A Full Characterization of Functions that Imply Fair Coin Tossing and Ramifications to Fairness*

Gilad Asharov† Yehuda Lindell† Tal Rabin‡

February 25, 2013

Abstract

It is well known that it is impossible for two parties to toss a coin fairly (Cleve, STOC 1986). This result implies that it is impossible to securely compute with fairness any function that can be used to toss a coin fairly. In this paper, we focus on the class of deterministic Boolean functions with finite domain, and we ask for which functions in this class is it possible to information-theoretically toss an unbiased coin, given a protocol for securely computing the function with fairness. We provide a *complete characterization* of the functions in this class that imply and do not imply fair coin tossing. This characterization extends our knowledge of which functions cannot be securely computed with fairness. In addition, it provides a focus as to which functions may potentially be securely computed with fairness, since a function that cannot be used to fairly toss a coin is not ruled out by the impossibility result of Cleve (which is the *only* known impossibility result for fairness). In addition to the above, we draw corollaries to the feasibility of achieving fairness in two possible fail-stop models.

Keywords: Fairness, coin tossing, secure computation, malicious and fail-stop adversaries

*An extended abstract of this work appeared at *TCC 2013*. The first two authors were funded 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).

†Dept. of Computer Science, Bar-Ilan University, Israel. email: asharog@cs.biu.ac.il, lindell@biu.ac.il. Much of this work was carried out while at the IBM T.J. Watson Research Center, New York.

‡IBM T.J. Watson Research Center, New York. email: talr@us.ibm.com

1 Introduction

Background. In the setting of secure multiparty computation, some mutually distrusting parties wish to compute some joint function of their inputs in the presence of adversarial behaviour. Loosely speaking, the security requirements from such a computation are that nothing is learned from the protocol other than the output (privacy), that the output is distributed according to the prescribed functionality (correctness), and that parties cannot choose their inputs as a function of the others inputs (independence of inputs). Another important property is that of *fairness* which, intuitively, means that either *everyone* receives the output or *no one* does.

It is well known that when a majority of the parties are honest, it is possible to securely compute any functionality while guaranteeing all of the security properties mentioned above, including fairness [7, 1, 3, 10]. Furthermore, when there is no honest majority, including the important case of two parties where one may be corrupted, it is possible to securely compute any functionality while guaranteeing all of the security properties mentioned above *except for fairness* [12, 7, 5]. The fact that fairness is not achieved in this latter case is inherent, as was shown in the seminal work of Cleve [4] who proved that there exist functions that cannot be computed by two parties with complete fairness. Specifically, Cleve showed that the very basic and natural functionality of coin-tossing, where two parties toss an unbiased coin, cannot be computed fairly. The impossibility result of Cleve implies that fairness cannot be achieved *in general*. That is, Cleve's result proves that it is impossible to securely compute with complete fairness any function that can be used to toss a fair coin (like the boolean XOR function).

Until recently, the accepted folklore from the time of Cleve's result was that *only trivial functions* can be securely computed with complete fairness without an honest majority. This changed recently with the surprising work of Gordon et al. [8] who showed that this folklore is incorrect and that there exist some non-trivial boolean functions that *can* be computed fairly, in the two party setting. They showed 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 in the malicious settings. Examples of functions without an embedded XOR include the boolean OR and AND functions and Yao's millionaires' problem [12] (i.e., the greater-than function). This possibility result completely changes our understanding regarding fairness, and re-opens the question of which functions can be computed with complete fairness. Given the possibility result mentioned above, and given the fact that Cleve's impossibility result rules out completely fair computation of boolean XOR, a natural conjecture is that the presence of an embedded XOR serves as a barrier to completely fair computation of a given function. However, [8] showed that this is also incorrect: they give an example of a function that *does* contain an embedded XOR and construct a protocol that securely computes this function with fairness.

Since [8], there have been no other works that further our understanding regarding which (boolean) functions can be computed fairly without an honest majority in the two party setting. Specifically, Cleve's impossibility result is the only known function that cannot be computed fairly, and the class of functions for which [8] 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.

Our work. Motivated by the fundamental question of characterizing which functions can be computed with complete fairness, we analyze which functions *cannot* be computed fairly since they

are already ruled out by Cleve's original result. That is, we show which finite-domain boolean functions "imply" the coin-tossing functionality. We provide a simple property (criterion) on the truth table of a given boolean function. We then show that for every function that satisfies this property, it holds that the existence of a protocol that fairly computes the given function implies the existence of a protocol for fair coin-tossing in the presence of a fail-stop adversary, in contradiction to Cleve's impossibility result. This implies that the functions that satisfy the property cannot be computed fairly. The property is very simple, clean and general.

The more challenging and technically interesting part of our work is a proof that the property is *tight*. Namely, we show that a function f that does not satisfy the property *cannot* be used to construct a fair coin-tossing protocol (in the information theoretic setting). More precisely, we show that it is impossible to construct a fair two-party coin-tossing protocol, even if the parties are given access to a trusted party that computes f *fairly* for them. We prove this impossibility by showing the existence of an (inefficient) adversary that can bias the outcome with non-negligible probability. Thus, we prove that it is not possible to toss a coin with information-theoretic security, when given access to fair computations of f. We stress that this "impossibility" result is actually a source of optimism, since it *may* be possible to securely compute such functions with complete fairness. Indeed, the fair protocols presented in [8] are for functions for which the property does not hold.[1]

It is important to note that our proof that functions that do not satisfy the property do not imply coin tossing is very different to the proof of impossibility by Cleve. Specifically, the intuition behind the proof by Cleve is that since the parties exchange messages in turn, there must be a point where one party has more information than the other about the outcome of the coin-tossing protocol. If that party aborts at this point, then this results in bias. This argument holds since the parties cannot exchange information simultaneously. In contrast, in our setting, the parties *can* exchange information simultaneously via the computation of f. Thus, our proof is conceptually very different to that of Cleve, and in particular, is not a reduction to the proof by Cleve.

The criterion. Intuitively, the property that we define over the function's truth table relates to the question of whether or not it is possible for one party to singlehandedly change the probability that the output of the function is 1 (or 0) based on how it chooses its input. In order to explain the criterion, we give two examples of functions that imply coin-tossing, meaning that a fair secure protocol for computing the function implies a fair secure protocol for coin tossing. We discuss how each of the examples can be used to toss a coin fairly, and this in turn will help us to explain the criterion. The functions are given below:

(a)

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

(b)

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

Consider function (a), and assume that there exists a fair protocol for this function. We show how to toss a fair coin using a single invocation of the protocol for f. Before doing so, we observe

[1] We remark that since our impossibility result is information theoretic, there is the possibility that some of the functions for which the property does not hold do imply coin tossing computationally. In such a case, the impossibility result of Cleve still applies to them. See more discussion in "open questions" below.

that the output of a single invocation of the function can be expressed by multiplying the truth-table matrix of the function by probability vectors. Specifically, assume that party P_1 chooses input x_i with probability p_i, for $i = 1, 2, 3$ (thus $p_1 + p_2 + p_3 = 1$ since it must choose some input); likewise, assume that P_2 chooses input y_i with probability q_i. Now, let M_f be the "truth table" of the function, meaning that $M_f[i, j] = f(x_i, y_j)$. Then, the output of the invocation of f upon the inputs chosen by the parties equals 1 with probability exactly $(p_1, p_2, p_3) \cdot M_f \cdot (q_1, q_2, q_3)^T$.

We are now ready to show how to toss a coin using f. First, note that there are two complementary rows; these are the rows specified by inputs x_1 and x_2. This means that if P_1 chooses one of the inputs in $\{x_1, x_2\}$ uniformly at random, then no matter what distribution over the inputs (corrupted) P_2 uses, the result is a uniformly chosen coin. In order to see this, observe that when we multiply the vector $(\frac{1}{2}, \frac{1}{2}, 0)$ (the distribution over the input of P_1) with the matrix M_f, the result is the vector $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. This means that no matter what input P_2 will choose, or what distribution over the inputs it may use, the output is 1 with probability $1/2$ (formally, the output is 1 with probability $\frac{1}{2} \cdot q_1 + \frac{1}{2} \cdot q_2 + \frac{1}{2} \cdot q_3 = \frac{1}{2}$ because $q_1 + q_2 + q_3 = 1$). This means that if P_1 is honest, then a corrupted P_2 cannot bias the output. Likewise, there are also two complementary columns (y_1 and y_3), and thus, if P_2 chooses one of the inputs in $\{y_1, y_3\}$ uniformly at random, then no matter what distribution over the inputs (a possibly corrupted) P_1 uses, the result is a uniform coin.

In contrast, there are no two complementary rows or columns in the function (b). However, if P_1 chooses one of the inputs $\{x_1, x_2, x_3\}$ uniformly at random (i.e., each input with probability one third), then no matter what distribution P_2 will use, the output is 1 with probability $1/3$. Similarly, if P_2 chooses a uniformly random input, then no matter what P_1 does, the output is 1 with the same probability. Therefore, a single invocation of the function f in which the honest party chooses the uniform distribution over its inputs results in a coin that equals 1 with probability exactly $\frac{1}{3}$, irrespective of what the other party inputs. In order to obtain an unbiased coin that equals 1 with probability $\frac{1}{2}$ the method of von-Neumann [11] can be used. This method works by having the parties use the function f to toss two coins. If the resulting coins are different (i.e, 01 or 10), then they output the result of the first invocation. Otherwise, they run the protocol again. This yields a coin that equals 1 with probability $\frac{1}{2}$ since the probability of obtaining 01 equals the probability of obtaining 10. Thus, conditioned on the results being different, the probability of outputting 0 equals the probability of outputting 1.

