Towards Characterizing Complete Fairness in Secure Two-Party Computation*

Gilad Asharov

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

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 amount* of functions can be computed with complete fairness. 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 and private matchmaking.

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 the 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).

Y. Lindell (Ed.): TCC 2014, LNCS 8349, pp. 291–316, 2014.

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 [14,6,8,25,13]. 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 [29,14,13]. 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 [9] 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 [17]. 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?

[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.

Recently, [3] 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 [17] 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.

To elaborate further, the work of [17] 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. Furthermore, it presents a generalization of this protocol that may potentially compute a large class of functions. It also shows how to construct a (rather involved) set of equations for a given function, that indicates whether the function can be computed fairly using this protocol.

These results are ground-breaking and completely change our perception regarding fairness. The fact that *something* non-trivial can be computed fairly is very surprising, it contradicts the aforementioned natural intuition and common belief 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 the 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 out 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, *almost all* functions with distinct domain sizes 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. 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 protocol of GHKL. This negative result does not rule out the possibility of these functions using some other protocols, however, it shows that the only known possibility result does not apply to this class of functions. Combining this result with [3] (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 [3], once we take off this advantage from the simulator – every function that contains an embedded XOR cannot be computed fairly, and fairness is almost always impossible.

Therefore, the algebraic 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 as follows:

(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

Fig. 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:

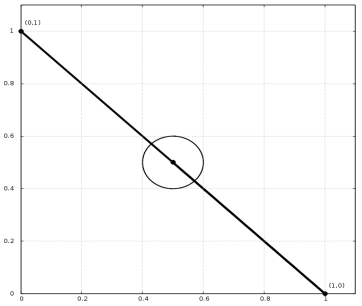
$$(q_1, q_2) = p_1 \cdot (0,1) + p_2 \cdot (1,0) + (1-p_1-p_2) \cdot (1,1) = (1-p_1, 1-p_2) \ .$$

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

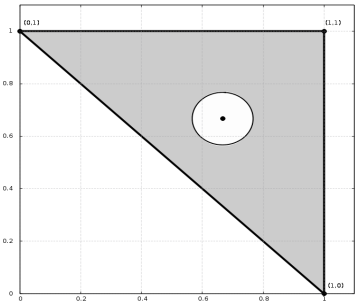
Geometrically, we can refer to the rows of the truth table as points in $\mathbb{R}^2$, and so in the XOR function we have the two points $(0,1)$ and $(1,0)$. All the output distribution vectors are of the form $p \cdot (0,1) + (1-p) \cdot (1,0)$ which is exactly the line segment between these two points (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 in 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 to bias 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. On the contrary, in 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 circuits 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 $(1,1)$

Fig. 2. The geometric objects defined by the XOR (a) and the GHKL (b) functions

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 ℓ points over $\mathbb{R}^m$, where the jth coordinate of the ith point is simply $f(x_i, y_j)$. 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. Under suitable cryptographic assumptions, if the geometric object defined by f is of full-dimension, then the function can be computed with complete fairness.*

For the proof, we simply use the extended GHKL protocol. 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 [17].

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). *Under suitable cryptographic assumptions,*

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 criteria 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 [12,1]. However, it seems that only Cleve's impossibility can be reduced into the family of boolean functions with finite domain. The work of [3] 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 ([10,15,5,19,4,21] to state a few. See [16] 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, it seems appealing to generalize our results 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 assume that the reader is familiar with the definitions of secure computation, and with the ideal-real paradigm. We distinguish between security-with-abort, for which the adversary may receive output while the honest party does not (security without fairness), and security with fairness, where all parties receive output (this is similar to security with respect to honest majority as in [7], although we do not have honest majority). In the following, we present the necessary notations, and we cover the mathematical background that is needed for our results.

Notations. 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 $\mathsf{poly}(\cdot)$ and all sufficiently large κ's it holds that $\mu(\kappa) < 1/\mathsf{poly}(\kappa)$. 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 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). All vectors will be assumed to be row vectors. 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 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 [26,20].

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 correct 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 $1 \leq i \leq \ell$. Then, the length m vector $\mathbf{q} = (q_{y_1}, \ldots, q_{y_m}) \stackrel{\text{def}}{=} \mathbf{p} \cdot M_f$ 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 $1 \leq j \leq m$, 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.

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.

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:

$$a_1 x_1 + \ldots a_m x_m = b \ ,$$

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 paper, 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 [17]

In the following, we give a high level overview of the protocol of [17]. 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.

