honest party P_1 has already learned the correct output in the real execution and also in the ideal. If the adversary aborts before i^*, the output of the honest party P_1 is determined according to $\mathsf{RandOut}_1(x)$, that is, $f(x, \hat{y})$ where $\hat{y}$ is chosen uniformly at random. Therefore, in case the adversary aborts before i^*, the simulator chooses $\hat{y}$ according to the uniform distribution over Y and sends this input to the trusted party. Overall, the protocol can be simulated *for any function* f. [GHKL08] proves the following claim:

Claim A.1 *For every function $f : \{x_1, \ldots, x_\ell\} \times \{y_1, \ldots, y_m\} \to \{0, 1\}$, for every set of distribution X_{real}, for every $0 < \alpha < 1$ such that $\alpha^{-1} \in O(\mathsf{poly}(\kappa))$, Protocol 3.3 securely computes f in the presence of malicious adversary corrupting P_2.*

Proof: Fix f, X_{real} and α. Let $\mathcal{A}$ be an adversary that corrupts P_2. We construct the simulator $\mathcal{S}$ as follows:

1. $\mathcal{S}$ invokes $\mathcal{A}$ on input y and with auxiliary input z.
2. $\mathcal{S}$ chooses $\hat{y} \leftarrow Y$ uniformly, as in algorithm $\mathsf{RandOut}_1$. This value will be sent to the trusted party as the input of $\mathcal{A}$ in case $\mathcal{A}$ aborts before i^*.
3. $\mathcal{S}$ receives y' from $\mathcal{A}$ as was sent to F_{dealer} (Step 3 in the protocol). It then verifies that $y' \in Y$, if not – it sends abort as response from F_{dealer}, sends $\hat{y}$ to the trusted party and halts.
4. $\mathcal{S}$ chooses i^* according to geometric distribution with parameter α.
5. For every $i = 1, \ldots, i^* - 1$:
   (a) If $\mathcal{S}$ receives $\mathsf{proceed}$ from $\mathcal{A}$, then it sets $b_i = \mathsf{RandOut}_2(y')$ and sends b_i to $\mathcal{A}$ as was given from F_{dealer}.
   (b) If $\mathcal{S}$ receives abort from $\mathcal{A}$, it sends to $\mathcal{A}$ the message abort as was given from F_{dealer}. In addition, it sends $\hat{y}$ to the trusted party computing f, outputs whatever $\mathcal{A}$ outputs and halts.

6. In the simulated round $i = i^*$:

   (a) If $\mathcal{S}$ receives proceed from $\mathcal{A}$, then it sends y' to the trusted party computing f and receives back the output $b_{out} = f(x, y')$. It then gives b_{out} back to $\mathcal{A}$.

   (b) If $\mathcal{S}$ receives abort from $\mathcal{A}$, it gives back to $\mathcal{A}$ the message abort as was sent from F_{dealer}. Then, it sends the default input $\hat{y}$ to the trusted party computing f, outputs whatever $\mathcal{A}$ outputs and halts.

7. In the simulated rounds $i = i^* + 1$ to R:

   (a) If $\mathcal{S}$ receives proceed from $\mathcal{A}$, then it gives b_{out} to $\mathcal{A}$.

   (b) If $\mathcal{S}$ receives abort from $\mathcal{A}$, it sends abort to $\mathcal{A}$ as was given from F_{dealer}.

8. If $\mathcal{S}$ has not halted yet and $i^* > R$ – $\mathcal{S}$ sends $\hat{y}$ to the trusted party.

9. $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs and halts.