The criterion is a simple generalization of the examples shown above. Let $f : \{x_1, \ldots, x_\ell\} \times \{y_1, \ldots, y_\ell\} \to \{0, 1\}$ be a function, and let M_f be the truth table representation as described above. We look for the existence of two probability vectors[2], $\mathbf{p} = (p_1, \ldots, p_m)$, $\mathbf{q} = (q_1 \ldots, q_\ell)$ such that $\mathbf{p} \cdot M_f$ and $M_f \cdot \mathbf{q}^T$ are both vectors that equal δ everywhere, for some $0 < \delta < 1$. Observe that if such probability vectors exist, then the function implies the coin-tossing functionality as we described above. Specifically, P_1 chooses its input according to distribution $\mathbf{p}$, and P_2 chooses its inputs according to the distribution $\mathbf{q}$. The result is then a coin that equals 1 with probability δ. Using the method of von-Neumann, this can be used to obtain a uniformly distributed coin. We conclude:

Theorem 1.1 (informal) *Let $f : \{x_1, \ldots, x_m\} \times \{y_1, \ldots, y_\ell\} \to \{0, 1\}$ be a function that satisfies the aforementioned criterion. Then, the existence of a protocol for securely computing f with complete fairness implies the existence of a fair coin tossing protocol.*

[2] $\mathbf{p} = (p_1, \ldots, p_m)$ is a probability vector if $p_i \geq 0$ for every $1 \leq i \leq m$, and $\sum_{i=1}^{m} p_i = 1$.

An immediate corollary of this theorem is that any such function cannot be securely computed with complete fairness, or this contradicts the impossibility result of Cleve [4].

As we have mentioned above, the more interesting and technically challenging part of our work is a proof that the criterion is tight. That is, we prove the following theorem:

Theorem 1.2 (informal) *Let $f : \{x_1, \ldots, x_m\} \times \{y_1, \ldots, y_\ell\} \to \{0, 1\}$ be a function that does* not *satisfy the aforementioned criterion. Then, there exists an exponential-time adversary that can bias the outcome of every coin-tossing protocol that uses ideal and fair invocations of f.*

This result has a number of ramifications. Most notably, it provides a focus of our attention for the question of fairness in two-party secure computation. Specifically, the only functions that can potentially be securely computed with fairness are those for which the property does not hold. These functions have the property that one of the parties can partially influence the outcome of the result singlehandedly, a fact that is used inherently in the protocol of [8] for the function with an embedded XOR. This does not mean that all functions of this type can be fairly computed. However, it provides a good starting point. In addition, our results define the set of functions for which Cleve's impossibility result suffices for proving that they cannot be securely computed with fairness. Given that no function other than those implying coin tossing has been ruled out since Cleve's initial result, understanding exactly what is included in this impossibility is of importance.

On fail-stop adversaries. Our main results above consider the case of malicious adversaries. In addition, we explore the fail-stop adversary model where the adversary follows the protocol like an honest party, but can halt early. This model is of interest since the impossibility result of Cleve [4] for achieving fair coin tossing holds also for fail-stop adversaries. In order to prove theorems regarding the fail-stop model, we first provide a definition of security with complete fairness for fail-stop adversaries that follows the real/ideal simulation paradigm. Surprisingly, this turns out not to be straightforward and we provide two natural formulations that are very different regarding feasibility. The formulations differ regarding the ideal-world adversary/simulator. The question that arises is whether or not the simulator is allowed to use a different input to the prescribed one. In the semi-honest model (which differs only in the fact that the adversary cannot halt early) the standard formulation is to not allow the simulator to change the prescribed input, whereas in the malicious model the simulator is always allowed to change the prescribed input. We therefore define two fail-stop models. In this first, called "fail-stop1", the simulator is allowed to either send the trusted party computing the function the prescribed input of the party or an abort symbol $\perp$, but nothing else. In the second, called "fail-stop2", the simulator may send any input that it wishes to the trusted party computing the function. (Note, however, that if there was no early abort then the prescribed input must be used because such an execution is identical to an execution between two honest parties.)

Observe that in the first model, the honest party is guaranteed to receive the output on the prescribed inputs, unless it receives abort. In addition, observe that any protocol that is secure in the presence of malicious adversaries is secure also in the fail-stop2 model. However, this is not true of the fail-stop1 model (this is due to the fact that the simulator in the ideal model for the case of malicious adversaries is more powerful that in the fail-stop2 ideal model since the former can send any input whereas the latter can only send the prescribed input or $\perp$).

We remark that Cleve's impossibility result holds in both models, since the parties do not have inputs in the coin-tossing functionality, and therefore there is no difference in the ideal-worlds of

the models in this case. In addition, the protocols of [8] that are secure for malicious adversaries are secure for fail-stop2 (as mentioned above, this is immediate), but are *not* secure for fail-stop1.

We show that in the fail-stop1 model, it is impossible to securely compute with complete fairness any function containing an embedded XOR. We show this by constructing a coin-tossing protocol from any such function, that is secure in the fail-stop model. Thus the only functions that can potentially be securely computed with fairness are those with no embedded XOR but with an embedded OR (if a function has neither, then it is trivial and can be computed unconditionally and fairly); we remark that there are very few functions with this property. We conclude that in the fail-stop1 model, fairness cannot be achieved for almost all non-trivial functions. We remark that [8] presents secure protocols that achieve complete fairness for functions that have no embedded XOR; however, they are not secure in the fail-stop1 model, as mentioned.

Regarding the fail-stop2 model, we prove an analogous result to Theorem 1.2. In the proof of Theorem 1.2, the adversary that we construct changes its input in one of the invocations of f and then continues honestly. Thus it is malicious and not fail-stop2. Nevertheless, we show how the proof can be modified in order to hold for the fail-stop2 model as well. We then show how we can modify the proof to hold in the fail-stop2 model as well.

These extensions for fail-stop adversaries deepen our understanding regarding the feasibility of obtaining fairness. Specifically, any protocol that achieves fairness for any non-trivial function (or at least any function that has an embedded XOR), must have the property that the simulator can send any input in the ideal model. Stated differently, the input that is effectively used by a corrupted party cannot be somehow committed, thereby preventing this behaviour. This also explains why the protocols of [8] have this property.

Open questions. In this work we provide an almost complete characterization regarding what functions imply and do not imply coin tossing. Our characterisation is not completely tight since the impossibility result of Theorem 1.2 only holds in the information-theoretic setting; this is due to the fact that the adversary needs to carry out inefficient computations. Thus, it is conceivable that coin tossing can be achieved computationally from some such functions. It is important to note, however, that any function that does not fulfil our criterion implies oblivious transfer (OT). Thus, any protocol that uses such a function has access to OT and all that is implied by OT (e.g., commitments, zero knowledge, and so on). Thus, any such computational construction would have to be inherently nonblack-box in some sense. Our work also only considers finite functions (where the size of the domain is not dependent on the security parameter); extensions to other function classes, including non-Boolean functions, is also of interest.

The main open question left by our work is to characterize which functions for which the criterion does not hold can be securely computed with complete fairness. Our work is an important step to answering this question by providing a clearer focus than was previously known. Observe that in order to show that a function that does not fulfil the criterion cannot be securely computed with complete fairness, a new impossibility result must be proven. In particular, it will not be possible to reduce the impossibility to Cleve [4] since such a function does not imply coin tossing.

2 Definitions and Preliminaries

We let n denote the security parameter. A function $\mu(\cdot)$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large n it holds that $\mu(n) < 1/p(n)$. A *distribution ensemble*

$X = \{X(a,n)\}_{a\in\mathcal{D},n\in\mathbb{N}}$ is an infinite sequence of random variables indexes by $a \in \mathcal{D}$ and $n \in \mathbb{N}$. In the context of secure computation, n is the security parameter and $\mathcal{D}$ denotes the domain of the parties' input. Two distribution ensembles $X = \{X(a,n)\}_{a\in\mathcal{D},n\in\mathbb{N}}$ and $Y = \{Y(a,n)\}_{a\in\mathcal{D},n\in\mathbb{N}}$ are *computationally indistinguishable*, denoted $X \stackrel{c}{\equiv} Y$, if for every non-uniform polynomial-time-algorithm D there exists a negligible function $\mu(\cdot)$ such that for every n and every $a \in \mathcal{D}$:

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

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 $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_\ell\}$, for constants $m, \ell \in \mathbb{N}$.

2.1 Secure Two-Party Computation with Fairness – Malicious Adversaries

We now define what it means for a protocol to be *secure with complete fairness*. Our definition follows the standard definition of [2, 5], 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 function $f : X \times Y \to \{0,1\}$ over a finite domain. In this work, we restrict our attention to functions that return the same output to both parties. 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^n 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 $f(x', y')$ and sends the result to P_1 and 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 $\text{IDEAL}_{f,\mathcal{S}(z)}(x, y, n)$ 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 $\text{REAL}_{\pi,\mathcal{A}(z)}(x, y, n)$ 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^n.

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:*

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

Protocol π computes f with statistical security *if for every $\mathcal{A}$ there exists an adversary $\mathcal{S}$ such that the* IDEAL *and* REAL *distributions are statistically close.*

Function implication. In the paper, we study whether or not a function f "implies" the coin-tossing functionality. We now formally define what we mean by "function implication". Our formulation uses the notion of a hybrid model, which is a combination of the ideal and real models described above. Specifically, let f be a function. Then, an execution in the f-hybrid model consists of real interaction between the parties (like in the real model) and ideal invocations of f (like in the ideal model). The ideal invocations of f take place via a trusted party that receives inputs and computes the output of f on those inputs, exactly like in the ideal model. Thus, since in our ideal model both parties receive the output of f simultaneously (i.e., it is fair), it follows that in the f-hybrid model, both parties receive the output of f simultaneously. We are now ready for the definition.

Definition 2.2 *Let $f : X \times Y \to Z$ and $g : X' \times Y' \to Z'$ be functions. We say that* function *f* implies function *g* in the presence of malicious adversaries *if there exists a protocol that securely computes g in the f-hybrid model with complete fairness, in the presence of static malicious adversaries. We say that f* information-theoretically *implies g if the above holds with statistical security.*

Note that if g can be securely computed with fairness (under some assumption), then every function f computationally implies g. Thus, this is only of interest for functions g that either cannot be securely computed with fairness, or for which this fact is not known.

2.2 Coin-Tossing Definitions

The coin-tossing functionality. We define the coin-tossing functionality simply by $f^{\mathsf{ct}}(\lambda,\lambda) = (U_1, U_1)$, where λ denotes the empty input and U_1 denotes the uniform distribution over $\{0,1\}$. That is, the functionality receives no input, chooses a uniformly chosen bit and gives the both parties the same bit. This yields the following definition:

Definition 2.3 (Coin-Tossing by Simulation) *A protocol π is a* secure coin-tossing protocol via simulation *if it securely computes f^{ct} with complete fairness in the presence of malicious adversaries.*

The above definition provides very strong simulation-based guarantees, which is excellent for our positive results. However, when proving impossibility, it is preferable to rule out even weaker, non-simulation based definitions. We now present a weaker definition where the guarantee is that

the honest party outputs an unbiased coin, irrespective of the cheating party's behaviour. However, we stress that since our impossibility result only holds with respect to an all-powerful adversary (as discussed in the introduction), our definition is stronger than above since it requires security in the presence of any adversary, and not just polynomial-time adversaries.