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 abort 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 some 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 $a_i = f(x, \hat{y})$ for some random $\hat{y}$, and for P_2 it computes $b_i = f(\hat{x}, y)$ for some random $\hat{x}$. For every round $i \geq i^*$, the parties receive the correct output $a_i = b_i = f(x, y)$. In case one of the party initially aborts (i.e., does not invoke the online-dealer in the first round and the parties do not receive a_1, b_1), each party can locally compute initial outputs a_0, b_0 similarly to the way the values a_i, b_i are computed by the online-dealer for $i < i^*$. Note that if we set $R = \alpha^{-1} \cdot \omega(\ln \kappa)$, then $i^* < R$ with overwhelming probability, and so correctness holds.

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 $f(x, y)$, 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 $f(x, y)$ 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 $f(x, y)$ at round i^*, unlike the real execution where it outputs a random value $f(\hat{x}, y)$. [17] 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 $f(\hat{x}, y)$, 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} distribution exists, which depends on the function f being computed. Due to lack of space, we now show the definitions of the desired output distribution vectors Q^{x,a_i} without getting into the calculations for why these are defined like that. We refer the reader to [17] or the full version of this paper [2] to see how the protocol defines these requirements.

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 $Q^{x,a} = (q_{y_1}^{x,a}, \ldots, q_{y_m}^{x,a})$ indexed by $y_j \in Y$ as follows:

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

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

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

Theorem 3.1. *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 $X_{ideal}^{x,a}$ for which:*

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

then the protocol securely computes f with complete fairness.

An alternative formulation of the above, is to require that for every x, a, the points $Q^{x,a}$ are 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, p_{y_j}) rather than properties of the function. We are interested in a simpler and easier way to validate criteria.

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 [17]. A full dimensional function is defined as follows:

Definition 4.1 (full-dimensional function). *Let $f : X \times Y \to \{0,1\}$ be a function, and let $X_1, \ldots, X_\ell$ be the ℓ rows of M_f over $\mathbb{R}^m$. We say that f is* a full-dimensional function *if* $\dim(\mathbf{aff}(\{X_1, \ldots, X_\ell\})) = m$.

Recall that for a set of points $S = \{X_1, \ldots, X_\ell\} \in \mathbb{R}^m$, 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 $\ell > m$. Therefore, we assume without loss of generality that $\ell > m$ (and consider the transposed function $f^T : \{y_1, \ldots, y_m\} \times \{x_1, \ldots, x_\ell\} \to \{0,1\}$, defined as $f^T(y,x) = f(x,y)$, otherwise). Overall, our property inherently holds only if $\ell \neq m$.

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 [3] (we will elaborate more about these two criterions in Subsection 4.3). The proof for the following claim appears in the full version [2]:

Claim 4.2. *Let $f : \{x_1, \ldots, x_\ell\} \times \{y_1, \ldots, y_m\} \to \{0,1\}$ be a function, let M_f be as above and let $S = \{X_1, \ldots, X_\ell\}$ 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, $\mathbf{conv}(\{X_1, \ldots, X_\ell\}) \not\subseteq \mathcal{H}(\mathbf{q}, \delta)$.
3. The function is full-dimensional.
   There exists a subset of $\{X_1, \ldots, X_\ell\}$ of cardinality $m+1$, that is affinely independent. *Thus,* $\dim(\mathbf{aff}(\{X_1, \ldots, X_\ell\})) = m$.

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

The Proof of Possibility. We now show that any function that is full dimensional can be computed with complete fairness, using the protocol of [17]. The proof for this Theorem is geometrical. Recall that by Theorem 3.1, 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 3.1) are in the convex-hull of the rows $\{X_1, \ldots, X_\ell\}$, and therefore there exist probability vectors $X^{x,a}_{ideal}$ 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 $q^{x,a}_{y_j}$ 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 Section 1). 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. 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 shows 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). The proof for this claim is a simple implication of the separation theorems for convex sets, see [26]. 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 locate $\mathbf{c}$ to be inside the simplex that is defined by S', by choosing X_{real} to be the uniform distribution over S' (i.e., the ith position of X_{real} is 0 if $X_i \notin S'$, and $1/(m+1)$ if $X_i \in S'$). We then let $\mathbf{c} = (p_{y_1}, \ldots, p_{y_m}) = X_{real} \cdot M_f$. Finally, we set $\alpha = 1/\ln \kappa$. We consider the GHKL protocol with the above parameters, and consider the set of points $\{Q^{x,a}\}_{x \in X, a \in \{0,1\}}$. 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. In a similar way, 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:

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

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

We now show that this 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[2] 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 $B(\mathbf{c}, \epsilon) \subset \mathbf{conv}(S')$. ∎

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 3.1 are satisfied, and the protocol securely computes f with complete fairness. ∎

[2] The convex-hull of a finite set S of vectors in $\mathbb{R}^m$ is a compact set, and therefore is closed (See [26, Theorem 15.4]).