There is no communication between P_1 and P_2, and all the view of P_2 consists of the outputs of the F_{dealer} functionality to P_2. Moreover, P_2 only sends y' in the first round to F_{dealer}, and all the rest of the messages are either proceed or abort. Therefore, after sending y', all what $\mathcal{A}$ can do is either send proceed or to abort the interaction. Assume that $\mathcal{A}$ aborts at some round i. If $i = 0$, i.e., if $\mathcal{A}$ does not send the input y' to F_{dealer}, then in the real execution P_1 outputs $\mathsf{RandOut}_1(x)$ which is independent of y'. In the ideal, $\mathcal{S}$ chooses $\hat{y}$ uniformly from Y, and sends to the trusted party computing f the value $\hat{y}$, which determines the output of P_1 to be $f(x, \hat{y})$. This is exactly the same as the implementation of $\mathsf{RandOut}_1(x)$. In case $\mathcal{A}$ aborts at some round $i < i^*$, the view of P_2 consists of i independent invocations of $\mathsf{RandOut}_2(y')$, while the output of the honest P_1 consists of $\mathsf{RandOut}_1(x)$. $\mathcal{S}$ works exactly in the same way – for every round until i^*, it sends to P_2 a fresh output that depends only on the value y' – $\mathsf{RandOut}_2(y')$. In case it aborts, it sends $\hat{y}$ as we above, resulting the output of P_1 to distribute identically as in the real execution. In case $\mathcal{A}$ aborts at i^* or after i^* (i.e., in case $i^* \geq i$), P_1 has already learned the output $b_{out} = f(x, y')$ in the real execution. Therefore, $\mathcal{S}$ can send the true input y' to the trusted party, which determines the output of P_1 to be $f(x, y')$. It learns the output b_{out}, and gives this value to $\mathcal{A}$ as the outputs of F_{dealer}, exactly as in the real execution. ■

A.2 Security with Respect to Corrupted P_1

This case is more complex. Intuitively, the adversary does have an advantage in the real execution, in case it succeeds to predict correctly and aborts exactly at round i^*. In such a case, the corrupted party P_1 learns the correct output $f(x, y)$, whereas P_2 learns a value according to $\mathsf{RandOut}_2(y)$. However, as we will see, this advantage in the real execution can be simulated in the ideal execution under certain circumstances, for *some* functions f.

The output vector distributions $Q^{x,a}$. Let $f : \{x_1, \ldots, x_\ell\} \times \{y_1, \ldots, y_m\} \to \{0, 1\}$. Fix X_{real}, and let U_Y denote the uniform distribution over Y. For every $x \in X$, denote by p_x the probability that the output of $\mathsf{RandOut}_1$ is 1 on the input x. Similarly, for every $y_j \in Y$, let p_{y_j} denote the probability that the output of $\mathsf{RandOut}_2$ is 1 on the input y_j. That is:

$$p_x \overset{\text{def}}{=} \Pr_{\hat{y} \leftarrow U_Y}[f(x, \hat{y}) = 1] \quad \text{and} \quad p_{y_j} \overset{\text{def}}{=} \Pr_{\hat{x} \leftarrow X_{real}}[f(\hat{x}, y_j) = 1]$$

For every $x \in X$, $a \in \{0,1\}$, define the m-dimensional row vectors $Q^{x,a} = (q_{y_1}^{x,a}, \dots, q_{y_m}^{x,a})$ indexed by $y_j \in Y$ as follows:

$$q_{y_j}^{x,0} \overset{\text{def}}{=} \begin{cases} p_{y_j} & \text{if } f(x,y_j)=1 \\ p_{y_j} + \frac{\alpha \cdot p_{y_j}}{(1-\alpha)\cdot(1-p_x)} & \text{if } f(x,y_j)=0 \end{cases} \qquad q_{y_j}^{x,1} \overset{\text{def}}{=} \begin{cases} p_{y_j} + \frac{\alpha\cdot(p_{y_j}-1)}{(1-\alpha)\cdot p_x} & \text{if } f(x,y_j)=1 \\ p_{y_j} & \text{if } f(x,y_j)=0 \end{cases} \tag{4}$$

We have:

Theorem A.2 *Let $f : \{x_1, \dots, x_\ell\} \times \{y_1, \dots, y_m\} \to \{0,1\}$ and let M_f be as above. If there exist probability vector X_{real}, parameter $0 < \alpha < 1$ such that $\alpha^{-1} \in O(\mathsf{poly}(\kappa))$, such that for every $x \in X$, $a \in \{0,1\}$, there exists a probability vector $X_{ideal}^{x,a}$ for which:*

$$X_{ideal}^{x,a} \cdot M_f = Q^{x,a} \ ,$$

then Protocol 3.3 securely computes f with complete fairness.

Proof: We start with the description of the simulator $\mathcal{S}$. The simulator chooses i^* as F_{dealer}. If the adversary aborts after i^*, then both parties learn output and so the simulator sends the true input to the trusted party and fairness is obtained. If the adversary aborts i^*, then the simulator needs to give the adversary the true output $f(x,y)$. Thus, it sends the true input to the trusted party. However, by doing so the honest party learn the correct output unlike the real execution, when it outputs a random value according to $\mathsf{RandOut}_2(y)$. Thus, if the adversary aborts before i^*, the simulator chooses the input to send to the trusted party not according to X_{real}, but according to $X_{ideal}^{x,a}$. This should balance the advantage that the simulator gives the honest party in case the adversary aborts exactly at i^*.