Notations. Denote by $\langle P_1, P_2 \rangle$ a two party protocol where both parties act honestly. Let $\mathsf{out}\langle P_1, P_2 \rangle$ denote the output of parties in an honest execution (if the outputs of the parties is not consistent, this predicate returns $\perp$). For $\ell \in \{1,2\}$, let $\mathsf{out}_\ell\langle P_1^*, P_2^* \rangle$ denote the output of party P_ℓ^* in an execution of P_1^* with P_2^*. In some cases, we also specify the random coins that the parties use in the execution; $\langle P_1(r_1), P_2(r_2) \rangle$ denotes an execution where P_1 acts honestly and uses random tape r_1 and P_2 acts honestly and uses random tape r_2. Let $r(n)$ be a polynomial that bounds the number of rounds of the protocol π, and let $c(n)$ be an upper bound on the length of the random tape of the parties. Let Uni denote the uniform distribution over $\{0,1\}^{c(n)} \times \{0,1\}^{c(n)}$. We are now ready to define a coin-tossing protocol:

Definition 2.4 (information-theoretic coin-tossing protocol) *A polynomial-time protocol $\pi = \langle P_1, P_2 \rangle$ is an* unbiased coin-tossing protocol*, if the following hold:*

1. Agreement: *There exists a negligible function $\mu(\cdot)$ such that for every n it holds that:*

$$\Pr_{r_1, r_2 \leftarrow \mathsf{Uni}} \left[\mathsf{out}_1\langle P_1(r_1), P_2(r_2) \rangle \neq \mathsf{out}_2\langle P_1(r_1), P_2(r_2) \rangle \right] \leq \mu(n) \ .$$

2. No bias: *For every adversary $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that for every $b \in \{0,1\}$ and every $n \in \mathbb{N}$:*

$$\Pr\left[\mathsf{out}_1\langle P_1, \mathcal{A} \rangle = b\right] \leq \frac{1}{2} + \mu(n) \quad \text{and} \quad \Pr\left[\mathsf{out}_2\langle \mathcal{A}, P_2 \rangle = b\right] \leq \frac{1}{2} + \mu(n) \ .$$

Observe that both requirements together guarantee that two honest parties will output the same uniformly distributed bit, except with negligible probability. That is, for every $b \in \{0,1\}$:

$$\left| \Pr_{r_1, r_2 \leftarrow \mathsf{Uni}} [\mathsf{out}\langle P_1(r_1), P_2(r_2) \rangle = b] - \frac{1}{2} \right| \leq \mu(n) \tag{1}$$

3 The Criterion

In this section we define the criterion, and explore its properties. We start with the definition of δ-balanced functions.

3.1 δ-Balanced Functions

A vector $\mathbf{p} = (p_1, \ldots, p_k)$ is a probability vector if $\sum_{i=1}^{k} p_i = 1$, and for every $1 \leq i \leq k$ it holds that $p_i \geq 0$. Let $\mathbf{1}_k$ be the all one vector of size k. In addition, for a given function $f : \{x_1, \ldots, x_m\} \times \{y_1, \ldots, y_\ell\} \to \{0,1\}$, let M_f denote the matrix defined by the truth table of f. That is, for every $1 \leq i \leq m$, $1 \leq j \leq \ell$, it holds that $M_f[i,j] = f(x_i, y_j)$.

Informally, a function is balanced if there exist probabilities over the inputs for each party that determine the probability that the output equals 1, irrespective of what input the other party uses. Assume that P_1 chooses its input according to the probability vector $(p_1, \ldots, p_m)$, meaning that it uses input x_i with probability p_i, for every $i = 1, \ldots, m$, and assume that party P_2 uses the jth input y_j. Then, the probability that the output equals 1 is obtained by multiplying $(p_1, \ldots, p_m)$ with the jth column of M_f. Thus, a function is balanced on the left, or with respect to P_1, if when multiplying $(p_1, \ldots, p_m)$ with M_f the result is a vector with values that are all equal. Formally:

Definition 3.1 *Let $f : \{x_1, \ldots, x_m\} \times \{y_1, \ldots, y_\ell\} \to \{0, 1\}$ be a function, and let $0 \leq \delta_1, \delta_2 \leq 1$ be constants. We say that f is δ_1-*left-balanced *if there exists a probability vector $\mathbf{p} = (p_1, \ldots, p_m)$ such that:*

$$(p_1, \ldots, p_m) \cdot M_f = \delta_1 \cdot \mathbf{1}_\ell = (\delta_1, \ldots, \delta_1) \ .$$

*Likewise, we say that the function f is δ_2-*right-balanced *if there exists a probability vector $\mathbf{q} = (q_1, \ldots, q_\ell)$ such that:*

$$M_f \cdot (q_1, \ldots, q_\ell)^T = \delta_2 \cdot \mathbf{1}_m^T \ .$$

*If f is δ_1-left-balanced and δ_2-right-balanced, we say that f is (δ_1, δ_2)-balanced. If $\delta_1 = \delta_2$, then we say that f is δ-*balanced*, where $\delta = \delta_1 = \delta_2$. We say that f is* strictly δ-balanced *if $\delta_1 = \delta_2$ and $0 < \delta < 1$.*

Note that a function may be δ_2-right-balanced for some $0 \leq \delta_2 \leq 1$ but not left balanced. For example, consider the function defined by the truth table $M_f \stackrel{\text{def}}{=} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$. This function is right balanced for $\delta_2 = \frac{1}{2}$ by taking $\mathbf{q} = (\frac{1}{2}, \frac{1}{2}, 0)$. However, it is not left-balanced for any δ_1 because for every probability vector $(p_1, p_2) = (p, 1-p)$ it holds that $(p_1, p_2) \cdot M_{\widetilde{f}} = (p, 1-p, 1)$, which is not balanced for any p. Likewise, a function may be δ_2-right-balanced, but not left balanced.

We now prove a simple but somewhat surprising proposition, stating that if a function is (δ_1, δ_2)-balanced, then δ_1 and δ_2 must actually equal each other. Thus, any (δ_1, δ_2)-balanced function is actually δ-balanced.

Proposition 3.2 *Let $f : \{x_1, \ldots, x_m\} \times \{y_1, \ldots, y_\ell\} \to \{0, 1\}$ be a (δ_1, δ_2)-balanced function for some constants $0 \leq \delta_1, \delta_2 \leq 1$. Then, $\delta_1 = \delta_2$, and so f is δ-balanced.*

Proof: Under the assumption that f is (δ_1, δ_2)-balanced, we have that there exist probability vectors $\mathbf{p} = (p_1, \ldots, p_m)$ and $\mathbf{q} = (q_1, \ldots, q_\ell)$ such that $\mathbf{p} \cdot M_f = \delta_1 \cdot \mathbf{1}_\ell$ and $M_f \cdot \mathbf{q}^T = \delta_2 \cdot \mathbf{1}_m^T$. Observe that since $\mathbf{p}$ and $\mathbf{q}$ are probability vectors, it follows that for every constant c we have $\mathbf{p} \cdot (c \cdot \mathbf{1}_m^T) = c \cdot (\mathbf{p} \cdot \mathbf{1}_m^T) = c$; likewise $(c \cdot \mathbf{1}_\ell) \cdot \mathbf{q}^T = c$. Thus,

$$\mathbf{p} \cdot M_f \cdot \mathbf{q}^T = \mathbf{p} \cdot \left(M_f \cdot \mathbf{q}^T \right) = \mathbf{p} \cdot \left(\delta_2 \cdot \mathbf{1}_m^T \right) = \delta_2$$

and

$$\mathbf{p} \cdot M_f \cdot \mathbf{q}^T = \left(\mathbf{p} \cdot M_f \right) \cdot \mathbf{q}^T = \left(\delta_1 \cdot \mathbf{1}_\ell \right) \cdot \mathbf{q}^T = \delta_1,$$

implying that $\delta_1 = \delta_2$. ■

Note that a function can be both δ_2-right-balanced and δ_2'-right-balanced for some $\delta_2 \neq \delta_2'$. For example, consider the function M_f, which was defined above. This function is δ_2-right-balanced

for every $1/2 \leq \delta_2 \leq 1$ (by multiplying with the probability vector $(1-\delta_2, 1-\delta_2, 2\delta_2 - 1)^T$ from the right). Nevertheless, in cases where a function is δ_2-right-balanced for multiple values, Proposition 3.2 implies that the function cannot be left-balanced for *any* δ_1. Likewise, if a function is δ_1-left balanced for more than one value of δ_1, it cannot be right-balanced.

3.2 The Criterion

The criterion for determining whether or not a function implies coin-tossing is simply the question of whether the function is *strictly* δ-balanced for some δ. Formally:

Property 3.3 *A function $f : \{x_1, \ldots, x_m\} \times \{y_1, \ldots, y_\ell\} \to \{0,1\}$ is* strictly balanced *if it is δ-balanced for some $0 < \delta < 1$.*

Observe that if M_f has a monochromatic row (i.e., there exists an input x such that for all y_i, y_j it holds that $f(x, y_i) = f(x, y_j)$), then there exists a probability vector $\mathbf{p}$ such that $\mathbf{p} \cdot M_f = 0 \cdot \mathbf{1}_\ell$ or $\mathbf{p} \cdot M_f = 1 \cdot \mathbf{1}_\ell$; likewise for a monochromatic column. Nevertheless, we stress that the existence of such a row and column does not imply f is strictly balanced since it is required that δ be strictly between 0 and 1, and not equal to either.