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 simply $1 - P_m$, 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 $(1/2+o(1))^m$, and recent results [23,22,28] 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. 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 [27]. The following Claim is based on [30, 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: An alternative question for the first item is 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 first 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

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 some additional requirements, the protocol of [17] cannot be simulated for any choice of parameters, with respect to

the specific simulation strategy defined in the proof of Theorem 3.1. 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 3.1.

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)):

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

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:

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

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:

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

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 [3], 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.

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.

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 [17]. As we will see, only negligible amount of functions with $|X| = |Y|$ do not satisfy these additional requirements. Thus, our characterization of [17] 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 [17]. 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

0 is negligible. From similar arguments, the probability that the hyperplane that is defined by the columns of the matrix contains either **1** or **0** is also negligible. ∎

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 **0** or **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 3.1 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 3.1 follows for f as well. ∎

The above implies an interesting and 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.

4.3 Conclusion: Symmetric Boolean Functions with Finite Domain

We summarize all the known results in complete fairness for symmetric boolean functions with finite domain, and we link our results to the balanced property of [3].

Characterization of Coin-tossing [3]. The work of Asharov, Lindell and Rabin [3] considers the task of coin-tossing, which was shown to be impossible to compute fairly [9]. 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 [24,11]. The property that [3] has defined is as follows:

Definition 4.12 (strictly-balanced property [3]). *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, [3] 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.

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 [3] with ours, 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 [3]:**
   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:

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

   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:

$$\mathbf{p} \cdot M_f = \delta_1 \cdot \mathbf{1}_m \qquad \text{and} \qquad M_f \cdot \mathbf{q}^T = \delta_2 \cdot \mathbf{1}_\ell^T$$

   then we say that the function is balanced with respect to arbitrary vectors. Then, the function does not (information-theoretically) imply fair-coin tossing [3]. 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.

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, thus, nothing is assumed about f_1.

In the full version of this paper [2], we revise the protocol of [17] to deal with this class of functions. This is done in a straightforward way, where the online dealer computes at each round the value a_i according to the function f_1, and b_i according to f_2. We then derive a set of equations, similarly to Eq. (1) and obtain an analogue theorem to Theorem 3.1. We then show the following Corollary:

Corollary 5.1. *Let $f : \{x_1,\ldots,x_\ell\} \times \{y_1,\ldots,y_m\} \to \{0,1\} \times \{0,1\}$, where $f = (f_1, f_2)$. If f_2 is a full-dimensional function, then f can be computed with complete fairness.*

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 $\Sigma = \{\sigma_1,\ldots,\sigma_k\}$ be an alphabet for some finite $k > 0$, 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. However, the analysis for this case is more involved. For instance, in the binary case for every input $y_j \in Y$, we considered the parameter p_{y_j}, the probability that P_2 receives 1 in each round before i^* when its input is y_j. In the non-binary case, we have to define an equivalent parameter $p_{y_j}(\sigma)$ for *any* symbol σ in the alphabet Σ (i.e., the probability that P_2 receives σ in each round before i^* when its input is y_j). This makes things harder, and in order to obtain fairness, several requirements should be satisfied simultaneously for every $\sigma \in \Sigma$.

In order to see this, fix some X_{real}. For any symbol $\sigma \in \Sigma$, and for every $y_j \in Y$, let $p_{y_j}(\sigma)$ denote the probability that b_i is σ (when $i \leq i^*$). That is:

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

Observe that $\sum_{\sigma \in \Sigma} p_{y_j}(\sigma) = 1$. For every $\sigma \in \Sigma$, we want to represent 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 M_f does not represent exactly what we want, and the multiplication $X_{real} \cdot M_f$ gives the "expected output distribution vector" and not exactly what we want. Instead, for any $\sigma \in \Sigma$ we define the binary matrix M_f^σ as follows:

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

Now, we can represent the vector $(p_{y_1}(\sigma), \ldots, p_{y_m}(\sigma))$ as $X_{real} \cdot M_f^\sigma$. However, here a *single* vector X_{real} determines the values of $|\Sigma|$ vectors, one for each $\sigma \in \Sigma$. Therefore, we overall get $|\Sigma|$ systems of equations, one for each symbol in the alphabet. In [2] we show that it is enough to consider only $|\Sigma| - 1$ systems since our probabilities sum-up to 1 (i.e., $\sum_{\sigma \in \Sigma} p_{y_j}(\sigma) = 1$), and provide the sets of equations that guarantees fairness. In the following, we provide a corollary of our result which provides a simpler criterion.