The simulator $\mathcal{S}$.

1. $\mathcal{S}$ invokes $\mathcal{A}$ with input x and auxiliary input z.
2. When $\mathcal{A}$ sends x' to F_{dealer}, $\mathcal{S}$ checks that $x' \in X$ and sets $x = x'$. If $x' \notin X$, $\mathcal{S}$ chooses a default input $\hat{x} \in X$ according to the distribution X_{real}, sends $\hat{x}$ to the trusted party, outputs whatever $\mathcal{A}$ outputs and halts.
3. $\mathcal{S}$ chooses i^* according to geometric distribution with parameter α.
4. For every $i = 1$ to $i^* - 1$:
   (a) $\mathcal{S}$ chooses $a_i = \mathsf{RandOut}_1(x)$ and gives a_i to $\mathcal{A}$ as was sent from F_{dealer}.
   (b) If $\mathcal{A}$ sends abort back to F_{dealer}, then $\mathcal{S}$ chooses $\hat{x}$ according to the distribution X_{ideal}^{x,a_i}. It then sends $\hat{x}$ to the trusted party computing f, outputs whatever $\mathcal{A}$ outputs and halts. If $\mathcal{A}$ sends $\mathsf{proceed}$, then $\mathcal{S}$ proceeds to the next iteration.
5. In round $i = i^*$:
   (a) $\mathcal{S}$ sends the input x to the trusted party computing f and receives $a_{out} = f(x,y)$.
   (b) $\mathcal{S}$ gives to $\mathcal{A}$ the value a_{out} as was sent from F_{dealer}.
6. In rounds $i = i^* + 1$ to R:

(a) If $\mathcal{A}$ sends $\mathsf{proceed}$, $\mathcal{S}$ gives back to $\mathcal{A}$ the value a_{out} and proceeds to the next iteration.

7. If $\mathcal{S}$ has not halted yet and $i^* > R$, then $\mathcal{S}$ chooses $\hat{x}$ according to the distribution X_{real}, it sends $\hat{x}$ to the trusted party.

8. $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs and halts.

Let $\text{VIEW}_{\mathsf{hyb}}(x,y)$ denote the view of P_1 in an execution of the protocol with the adversary $\mathcal{A}$, where $\mathcal{A}$'s input is x, its auxiliary input is z and P_2's input is y. Let $\text{OUT}_{\mathsf{hyb}}(x,y)$ be the output of P_2 in such an execution. Let $\text{VIEW}_{\mathsf{ideal}}(x,y)$ denote the view of the adversary in the ideal execution where the parties are invoked with the inputs (x,y), respectively, and the auxiliary input of the simulator is z. Let $\text{OUT}_{\mathsf{ideal}}$ be the output of P_2 in the ideal execution.

Since $\mathcal{A}$ can only abort the execution, let i denote the round for which $\mathcal{A}$ has aborted. In case $\mathcal{A}$ does not abort during the execution of the protocol, then set $i = R+1$. Note that $\mathcal{A}$ may also abort *before* the invocation of F_{dealer}; in such a case we say that $i=0$. We want to show that for every $(\vec{a}, b) \in \{0,1\}^{i+1}$, it holds that:

$$\Pr\left[(\text{VIEW}_{\mathsf{hyb}}(x,y), \text{OUT}_{\mathsf{hyb}}(x,y)) = (\vec{a}, b)\right] = \Pr\left[(\text{VIEW}_{\mathsf{ideal}}(x,y), \text{OUT}_{\mathsf{ideal}}(x,y)) = (\vec{a}, b)\right] \qquad (5)$$

If $\mathcal{A}$ aborts before the invocation of F_{dealer} (or has sent an invalid input to F_{dealer}), then the view of the adversary is empty, and the honest party P_2 outputs a value according to $\mathsf{RandOut}_2(y)$. The same happens in the ideal execution, since $\mathcal{S}$ sends in such a case a value $\hat{x}$ distributed according to X_{real}. In such a case, both terms of Eq. (5) are 0 in case $\vec{a}$ is not empty, and clearly both terms have the same probability for any value of b. Thus, we get that for every $b \in \{0,1\}$:

$$\Pr\left[(\text{VIEW}_{\mathsf{hyb}}(x,y), \text{OUT}_{\mathsf{hyb}}(x,y)) = (\lambda, b) \mid i=0\right] = \Pr\left[(\text{VIEW}_{\mathsf{ideal}}(x,y), \text{OUT}_{\mathsf{ideal}}(x,y)) = (\lambda, b) \mid i=0\right] .$$