3.3 Exploring the δ-Balanced Property

In this section we prove some technical lemmas regarding the property that we will need later in the proof. First, we show that if a function f is not left-balanced for any $0 \leq \delta \leq 1$ (resp. not right balanced), then it is *not close* to being balanced. More precisely, it seems possible that a function f can be not δ-balanced, but is only negligibly far from being balanced (i.e., there may exist some probability vector $\mathbf{p} = \mathbf{p}(n)$ (that depends on the security parameter n) such that all the values in the vector $\mathbf{p} \cdot M_f$ are at most negligibly far from δ, for some $0 \leq \delta \leq 1$). In the following claim, we show that this situation is impossible. Specifically, we show that if a function is not δ balanced, then there exists some *constant* $c > 0$, such that for any probability vector $\mathbf{p}$, there is a distance of at least c between two values in the vector $\mathbf{p} \cdot M_f$. This holds also for probability vectors that are functions of the security parameter n (as can be the case in our setting of secure protocols).

Lemma 3.4 *Let $f : \{x_1, \ldots, x_m\} \times \{y_1, \ldots, y_\ell\} \to \{0,1\}$ be a function that is not left balanced for any $0 \leq \delta_1 \leq 1$ (including $\delta_1 = 0, 1$). Then, there exists a constant $c > 0$, such that for any probability vector $\mathbf{p}$, it holds that:*

$$\max_i(\delta_1, \ldots, \delta_\ell) - \min_i(\delta_1, \ldots, \delta_\ell) \geq c$$

where $(\delta_1, \ldots, \delta_\ell) = \mathbf{p} \cdot M_f$, and M_f is the matrix representation of the function f.

Proof: Let P^m be the set of all probability vectors of size m. That is, $\mathrm{P}^m \subseteq [0,1]^m$ (which itself is a subset of $\mathbb{R}^m$), and each vector sums up to one. P^m is a closed and bounded space. Therefore using the Heine-Borel theorem, P^m is a compact space.

We start by defining a function $\phi : \mathrm{P}^m \to [0,1]$ as follows:

$$\phi(\mathbf{p}) = \max_i(\mathbf{p} \cdot M_f) - \min_i \cdot(\mathbf{p} \cdot M_f)$$

Clearly, the function $\mathbf{p} \cdot M_f$ (where M_f is fixed and $\mathbf{p}$ is the variable) is a continuous function. Moreover, the maximum (resp. minimum) of a continuous function is itself a continuous function. Therefore, from composition of continuous functions we have that the function ϕ is continuous. Using the extreme value theorem (a continuous function from a compact space to a subset of the real numbers attains its maximum and minimum), there exists some probability vector $\mathbf{p}_{\min}$ for which for all $\mathbf{p} \in \mathrm{P}^m$, $\phi(\mathbf{p}_{\min}) \leq \phi(\mathbf{p})$. Since f is not δ-balanced, $\mathbf{p}_{\min} \cdot M_f \neq \delta \cdot \mathbf{1}_\ell$ for any $0 \leq \delta \leq 1$, and so $\phi(\mathbf{p}_{\min}) > 0$. Let $c \stackrel{\text{def}}{=} \phi(\mathbf{p}_{\min})$. This implies that for any probability vector $\mathbf{p}$, we have that $\phi(\mathbf{p}) \geq \phi(\mathbf{p}_{\min}) = c$. That is:

$$\max_i(\delta_1, \ldots, \delta_\ell) - \min_i(\delta_1, \ldots, \delta_\ell) \geq c \tag{2}$$

where $(\delta_1, \ldots, \delta_\ell) = \mathbf{p} \cdot M_f$. We have proven this for *all* probability vectors of size m. Thus, it holds also for every probability vector $\mathbf{p}(n)$ that is a function of n, and for all n's (this is true since for every n, $\mathbf{p}(n)$ defines a concrete probability vector for which Eq. (2) holds). ■

A similar claim can be stated and proven for the case where f is not right balanced.

0-balance and 1-balanced functions. In our proof that a function that is not strictly-balanced does not imply the coin-tossing functionality, we deal separately with functions that are 0-balanced or 1-balanced (since they are balanced, but not strictly balanced). We now prove a property that will be needed for this case. Specifically, we show that a function f is 1-left-balanced (resp. 0-left-balanced) if and only if the matrix M_f contains the all 1 row (resp., the all 0 row). A similar argument holds for the case where the function is 1-right-balanced (or 0-right-balanced) and the all 1 column (or all 0 column).

Lemma 3.5 *Let $f : \{x_1, \ldots, x_m\} \times \{y_1, \ldots, y_\ell\} \to \{0, 1\}$ be a function. The function f is left-1-balanced if and only if the matrix M_f contains the all one row.*

Proof: We first show that if there exists a probability vector $\mathbf{p}$ for which $\mathbf{p} \cdot M_f = \mathbf{1}_\ell$, then M_f contains the all-one row. Let $\mathbf{p}$ be a probability vector for which $\mathbf{p} \cdot M_f = \mathbf{1}_\ell$. Since $\mathbf{p}$ is a probability vector (meaning that its sum is 1), we have that: $\langle \mathbf{p}, \mathbf{1}_m \rangle = 1$. Denote by C_i the ith column of the matrix M_f. Since $\mathbf{p} \cdot M_f = \mathbf{1}_\ell$, it follows that for every i it holds that $\langle \mathbf{p}, C_i \rangle = 1$. Combining this with the fact that $\langle \mathbf{p}, \mathbf{1}_m \rangle = 1$, we have that $\langle \mathbf{p}, \mathbf{1}_m - C_i \rangle = 0$.

Let $\mathbf{p} = (p_1, \ldots, p_m)$, and $C_i = (a_1, \ldots, a_m)$. We have that: $\sum_{k=1}^m p_k \cdot (1 - a_k) = 0$. As all values are positive it must be that if $p_k \neq 0$ then $a_k = 1$ (otherwise the result cannot be 0). Since this is true for any column C_i, we conclude that for every k such that $p_k \neq 0$, the kth *row* is the all one vector. Since p is a probability vector, there exists at least one k such that $p_k \neq 0$, implying that there exists an all-one row.

For the other direction, let k be the index of the all one row in the matrix M_f. Then, clearly for the probability vector e_k (the kth elementary vector, namely, all the indices are zero except for the kth index which equals 1), the product $e_k \cdot M_f = (1, \ldots, 1) = \mathbf{1}_m$, and so the matrix is left-1-balanced. ■

Using Proposition 3.2, we conclude that if a function f is 1-left-balanced but not left-balanced for any other δ_1, then it is either 1-right-balanced as well, or not right-balanced for any $0 \leq \delta_2 \leq 1$. This observation will be used in the proof.

4 Strictly-Balanced Functions Imply Coin Tossing

In this section, we show that any function f that is strictly δ-balanced can be used to fairly toss a coin. Intuitively, this follows from the well known method of Von Neumann [11] for obtaining an unbiased coin toss from a biased one. Specifically, given a coin that is heads with probability ϵ and tails with probability $1-\epsilon$, Von Neumann showed that you can toss a coin that is heads with probability exactly $1/2$ by tossing the coin twice in each phase, and stopping the first time that the pair is either heads-tails or tails-heads. Then, the parties output heads if the pair is heads-tails, and otherwise they output tails. This gives an unbiased coin because the probability of heads-tails equals the probability of tails-heads (namely, both probabilities equal $\epsilon \cdot (1-\epsilon)$). Now, since the function f is strictly δ-balanced it holds that if party P_1 chooses its input via the probability vector $(p_1, \ldots, p_m)$ then the output will equal 1 with probability δ, irrespective of what input is used by P_2; likewise if P_2 chooses its input via $(q_1, \ldots, q_\ell)$ then the output will be 1 with probability δ irrespective of what P_1 does. This yields a coin that equals 1 with probability δ and thus Von Neumann's method can be applied to securely implement the coin-tossing functionality f^{ct} defined in Section 2.2. We stress that if one of the parties aborts early and refuses to participate, then the other party proceeds by itself (essentially, tossing a coin with probability δ until it concludes). We have the following theorem:

Theorem 4.1 *Let $f : \{x_1, \ldots, x_m\} \times \{y_1, \ldots, y_\ell\} \to \{0,1\}$ be a strictly-balanced function for some constant $0 < \delta < 1$, as in Property 3.3. Then, f information-theoretically and computationally implies the coin-tossing functionality f^{ct} with malicious adversaries.*

Proof Sketch: The proof of this theorem follows from the aforementioned intuition, and is quite straightforward. We therefore just sketch the details here. First, we define a δ-biased coin tossing functionality that outputs a coin to both parties that equals 1 with probability δ and equals 0 with probability $1-\delta$. As we have described, this functionality can be securely computed using a single invocation of f, where the parties P_1 and P_2 choose their inputs via the probability vectors $\mathbf{p}$ and $\mathbf{q}$, respectively, that are guaranteed to exist by Property 3.3. The simulation of this protocol is trivial since neither party can do anything to change the probability of the output of the call to f being 1, and this probability is exactly the probability that the coin tossing functionality outputs 1. Note that if one of the parties does not participate at all (i.e., sends no input), then by the security definition of fairness the honest party still receives output and so this is the same as above. That is, if one of the parties does not participate, then the other essentially continues with the protocol by itself until either 01 or 10 is obtained.

Next, we obtain a fair unbiased coin toss by having the parties invoke the δ-biased coin tossing functionality twice and repeating until two different coins are obtained. When, this happens they output the result of the first coin in the two coin tosses. If they do not terminate within $\frac{n}{2\delta(1-\delta)}$ repetitions, then they output $\perp$ and halt. Since the probability that they terminate in any given attempt is $\delta(1-\delta) + (1-\delta)\delta = 2\delta(1-\delta)$ (because they halt after receiving either 10 or 01), it follows that they output $\perp$ with probability only

$$\Big(1 - 2\delta(1-\delta)\Big)^{\frac{n}{2\delta(1-\delta)}} < e^{-n}$$

which is negligible. The simulator works as follows. It receives the bit $b \in \{0,1\}$ from the trusted party computing the unbiased coin tossing functionality, and then runs the δ-biased coin tossing

functionality like in the protocol playing the trusted party computing f and giving the corrupted party the expected outputs, until the first pair with different coins is received (or until $\frac{n}{2\delta(1-\delta)}$ attempts were made). When this happens, the simulator hands the adversary the outputs of the δ-biased coin tossing that the corrupted party receives to be b and then $1-b$ (and thus the output is supposed to be b). Since the probability of receiving b and then $1-b$ is the same as the probability of receiving $1-b$ and then b, this view generated by the simulator is identical to that seen by the adversary in a real execution. The only difference between the real and ideal distributions is if too many attempts are made, since in the former case the honest party outputs $\perp$ whereas in the latter case it always outputs a bit. This completes the proof. ■

Application to fairness. Cleve [4] showed that there does not exist a protocol that securely computes the fair coin-tossing functionality in the plain model. Since any strictly-balanced function f implies the coin-tossing functionality, a protocol for f implies the existence of a protocol for coin-tossing. We therefore conclude:

Corollary 4.2 *Let $f : \{x_1, \ldots, x_m\} \times \{y_1, \ldots, y_\ell\} \to \{0, 1\}$ be a strictly-balanced function. Then, there does not exist a protocol that securely computes f with fairness.*

5 Unbalanced Functions Do Not Information-Theoretically Imply Coin Tossing

We now show that any function f that is *not* strictly-balanced (for all δ) does *not* information-theoretically imply the coin-tossing functionality. Stated differently, there does not exist a protocol for fairly tossing an unbiased coin in the f-hybrid model, with statistical security. Observe that in the f-hybrid model, it is possible for the parties to simultaneously exchange information, in some sense, since both parties receive output from calls to f at the same time. Thus, Cleve-type arguments [4] that are based on the fact that one party must know more than the other party at some point do not hold. We prove our result by showing that for every protocol there exists an unbounded malicious adversary that can bias the result. Our unbounded adversary needs to compute probabilities, which can actually be approximated given an $\mathcal{NP}$-oracle. Thus, it is possible to interpret our technical result also as a black-box separation, if desired.