Given 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 $|Y^{\Sigma_\rho}| = (|\Sigma| - 1) \cdot |Y|$. We show that if the boolean function f' is full-dimensional, then the function f can be computed with complete-fairness. Observe that this property can 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:

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

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

References

1. Agrawal, S., Prabhakaran, M.: On fair exchange, fair coins and fair sampling. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 259–276. Springer, Heidelberg (2013)
2. Asharov, G.: Towards characterizing complete fairness in secure two-party computation. IACR Cryptology ePrint Archive (to appear)
3. Asharov, G., Lindell, Y., Rabin, T.: A full characterization of functions that imply fair coin tossing and ramifications to fairness. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 243–262. Springer, Heidelberg (2013)
4. Beimel, A., Lindell, Y., Omri, E., Orlov, I.: $1/p$-secure multiparty computation without honest majority and the best of both worlds. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 277–296. Springer, Heidelberg (2011)
5. Ben-Or, M., Goldreich, O., Micali, S., Rivest, R.L.: A fair protocol for signing contracts. IEEE Transactions on Information Theory 36(1), 40–46 (1990)
6. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC, pp. 1–10 (1988)
7. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptology 13(1), 143–202 (2000)
8. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: STOC, pp. 11–19 (1988)
9. Cleve, R.: Limits on the security of coin flips when half the processors are faulty (extended abstract). In: STOC, pp. 364–369 (1986)
10. Cleve, R.: Controlled gradual disclosure schemes for random bits and their applications. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 573–588. Springer, Heidelberg (1990)
11. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. In: CRYPTO, pp. 205–210 (1982)
12. Even, S., Yacobi, Y.: Relations among public key signature schemes. Technical Report #175, Technion Israel Institute of Technology, Computer Science Department (1980), `http://www.cs.technion.ac.il/users/wwwb/cgi-bin/trinfo.cgi/1980/CS/CS0175`
13. Goldreich, O.: The Foundations of Cryptography - Basic Applications, vol. 2. Cambridge University Press (2004)
14. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229 (1987)
15. Goldwasser, S., Levin, L.: Fair computation of general functions in presence of immoral majority. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 77–93. Springer, Heidelberg (1991)
16. Gordon, S.D.: On fairness in secure computation. PhD thesis, University of Maryland (2010)
17. Gordon, S.D., Hazay, C., Katz, J., Lindell, Y.: Complete fairness in secure two-party computation. In: STOC, pp. 413–422 (2008); Extended full version available on: `http://eprint.iacr.org/2008/303`. Journal version: [18]
18. Gordon, S.D., Hazay, C., Katz, J., Lindell, Y.: Complete fairness in secure twoparty computation. J. ACM 58(6), 24 (2011)
19. Gordon, S.D., Katz, J.: Partial fairness in secure two-party computation. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 157–176. Springer, Heidelberg (2010)

20. Grünbaum, B.: Convex Polytopes. In: Kaibel, V., Klee, V., Ziegler, G. (eds.) Graduate Texts in Mathematics, 2nd edn. Springer (May 2003)
21. Halpern, J.Y., Teague, V.: Rational secret sharing and multiparty computation: extended abstract. In: STOC, pp. 623–632 (2004)
22. Kahn, J., Komlòs, J., Szemerèdi, E.: On the probability that a random ±1-matrix is singular. Journal of Amer. Math. Soc. 8, 223–240 (1995)
23. Komlòs, J.: On the determinant of $(0, 1)$ matrices. Studia Sci. Math. Hungar 2, 7–21 (1967)
24. Rabin, M.O.: How to exchange secrets with oblivious transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University (1981)
25. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: STOC, pp. 73–85 (1989)
26. Roman, S.: Advanced Linear Algebra, 3rd edn. Graduate Texts in Mathematics, vol. 135, p. xviii. Springer, New York (2008)
27. Voigt, T., Ziegler, G.M.: Singular 0/1-matrices, and the hyperplanes spanned by random 0/1-vectors. Combinatorics, Probability and Computing 15(3), 463–471 (2006)
28. Wood, P.J.: On the probability that a discrete complex random matrix is singular. PhD thesis, Rutgers University, New Brunswick, NJ, USA, AAI3379178 (2009)
29. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: FOCS, pp. 162–167 (1986)
30. Ziegler, G.M.: Lectures on 0/1-polytopes. Polytopes: Combinatorics and Computation, Birkhauser, Basel. DMV Seminar, vol. 29, pp. 1–40 (2000)