In case $i = R+1$ (i.e., the adversary does not abort), in the real the adversary sees values $a_1, \ldots, a_{i^*-1}$ that are independent to y, and all the values $a_{i^*}, \ldots, a_R$ are $f(x,y)$. In the ideal, we have the exact same thing. Therefore, Eq. (5) holds conditioning on the event $i = R+1$. Moreover, the above is true whenever the adversary aborts at a round $i > i^*$, and thus the equation holds also when we condition on the event $i > i^*$.

We remain with the case where $1 \leq i \leq i^*$. If $i = i^*$, in the real execution the output of P_2 is independent of x, and is determined according to $\mathsf{RandOut}_1(x)$. However, In the ideal execution, the simulator $\mathcal{S}$ queries the trusted party computing f on x, receives back $f(x,y)$, which determines the output of P_2 to be the correct output. Therefore, if $\mathcal{A}$ aborts exactly at i^*, in the real execution it learns the correct output while P_2 does not, and in the ideal execution *both* parties output the true output $f(x,y)$. Therefore, in order to simulate the protocol correctly, the simulator modifies the output of P_2 in case the adversary aborts before i^*, and chooses $\hat{x}$ according to some distribution X_{ideal}^{x,a_i} and not according to X_{real} as apparently expected. In the following, we show that if $X_{ideal}^{x,a_i} \cdot M_f = Q^{x,a_i}$, then Eq. (5) is satisfied.

Let $\vec{a} = (\vec{a}_{i-1}, a)$, and $\text{VIEW}_{\mathsf{hyb}}(x,y) = (\overrightarrow{\text{VIEW}}_{\mathsf{hyb}}^{i-1}, \text{VIEW}_{\mathsf{hyb}}^{i})$. We have that:

$$\begin{aligned}
&\Pr\left[(\text{VIEW}_{\mathsf{hyb}}(x,y), \text{OUT}_{\mathsf{hyb}}(x,y)) = (\vec{a}, b) \mid i \leq i^*\right] \\
&\quad = \Pr\left[(\text{VIEW}_{\mathsf{hyb}}^{i}, \text{OUT}_{\mathsf{hyb}}) = (a,b) \mid \overrightarrow{\text{VIEW}}_{\mathsf{hyb}}^{i-1} = \vec{a}_{i-1},\ i \leq i^*\right] \cdot \Pr\left[\overrightarrow{\text{VIEW}}_{\mathsf{hyb}}^{i-1} = \vec{a}_{i-1},\ i \leq i^*\right] \\
&\quad = \Pr\left[(\text{VIEW}_{\mathsf{hyb}}^{i}, \text{OUT}_{\mathsf{hyb}}) = (a,b) \mid i \leq i^*\right] \cdot \Pr\left[\overrightarrow{\text{VIEW}}_{\mathsf{hyb}}^{i-1} = \vec{a}_{i-1} \ \mid\ i \leq i^*\right]
\end{aligned}$$

where the last equations is true since conditioning on the even that $i \leq i^*$, the random variables $(\text{VIEW}^i_{\mathsf{hyb}}, \text{OUT}_{\mathsf{hyb}})$ are independent of $\overrightarrow{\text{VIEW}}^{i-1}_{\mathsf{hyb}}$. Similarly, write $\text{VIEW}_{\mathsf{ideal}}(x,y) = (\overrightarrow{\text{VIEW}}^{i-1}_{\mathsf{ideal}}, \text{VIEW}^i_{\mathsf{ideal}})$, we have:

$$\begin{aligned}
&\Pr\left[(\text{VIEW}_{\mathsf{ideal}}(x,y), \text{OUT}_{\mathsf{ideal}}(x,y)) = (\vec{a}, b) \mid i \leq i^*\right] \\
&= \Pr\left[(\text{VIEW}^i_{\mathsf{ideal}}, \text{OUT}_{\mathsf{ideal}}) = (a,b) \mid \overrightarrow{\text{VIEW}}^{i-1}_{\mathsf{ideal}} = \vec{a}_{i-1},\ i \leq i^*\right] \cdot \Pr\left[\overrightarrow{\text{VIEW}}^{i-1}_{\mathsf{ideal}} = \vec{a}_{i-1},\ i \leq i^*\right] \\
&= \Pr\left[(\text{VIEW}^i_{\mathsf{ideal}}, \text{OUT}_{\mathsf{ideal}}) = (a,b) \mid i \leq i^*\right] \cdot \Pr\left[\overrightarrow{\text{VIEW}}^{i-1}_{\mathsf{ideal}} = \vec{a}_{i-1} \ \mid\ i \leq i^*\right]
\end{aligned}$$

It is easy to see that for any values $\vec{a}_{i-1} \in \{0,1\}^{i-1}$:

$$\Pr\left[\overrightarrow{\text{VIEW}}^{i-1}_{\mathsf{ideal}} = \vec{a}_{i-1} \ \mid\ i \leq i^*\right] = \Pr\left[\overrightarrow{\text{VIEW}}^{i-1}_{\mathsf{hyb}} = \vec{a}_{i-1} \ \mid\ i \leq i^*\right]$$

since in both the simulation and the real execution, the adversary receives values according to $\mathsf{RandOut}_1(x)$.

It is left to show that for every $(a,b) \in \{0,1\}^2$, it holds that:

$$\Pr\left[(\text{VIEW}^i_{\mathsf{hyb}}, \text{OUT}_{\mathsf{hyb}}) = (a,b) \ \mid\ i \leq i^*\right] = \Pr\left[(\text{VIEW}^i_{\mathsf{ideal}}, \text{OUT}_{\mathsf{ideal}}) = (a,b) \ \mid\ i \leq i^*\right] \tag{6}$$

We have already seen it in the proof sketch in Section 3. We just give the high-level overview.

In case $f(x,y) = 0$. The probabilities are as follows:

output (a,b)	real	ideal
$(0,0)$	$(1-\alpha)\cdot(1-p_x)\cdot(1-p_y) + \alpha\cdot(1-p_y)$	$(1-\alpha)\cdot(1-p_x)\cdot(1-q_y^{x,0}) + \alpha$
$(0,1)$	$(1-\alpha)\cdot(1-p_x)\cdot p_y + \alpha\cdot p_y$	$(1-\alpha)\cdot(1-p_x)\cdot q_y^{x,0}$
$(1,0)$	$(1-\alpha)\cdot p_x\cdot(1-p_y)$	$(1-\alpha)\cdot p_x\cdot(1-q_y^{x,1})$
$(1,1)$	$(1-\alpha)\cdot p_x\cdot p_y$	$(1-\alpha)\cdot p_x\cdot q_y^{x,1}$

Therefore, we get the following constraints:

$$q_y^{x,0} = p_y + \frac{\alpha\cdot p_y}{(1-\alpha)\cdot(1-p_x)} \qquad \text{and} \qquad q_y^{x,1} = p_y\ ,$$

which are satisfied according to our assumption in the theorem.

In case $f(x,y) = 1$. Similarly, for the case of $f(x,y) = 1$, we have:

output (a,b)	real	ideal
$(0,0)$	$(1-\alpha)\cdot(1-p_x)\cdot(1-p_y)$	$(1-\alpha)\cdot(1-p_x)\cdot(1-q_y^{x,0})$
$(0,1)$	$(1-\alpha)\cdot(1-p_x)\cdot p_y$	$(1-\alpha)\cdot(1-p_x)\cdot q_y^{x,0}$
$(1,0)$	$(1-\alpha)\cdot p_x\cdot(1-p_y) + \alpha\cdot(1-p_y)$	$(1-\alpha)\cdot p_x\cdot(1-q_y^{x,1})$
$(1,1)$	$(1-\alpha)\cdot p_x\cdot p_y + \alpha\cdot p_y$	$(1-\alpha)\cdot p_x\cdot q_y^{x,1} + \alpha$

we again get the following constraints:

$$q_y^{x,0} = p_y \qquad \text{and} \qquad q_y^{x,1} = p_y + \frac{\alpha\cdot(p_y - 1)}{(1-\alpha)\cdot p_x}\ .$$

However, since $X^{x,a}_{ideal} \cdot M_f = Q^{x,a}$, the above constraints are satisfied. ■