As we have mentioned in the introduction, although we prove an "impossibility result" here, the implication is the opposite. Specifically, our proof that an unbalanced[3] f cannot be used to toss an unbiased coin implies that it may be possible to securely compute such functions with fairness. Indeed, the functions that were shown to be securely computable with fairness in [8] are unbalanced.

Recall that a function is not strictly balanced if is not δ-balanced for any $0 < \delta < 1$. We treat the case that the function is not δ-balanced at all separately from the case that it *is* δ-balanced but for $\delta = 0$ or $\delta = 1$. In the proof we show that in both of these cases, such a function cannot be used to construct a fair coin tossing protocol.

In the f-hybrid model, the parties may invoke the function f in two different "directions". Specifically, in some of the invocations P_1 selects the first input of f (i.e., x) and P_2 selects the

[3]Note, that the name unbalanced is a bit misleading as the complement of not being strictly balanced also includes being 1 or 0-balanced.

second input of f (i.e., y), while in other invocation the roles may be reversed (with P_1 selecting the y input and P_2 selecting the x input). The question of which party plays which role is determined by the protocol. For convenience, we assume that P_1 always selects the first input, and P_2 always selects the second input, but the parties have access to two ideal functions f and f^T. Thus, calls to f in the protocol where P_2 selects the second input and P_1 selects the first input are modeled by a call to f^T where P_1 selects the first input and P_2 selects the second input.

Theorem 5.1 *Let $f : \{x_1, \ldots, x_m\} \times \{y_1, \ldots, y_\ell\} \to \{0, 1\}$ be a function that is* not *left-balanced, for any $0 \leq \delta_1 \leq 1$. Then, f does not information-theoretically imply the coin-tossing functionality with security in the presence of malicious adversaries.*

Before proceeding to the proof, we provide high level intuition. We begin by observing that if f does not contain an embedded OR (i.e., inputs x_0, x_1, y_0, y_1 such that $f(x_0, y_0) = f(x_1, y_0) = f(x_0, y_1) \neq f(x_1, y_1)$) or an embedded XOR (i.e., inputs x_0, x_1, y_0, y_1 such that $f(x_0, y_0) = f(x_1, y_1) \neq f(x_0, y_1) = f(x_1, y_0)$), then it is trivial and can be computed by simply having one party send the output to the other. This is because such a function depends only on the input of one party. Thus, by [4], it is impossible to fairly toss an unbiased coin in the f-hybrid model, since this is the same as fairly tossing an unbiased coin in the plain model. Thus, we consider only functions f that have an embedded OR or an embedded XOR.

In addition, we can consider coin-tossing protocols that consist of calls to f and f^T only, and no other messages. This is due to the fact that we can assume that any protocol consists of rounds, where each round is either an invocation of f (or f^T) or a message consisting of a single bit being sent from one party to the other. Since f has an embedded OR or an embedded XOR, messages of a single bit can be sent by invoking f. This is because in both cases there exist inputs x_0, x_1, y_0, y_1 such that $f(x_1, y_0) \neq f(x_1, y_1)$ and $f(x_0, y_1) \neq f(x_1, y_1)$. Thus, in order for P_2 to send P_1 a bit, the protocol can instruct the parties to invoke f where P_1 always inputs x_1, and P_2 inputs y_0 or y_1 depending on the bit that it wishes to send; likewise for P_1. Thus, any non-trivial function f enables "bit transmission" in the above sense. Observe that if one of the parties is malicious and uses an incorrect input, then this simply corresponds to sending an incorrect bit in the original protocol. Therefore, we can assume that a protocol consists of rounds, where in each round there is an invocation of f (where P_1 selects the x input and P_2 selects the y input) followed by an invocation of f^T (where here P_1 selects the y input and P_2 selects the x input).

Intuition. The fact that f is not balanced implies that in any single invocation of f, one party is able to have some effect on the output by choosing its input appropriately. That is, if the function is non-balanced on the left then the party on the right can use an input not according to the prescribed distribution in the protocol, and this will change the probability of the output of the invocation being 1 (for example). However, it may be possible that the ability to somewhat influence the output in individual invocations is not sufficient to bias the overall computation, due to the way that the function calls are composed. Thus, in the proof we need to show that an adversary is in fact capable of biasing the overall protocol. We demonstrate this by showing that there exist crucial invocations where the ability to bias the outcome in these invocation suffice for biasing the overall outcome. Then, we show that such invocations are always reached in any execution of the protocol, and that the adversary can (inefficiently) detect when such an invocation has been reached and can (inefficiently) compute which input it needs to use in that invocation in order to bias the output.

We prove the above by considering the execution tree of the protocol, which is comprised of calls to f and the flow of the computation based on the output of f in each invocation (i.e., the parties proceed left in the tree if the output of f in this invocation is 0; and right otherwise). Observe that a path from the root of the tree to a leaf-node represents a protocol execution. We show that in *every* path from the root to a leaf, there exists at least one node with the property that influencing the output of the single invocation of that node yields a bias in the final outcome. In addition, we describe the strategy of the adversary to detect such a node and choose its input for that node in order to obtain a bias.

In more detail, for every node v in the execution tree of the protocol, the adversary calculates (in an inefficient manner) the probability that the output of the computation equals 1, assuming that v is reached in the execution. Observe that the probability of obtaining 1 at the root node is at most negligibly far from 1/2 (since it is a secure coin-tossing protocol), and that the probability of obtaining 1 at a leaf node is either 1 or 0, depending on whether the output at the given leaf is 1 or 0 (the way that we define the tree is such that the output is fully determined by the leaf). Using a pigeon-hole like argument, we show that on every path from the root to a leaf there must be at least one node where the probability of outputting a 1 given that this node is reached is significantly different than the probability of outputting 1 given that the node's child on the path is reached. We further show that this difference implies that the two children of the given node yield significantly different probabilities of outputting 1 (since the probability of outputting 1 at a node v is the weighted-average of outputting 1 at the children, based on the probability of reaching each child according to the protocol). This implies that in every protocol execution, there exists an invocation of f where the probability of outputting 1 in the entire protocol is significantly different if the output of this invocation of f is 0 or 1. Since f is not balanced, it follows that for any distribution used by the honest party to choose its input for this invocation, there exist two inputs that the corrupted party can use that result in significantly different probabilities of obtaining 1. In particular, at least one of these probabilities is *significantly different from the probability of obtaining 1 in this call when both parties are honest and follow the protocol.*[4] Thus, the adversary can cause the output of the entire execution to equal 1 with probability significantly different to 1/2, which is the probability when both parties play honestly.

The above description does not deal with question of whether the output will be biased towards 0 or 1. In fact we design two adversaries, one that tries to bias the output towards 0 and the other towards 1. Then we show that at least one of these adversaries will be successful (see Footnote 4 for an explanation as to why only one of the adversaries may be successful). The two adversaries are similar and very simple. They search for the node on the path of the execution where the bias can be created and there make their move. In all nodes until and after that node they behave honestly (i.e., choose inputs for the invocations of f according to the input distribution specified by the protocol). We analyze the success of the adversaries and show that at least one of them biases the output with noticeable probability.

Proof of Theorem 5.1: Assume that π is a coin tossing protocol that implements the coin-tossing functionality in the f-hybrid model. That is, when both parties are honest, they output the

[4]Observe that one of these probabilities may be the same as the probability of obtaining 1 in an honest execution, in which case choosing that input will not result in any bias. Thus, the adversary may be able to bias the output of the entire protocol towards 1 or may be able to bias the output of the entire protocol towards 0, but not necessarily both.

same uniformly distributed bit, except with negligible probability. We construct an exponential-time malicious adversary who biases the outcome, in contradiction to the assumption that π is a fair coin-tossing protocol.

As described above we assume that π consists only of invocations of f and f^T. Let $r(n)$ be the number of rounds in π and let $c(n)$ be an upper bound on the number of random coins used by each party in the protocol. We assume that the function is not left-balanced and therefore construct the adversary $\mathcal{A}$ that controls P_2.

Simplifying assumptions. We first prove the theorem under three simplifying assumptions regarding the protocol π; we show how to remove these assumptions later. The first assumption is that the protocol π only makes calls to f and not to f^T (i.e., in all calls to f, party P_1 sends x and party P_2 sends y and the output is $f(x,y)$). The second assumption is that honest parties always agree on the same output bit (i.e., the probability of them outputting different coins or aborting is zero). The third assumption is that the output of the parties is a deterministic function of the public transcript of the protocol alone.

The transcript tree and execution probabilities. We define a binary tree $\mathcal{T}$ of depth $r(n)$ that represents the transcript of the execution. Each node in the tree indicates an invocation of f, and the left and right edges from a node indicate invocations of f where the output was 0 and 1, respectively.

We define the information that is saved in the internal nodes and leaves. Given a pair of random tapes (r_1, r_2) used by P_1 and P_2, respectively, it is possible to generate a full transcript of an honest execution of the protocol that contains the inputs that are sent to f in each round by both parties, the output of f in that round and the output of the parties at the end of the execution. Thus, we write on each (internal) node $v \in \mathcal{T}$ all the pairs of random tapes that reach this node, and the inputs that are sent to the trusted party for the associated call to f for those given tapes.

The leaves of the tree contain the output of the execution. Note that by the second and third simplifying assumption on protocol π described above, each leaf determines a single output (for both parties and for all random tapes reaching the leaf).

Given the distribution of the transcripts on the nodes of the tree we define two values for each node $v \in \mathcal{T}$. The first value which we denote by p_v is the probability that the parties output 1 at the end of the computation conditioned on the execution reaching the node v. This is simply computed as the ratio between the number of random tapes that reach the node v and result in an output 1 and all random tapes that reach v. The second value, denoted by γ^v, is the probability that the output of the invocation of f in this node is 1 (when both parties play honestly). This is calculated by computing the probabilities over the inputs of the parties in the invocation, as specified by the protocol and based on the random tapes in the node v. Denote the distributions over the inputs of P_1 and P_2 in the leaf by $(\alpha_1^v, \ldots, \alpha_m^v)$ and $(\beta_1^v, \ldots, \beta_\ell^v)$, respectively (i.e., in leaf v, P_1 chooses x_i with probability α_i^v, and P_2 chooses y_j with probability β_j^v). It holds that the probability that the parties receive 1 as output in the invocation of f at node v is $\gamma^v = (\alpha_1^v, \ldots, \alpha_m^v) \cdot M_f \cdot (\beta_1^v, \ldots, \beta_\ell^v)^T$.

The adversaries. Let $\epsilon = \epsilon(n) = \frac{1/2 - \mu(n)}{r(n)}$, where $\mu(\cdot)$ is from Eq. (1); i.e., the probability that P_1 and P_2 output 1 (or 0) is $1/2 \pm \mu(n)$ (this means that p_v where v is the root node of the tree $\mathcal{T}$ equals $1/2 \pm \mu(n)$). Denote the two children of $v \in \mathcal{T}$ by $left(v)$ and $right(v)$ and they are reached by output 0 and 1 in the call to f resp. We construct two adversaries controlling P_2: one

which attempts to bias the outcome towards 1 and the other which attempts to bias the outcome towards 0. The adversaries search for a node $v \in \mathcal{T}$ for which the difference in the probability of outputting 1 at v and the probability of outputting 1 at one of its children is at least ϵ. The adversaries behave honestly until they reach such a node, at which point they have the opportunity to bias the outcome by deviating from the protocol in the invocation at this node. In the remaining nodes they again act honestly.

Adversary $\mathcal{A}^1$ controlling P_2 (biasing towards 1):
Let v be the current node in the tree $\mathcal{T}$ (starting at the root) and $\boldsymbol{\alpha}^v = (\alpha_1^v, \ldots, \alpha_n^v)$ be the input distribution for P_1 (as explained above, this can be computed). Compute $(\delta_1^v, \ldots, \delta_\ell^v) = \boldsymbol{\alpha}^v \cdot M_f$. Let i and k be indices such that $\delta_i^v = \max_{1 \le j \le \ell}\{\delta_j^v\}$ and $\delta_k^v = \min_{1 \le j \le \ell}\{\delta_j^v\}$. In the *first* node v in the execution for which $|p_{right(v)} - p_v| \ge \epsilon$ or $|p_{left(v)} - p_v| \ge \epsilon$, act according to the following:

1. If $p_{right(v)} \ge p_v + \epsilon$ or $p_{left(v)} \le p_v - \epsilon$ then send input y_i in this invocation of f (this increases the probability of reaching node $right(v)$ because the probability of obtaining 1 in this invocation is δ_i^v which is *maximal*).
2. If $p_{left(v)} \ge p_v + \epsilon$ or $p_{right(v)} \le p_v - \epsilon$ then send input y_k in this invocation of f (this increases the probability of reaching node $left(v)$ because the probability of obtaining 1 in this invocation is δ_k^v which is *minimal*).

In all other nodes act honestly.

The adversary $\mathcal{A}^0$ controlling P_2 (biasing towards 0): The adversary is the same as $\mathcal{A}^1$ with the following differences:

- Step 1: If $p_{right(v)} \ge p_v + \epsilon$ or $p_{left(v)} \le p_v - \epsilon$: then send input y_k in this invocation of f (where δ_k^v is minimal).
- Step 2: If $p_{left(v)} \ge p_v + \epsilon$ or $p_{right(v)} \le p_v - \epsilon$: then send input y_i in this invocation of f (where δ_i^v is maximal).

First, we show that in any honest execution there exists a node v for which $|p_v - p_{right(v)}| \ge \epsilon$ or $|p_v - p_{left(v)}| \ge \epsilon$. Then, we show that once the execution reaches such a node v, one of the adversaries succeeds in biasing the outcome.

The set $\mathcal{V}$. Let $\mathcal{V}$ be the set of all nodes v in $\mathcal{T}$ that have two children, for which $|p_v - p_{right(v)}| \ge \epsilon$ or $|p_v - p_{left(v)}| \ge \epsilon$, and v is the first node in the path between the root and v that satisfies the condition. In order to see that in any execution, the adversaries reachs such a node, we show that for every pair of random coins $(r_1, r_2) \in \mathsf{Uni}$ there exists a node $v \in \mathcal{V}$ such that an honest execution with (r_1, r_2) reaches v. Fix (r_1, r_2). Note that any pair of random coins define a unique path $u_1, \ldots, u_{r(n)}$ from the root u_1 to a leaf $u_{r(n)}$.

We start with the case where $u_{r(n)}$ defines output 1. From Eq. (1), we have that $p_{u_1} \le 1/2 + \mu(n)$. In addition, $p_{u_{r(n)}} = 1$ since the output is 1. Therefore, we have that:

$$\frac{\sum_{i=1}^{r(n)-1} \left(p_{u_{i+1}} - p_{u_i}\right)}{r(n)} = \frac{p_{u_{r(n)}} - p_{u_1}}{r(n)} \ge \frac{1 - \frac{1}{2} - \mu(n)}{r(n)} = \frac{\frac{1}{2} - \mu(n)}{r(n)} = \epsilon.$$

Thus, the average of differences is greater than or equal to ϵ, which means that there exists an i such that $p_{u_{i+1}} - p_{u_i} \geq \epsilon$. Moreover, this u_i must have two children since otherwise $p_{u_i} = p_{u_{i+1}}$. Since $u_{i+1} \in \{right(u_i), left(u_i)\}$, we conclude that there exists a node u_i such that either $p_{right(u_i)} \geq p_{u_i} + \epsilon$ or $p_{left(u_i)} \geq p_{u_i} + \epsilon$.

The second case is where $u_{r(n)}$ defines output 0, and $p_{u_1} \geq 1/2 - \mu(n)$. Similarly to the above, we have that:

$$\frac{\sum_{i=1}^{r(n)-1} \left(p_{u_{i+1}} - p_{u_i}\right)}{r(n)} = \frac{p_{u_{r(n)}} - p_{u_1}}{r(n)} \leq \frac{0 - \frac{1}{2} + \mu(n)}{r(n)} = -\epsilon,$$

which implies that there exists an i such that $p_{u_{i+1}} - p_{u_i} \leq -\epsilon$. Moreover, this u_i must have two children, otherwise we must have that $p_{u_i} = p_{u_{i+1}}$. Since $u_{i+1} \in \{right(u_i), left(u_i)\}$, we conclude that there exists a node u_i such that either $p_{right(u_i)} \leq p_{u_i} - \epsilon$ or $p_{left(u_i)} \leq p_{u_i} - \epsilon$.

We conclude that in every execution of the protocol, the parties reach a node $v \in \mathcal{V}$.

We know that in every node $v \in \mathcal{V}$, the difference between $p_{left(v)}$ or $p_{right(v)}$ and p_v is large. We now show that this implies that the difference between $p_{right(v)}$ and $p_{left(v)}$ themselves is large. This is the basis for the ability of the adversary to bias the output. Let $v \in \mathcal{V}$ be a fixed node, let $\boldsymbol{\alpha}^v$ be the distribution over the input of the honest P_1 to f at node v, and let $\boldsymbol{\beta}^v$ be the distribution over the input of the honest P_2 to f at node v (as described above, these distributions over the inputs are fully defined and inefficiently computable). Let $\gamma^v = \boldsymbol{\alpha}^v \cdot M_f \cdot \boldsymbol{\beta}^v$ be the probability that the honest parties obtain output 1 in this node (when sending inputs according to the distributions). By the definition of p_v and γ^v, we have that:

$$p_v = \gamma^v \cdot p_{right(v)} + (1 - \gamma^v) \cdot p_{left(v)}.$$

This is because with probability γ^v the honest parties proceed to $right(v)$ and with probability $1 - \gamma^v$ they proceed to $left(v)$. Thus, the probability of outputting 1 at v is as above. We now prove:

Claim 5.2 *For every node $v \in \mathcal{V}$:*

- $0 < \gamma^v < 1$.
- *Only one of the following holds: $|p_v - p_{right(v)}| \geq \epsilon$ or $|p_v - p_{left(v)}| \geq \epsilon$ but not both.*
- *If $p_{right(v)} \geq p_v + \epsilon$, then $p_{right(v)} \geq p_{left(v)} + \epsilon$.*
- *If $p_{left(v)} \geq p_v + \epsilon$, then $p_{left(v)} \geq p_{right(v)} + \epsilon$.*
- *If $p_{right(v)} \leq p_v - \epsilon$, then $p_{right(v)} \leq p_{left(v)} - \epsilon$.*
- *If $p_{left(v)} \leq p_v - \epsilon$, then $p_{left(v)} \leq p_{right(v)} - \epsilon$.*

Proof: Recall that the probability γ^v can be computed as the number of random tapes that reach the right child of v over the number of random tapes that are consistent with v. Therefore, if $\gamma^v = 0$, it means that no execution reaches the right child of v. Similarly, in case $\gamma^v = 1$, we have that there is no left child of v. In any case, both cases are in contradiction the the fact that $v \in \mathcal{V}$, which means above the other properties, that the node v has two children.

For the second item, assume that both $|p_v - p_{right(v)}| \geq \epsilon$ and $|p_v - p_{left(v)}| \geq \epsilon$ hold simultaneously. This means that:

$$\begin{aligned} -\left(p_v - p_{right(v)}\right) &\leq \epsilon \leq p_v - p_{right(v)} \\ -\left(p_v - p_{left(v)}\right) &\leq \epsilon \leq p_v - p_{left(v)} \end{aligned}$$

When we multiply the first equation by γ^v and the second by $1-\gamma^v$ (both are not zero, according to the first item), we obtain that:

$$\begin{aligned} -\gamma^v \cdot \left(p_v - p_{right(v)}\right) &\leq \quad \gamma^v \cdot \epsilon \quad \leq \gamma^v \cdot \left(p_v - p_{right(v)}\right) \\ -(1-\gamma^v) \cdot \left(p_v - p_{left(v)}\right) &\leq (1-\gamma^v) \cdot \epsilon \leq (1-\gamma^v) \cdot \left(p_v - p_{left(v)}\right) \end{aligned}$$

Summing the above two equations together, and recalling that $p_v = \gamma^v \cdot p_{right(v)} + (1-\gamma^v) \cdot p_{left(v)}$, we have that:

$$0 = -p_v + p_v \leq \epsilon \leq p_v - p_v = 0$$

and so $\epsilon = 0$, in contradiction to the fact that: $\epsilon = \frac{1/2 - \mu(n)}{r(n)} > 0$.

For the third item, assume that $p_{right(v)} \geq p_v + \epsilon$. We have:

$$\begin{aligned} p_{right(v)} &\geq p_v + \epsilon \\ p_{right(v)} &\geq \gamma^v \cdot p_{right(v)} + (1-\gamma^v) \cdot p_{left(v)} + \epsilon \\ (1-\gamma^v) \cdot p_{right(v)} &\geq (1-\gamma^v) \cdot p_{left(v)} + \epsilon \\ p_{right(v)} &\geq p_{left(v)} + \frac{\epsilon}{1-\gamma^v} \geq p_{left(v)} + \epsilon \end{aligned}$$

where the last inequality holds since $0 < 1 - \gamma^v < 1$.

The other items are derived through similar computations. ■

Analyzing the adversary's success. We are now ready to show that at least one of the adversaries $\mathcal{A}^1$ or $\mathcal{A}^0$ succeeds in biasing the output.

We start with some notation. For a given node v and a bit $b \in \{0,1\}$, denote by $\mathsf{out}^b_{|v}(P_1, P_2^*)$ the probability that an honest party P_1 outputs the bit b in an execution with P_2^*, conditioned on the event that the execution reached node v. Note that in an execution between two honest parties it holds that :

$$\mathsf{out}^1_{|v}(P_1,P_2) = \Pr_{(r_1,r_2)\leftarrow \mathsf{Uni}}\left[\mathsf{out}\langle P_1(r_1), P_2(r_2)\rangle = 1 \;\middle|\; v \text{ is reached}\right] = p_v = \gamma^v \cdot p_{right(v)} + (1-\gamma^v) \cdot p_{left(v)}.$$

Similarly, $\mathsf{out}^0_{|v}(P_1,P_2) = 1 - p_v$. Note also that when v is the root node, $\mathsf{out}^1_{|v}(P_1,P_2) = \frac{1}{2} \pm \mu(n)$.

Let $v \in \mathcal{V}$. We consider two executions: in both executions P_1 is honest, while in the first execution P_2 is controlled by $\mathcal{A}^1$, and in the second execution P_2 is controlled by $\mathcal{A}^0$. We compute the probabilities $\mathsf{out}^1_{|v}(P_1,\mathcal{A}^1)$ and $\mathsf{out}^0_{|v}(P_1,\mathcal{A}^0)$ and compare these to the output when both parties are honest. Specifically, we show that for every $v \in \mathcal{V}$ there is a difference between the output probability at v when honest parties run, and when an honest P_1 runs with a dishonest P_2 (we actually consider the sum of the differences, and will later use this to show that at least one of $\mathcal{A}^0$ and $\mathcal{A}^1$ must succeed in biasing).

Claim 5.3 *There exists a constant $c > 0$ such that for every node $v \in \mathcal{V}$ it holds that:*

$$\Big(\mathsf{out}^0_{|v}(P_1, \mathcal{A}^0) - \mathsf{out}^0_{|v}(P_1, P_2)\Big) + \Big(\mathsf{out}^1_{|v}(P_1, \mathcal{A}^1) - \mathsf{out}^1_{|v}(P_1, P_2)\Big) \geq c \cdot \epsilon$$

Proof: Let $v \in \mathcal{V}$ and denote $\delta^v_{\max} = \max_{1 \leq i \leq \ell}\{\delta^v_i\}$ and $\delta^v_{\min} = \min_{1 \leq j \leq \ell}\{\delta^v_j\}$, where $\delta^v_1, \ldots, \delta^v_\ell$ are as computed by the adversaries. Since $v \in \mathcal{V}$, either $p_{right(v)} \geq p_v + \epsilon$, $p_{left(v)} \geq p_v + \epsilon$, $p_{right}(v) \leq p_v - \epsilon$ or $p_{left(v)} \leq p_v - \epsilon$. We consider two cases:

Case 1 – $\boldsymbol{p_{right(v)} \geq p_v + \epsilon}$ or $\boldsymbol{p_{left(v)} \leq p_v - \epsilon}$: In these cases, the adversary $\mathcal{A}^1$ sends y_i for which $\delta_i = \delta^v_{\max}$, while the adversary $\mathcal{A}^0$ sends y_k for which $\delta_k = \delta^v_{\min}$.

We now compute the difference between the probabilities that P_1 outputs 1 when it interacts with and honest P_2, and with the adversary $\mathcal{A}^1$, where in both executions we assume that v is reached.[5]

$$\begin{aligned}
&\mathsf{out}^1_{|v}(P_1, \mathcal{A}^1) - \mathsf{out}^1_{|v}(P_1, P_2) \\
&\quad= \Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}\Big[\mathsf{out}\langle P_1(r_1), \mathcal{A}^1(r_2)\rangle = 1 \;\Big|\; v\Big] - \Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}\Big[\mathsf{out}\langle P_1(r_1), P_2(r_2)\rangle = 1 \;\Big|\; v\Big] \\
&\quad= \delta^v_{\max} \cdot p_{right(v)} + (1 - \delta^v_{\max}) \cdot p_{left(v)} - \big(\gamma^v \cdot p_{right(v)} + (1 - \gamma^v) \cdot p_{left(v)}\big) \\
&\quad= (\delta^v_{\max} - \gamma^v) \cdot p_{right(v)} + (\gamma^v - \delta^v_{\max}) \cdot p_{left(v)} \\
&\quad= (\delta^v_{\max} - \gamma^v) \cdot \big(p_{right(v)} - p_{left(v)}\big) \\
&\quad\geq (\delta^v_{\max} - \gamma^v) \cdot \epsilon \;.
\end{aligned}$$

where the last inequality follows from Claim 5.2 which states that $p_{right(v)} - p_{left(v)} \geq \epsilon$ in this case. Observe that when $\delta^v_{\max} = \gamma^v$, then there is no bias (i.e., the probability of obtaining output 1 may be the same as when P_2 is honest).

In a similar way, we compute the difference between the probabilities that P_1 outputs 0 when it interacts with and honest P_2, and with the adversary $\mathcal{A}^0$, conditioned on the event that the node $v \in \mathcal{V}$ is reached. We have

$$\begin{aligned}
&\mathsf{out}^0_{|v}(P_1, \mathcal{A}^0) - \mathsf{out}^0_{|v}(P_1, P_2) \\
&= \Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}\Big[\mathsf{out}\langle P_1(r_1), \mathcal{A}^0(r_2)\rangle = 0 \;\Big|\; v\Big] - \Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}\Big[\mathsf{out}\langle P_1(r_1), P_2(r_2)\rangle = 0 \;\Big|\; v\Big] \\
&= \left(1 - \Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}\Big[\mathsf{out}\langle P_1(r_1), \mathcal{A}^0(r_2)\rangle = 1 \;\Big|\; v\Big]\right) - \left(1 - \Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}\Big[\mathsf{out}\langle P_1(r_1), P_2(r_2)\rangle = 1 \;\Big|\; v\Big]\right) \\
&= \Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}\Big[\mathsf{out}\langle P_1(r_1), P_2(r_2)\rangle = 1 \;\Big|\; v\Big] - \Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}\Big[\mathsf{out}\langle P_1(r_1), \mathcal{A}^0(r_2)\rangle = 1 \;\Big|\; v\Big] \\
&= \big(\gamma^v \cdot p_{right(v)} + (1 - \gamma^v) \cdot p_{left(v)}\big) - \big(\delta^v_{\min} \cdot p_{right(v)} + (1 - \delta^v_{\min}) \cdot p_{left(v)}\big) \\
&= (\gamma^v - \delta^v_{\min}) \cdot \big(p_{right(v)} - p_{left(v)}\big) \\
&\geq (\gamma^v - \delta^v_{\min}) \cdot \epsilon \;.
\end{aligned}$$

where the last equality it true from Claim 5.2. Once again, when $\delta^v_{\min} = \gamma^v$, then there is no bias.

[5]Note that when we write $\mathsf{out}\langle P_1(r_1), \mathcal{A}^1(r_2)\rangle$ we mean the output of P_1 when interacting with $\mathcal{A}^1$ with respective random tapes r_1 and r_2. Since $\mathcal{A}^1$ behaves according to the honest strategy except in the node $v \in \mathcal{V}$, we have that $\mathcal{A}^1$ uses the coins r_2 just like an honest P_2 except in node v. Moreover, when conditioning on the event that the node v is reached, we just write "v" for short, instead of "v is reached".

The crucial observation is now that since f is *not* δ balanced, it holds that $\delta^v_{\min} \neq \delta^v_{\max}$ and thus either $\delta^v_{\max} \neq \gamma^v$ or $\delta^v_{\min} \neq \gamma^v$ or both. We can compute the overall bias of the adversaries as:

$$\begin{aligned}\Big(\mathsf{out}^0_{|v}(P_1,\mathcal{A}^0) - \mathsf{out}^0_{|v}(P_1,P_2)\Big) + \Big(\mathsf{out}^1_{|v}(P_1,\mathcal{A}^1) - \mathsf{out}^1_{|v}(P_1,P_2)\Big)\\ \geq \quad (\delta^v_{\max} - \gamma^v)\cdot\epsilon + (\gamma^v - \delta^v_{\min})\cdot\epsilon = (\delta^v_{\max} - \delta^v_{\min})\cdot\epsilon\end{aligned}$$

Now, by the fact that f is *not* left-balanced, Claim 3.4 states that there exists a constant $c > 0$ such that for all probability vectors $\mathbf{p}$: $\max_i(\mathbf{p}\cdot M_f) - \min_i(\mathbf{p}\cdot M_f) \geq c$. In particular, this means that $\delta^v_{\max} - \delta^v_{\min} \geq c$. Therefore, we can conclude that:

$$\Big(\mathsf{out}^0_{|v}(P_1,\mathcal{A}^0) - \mathsf{out}^0_{|v}(P_1,P_2)\Big) + \Big(\mathsf{out}^1_{|v}(P_1,\mathcal{A}^1) - \mathsf{out}^1_{|v}(P_1,P_2)\Big) \geq (\delta^v_{\max} - \delta^v_{\min})\cdot\epsilon \geq c\cdot\epsilon$$

Case 2 – $p_{left(v)} \geq p_v + \epsilon$ or $p_{right(v)} \leq p_v - \epsilon$: In this case, $\mathcal{A}^1$ sends y_k for which $\delta_k = \delta^v_{\min}$, and $\mathcal{A}^0$ sends y_i for which $\delta_i = \delta^v_{\max}$. Using a similar calculation to the previous case, and using Claim 5.2 which states that $p_{left(v)} - p_{right(v)} \geq \epsilon$ in this case, we have:

$$\begin{aligned}&\mathsf{out}^1_{|v}(P_1,\mathcal{A}^1) - \mathsf{out}^1_{|v}(P_1,P_2)\\ &= \Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}\Big[\mathsf{out}\langle P_1(r_1),\mathcal{A}^1(r_2)\rangle = 1 \;\Big|\; v\Big] - \Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}\Big[\mathsf{out}\langle P_1(r_1),P_2(r_2)\rangle = 1 \;\Big|\; v\Big]\\ &= \delta^v_{\min}\cdot p_{right(v)} + (1-\delta^v_{\min})\cdot p_{left(v)} - \gamma^v\cdot p_{right(v)} - (1-\gamma^v)\cdot p_{left(v)}\\ &= (\gamma^v - \delta^v_{\min})\cdot\big(p_{left(v)} - p_{right(v)}\big)\\ &\geq (\gamma^v - \delta^v_{\min})\cdot\epsilon\end{aligned}$$

and

$$\begin{aligned}&\mathsf{out}^0_{|v}(P_1,\mathcal{A}^0) - \mathsf{out}^0_{|v}(P_1,P_2)\\ &= \Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}\Big[\mathsf{out}\langle P_1(r_1),\mathcal{A}^0\rangle = 1 \;\Big|\; v\Big] - \Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}\Big[\mathsf{out}\langle P_1(r_1),P_2(r_2)\rangle = 1 \;\Big|\; v\Big]\\ &= \gamma^v\cdot p_{right(v)} + (1-\gamma^v)\cdot p_{left(v)} - \delta^v_{\max}\cdot p_{right(v)} - (1-\delta^v_{\max})\cdot p_{left(v)}\\ &= (\delta^v_{\max} - \gamma^v)\cdot\big(p_{left(v)} - p_{right(v)}\big)\\ &\geq (\delta^v_{\max} - \gamma^v)\cdot\epsilon\ .\end{aligned}$$

Therefore, the overall bias is:

$$\begin{aligned}\Big(\mathsf{out}^0_{|v}(P_1,\mathcal{A}^0) - \mathsf{out}^0_{|v}(P_1,P_2)\Big) + \Big(\mathsf{out}^1_{|v}(P_1,\mathcal{A}^1) - \mathsf{out}^1_{|v}(P_1,P_2)\Big)\\ \geq \quad (\delta^v_{\max} - \gamma^v)\cdot\epsilon + (\gamma^v - \delta^v_{\min})\cdot\epsilon = (\delta^v_{\max} - \delta^v_{\min})\cdot\epsilon\end{aligned}$$

Again, by the fact that f is *not* left-balanced, Claim 3.4 implies that for the same constant $c > 0$ as above it holds:

$$\Big(\mathsf{out}^0_{|v}(P_1,\mathcal{A}^0) - \mathsf{out}^0_{|v}(P_1,P_2)\Big) + \Big(\mathsf{out}^1_{|v}(P_1,\mathcal{A}^1) - \mathsf{out}^1_{|v}(P_1,P_2)\Big) \geq (\delta^v_{\max} - \delta^v_{\min})\cdot\epsilon \geq c\cdot\epsilon.$$

This completes the proof of Claim 5.3. ■

Computing the overall bias. Since in any execution exactly one node $v \in \mathcal{V}$ is reached, and since in any execution a node in $\mathcal{V}$ is reached, we have that for any party P_2^* (in particular, an honest P_2, the adversary $\mathcal{A}^0$ or the adversary $\mathcal{A}^1$), and for every bit $b \in \{0,1\}$:

$$\Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}[\mathsf{out}\langle P_1(r_1), P_2^*(r_2)\rangle = b] = \sum_{v\in\mathcal{V}} \Pr\left[v \text{ is reached in } \langle P_1(r_1), P_2^*(r_2)\rangle\right] \cdot \mathsf{out}^b_{|v}(P_1(r_1), P_2^*(r_2))$$

We denote the probability above by $\mathsf{out}^b(P_1, P_2^*)$.

Note that since both adversaries act honestly until they reach a node $v \in \mathcal{V}$, the executions $\langle P_1, P_2^*\rangle$ (where $P_2^* \in \{P_2, \mathcal{A}^1, \mathcal{A}^0\}$) are the same up to the point when they reach the node v, and thus:

$$\begin{aligned}\Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}[v \text{ is reached in } \langle P_1(r_1), P_2(r_2)\rangle] &= \Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}\left[v \text{ is reached in } \langle P_1(r_1), \mathcal{A}^1(r_2)\rangle\right] \\ &= \Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}\left[v \text{ is reached in } \langle P_1(r_1), \mathcal{A}^0(r_2)\rangle\right]\end{aligned}$$

we therefore denote this equal value as reach_v. Moreover, since any execution reaches exactly one node $v \in \mathcal{V}$, we have: $\sum_{v\in\mathcal{V}} \mathsf{reach}_v = 1$.

Combining the above, we conclude that:

$$\begin{aligned}&\left(\mathsf{out}^1(P_1,\mathcal{A}^1) - \mathsf{out}^1(P_1,P_2)\right) + \left(\mathsf{out}^0(P_1,\mathcal{A}^1) - \mathsf{out}^0(P_1,P_2)\right) \\ &= \sum_{v\in\mathcal{V}} \mathsf{reach}_v \cdot \left[\left(\mathsf{out}^1_{|v}(P_1,\mathcal{A}^1) - \mathsf{out}^1_{|v}(P_1,P_2)\right) + \left(\mathsf{out}^0_{|v}(P_1,\mathcal{A}^0) - \mathsf{out}^0_{|v}(P_1,P_2)\right)\right] \\ &\geq \sum_{v\in\mathcal{V}} \mathsf{reach}_v \cdot (c\cdot\epsilon) = c\cdot\epsilon\end{aligned}$$

where the last inequality is true by Claim 5.3. Since π is a coin-tossing protocol, by Eq. (1) we have that:

$$\left|\mathsf{out}^0(P_1,P_2) - \frac{1}{2}\right| \leq \mu(n) \quad \text{and} \quad \left|\mathsf{out}^1(P_1,P_2) - \frac{1}{2}\right| \leq \mu(n)\ .$$

Thus, we have that:

$$\left(\mathsf{out}^1(P_1,\mathcal{A}^1) - \frac{1}{2}\right) + \left(\mathsf{out}^0(P_1,\mathcal{A}^0) - \frac{1}{2}\right) \geq c\cdot\epsilon - 2\mu(n).$$

Since c is a constant, and $\epsilon = \epsilon(n) = \frac{1/2-\mu(n)}{r(n)}$, we have that the sum of the biases is non-negligible, and therefore at least one of $\mathcal{A}^0$ and $\mathcal{A}^1$ succeeds in biasing the output of the coin-tossing protocol when running with an honest P_1.

The general case – removing the simplifying assumptions. In our proof above, we relied on three simplifying assumptions regarding the coin-tossing protocol:

- The hybrid model allows invocations of the function f only, and not invocations of f^T.
- Honest parties agree on the same output with probability 1.
- The leaves of the tree fully define the output. Namely, the output of the parties are a deterministic function of the transcript of the protocol.

The first item is solved as follows. Instead of considering protocols where the only invocations are to the function f, we consider protocols where each round consists of an invocation of f, followed by an invocation of f^T. In the execution tree, each node at an even level indicates an invocation of f, whereas nodes at an odd level indicate executions of f^T. The variables $\gamma^v, left(v), right(v)$ are defined similarly as before.

As the analysis shows, at any execution there is a "jump" and there exists a node v for which $|p_{right(v)} - p_v|$ or $|p_{left(v)} - p_v|$ is greater than ϵ. This holds also for our case as well, where the tree consists nodes of f^T and f. It is easy to see that in the case that all the nodes v are in fact invocations of f, the same adversaries as before and the same analysis hold, and the adversaries can bias the result. However, in general, the "jump" may occur on executions of f^T. However, in such a case, the adversaries that we have presented control the "x-inputs" and therefore cannot cheat (since it may be that the function is not left-balanced but *is* right-balanced). Indeed, if all the "jumps" in the protocol are in nodes which are executions of f^T, the adversaries that we have proposed may not be able to bias the result at all.

We solve this problem by "splitting" the adversary $\mathcal{A}^1$ into two adversaries: $\mathcal{A}_1^1$ which controls P_1 and adversary $\mathcal{A}_2^1$ which controls P_2 (similarly, we "split" A^0 into $\mathcal{A}_1^0, \mathcal{A}_2^0$ who control P_1 and P_2, respectively.). The adversary $\mathcal{A}_2^1$ "attacks" the protocol exactly as $\mathcal{A}^1$, with the only restriction that it checks that the node v (for which there is a "jump") is an invocation of f. In case where v is an invocation of f^T, the adversary $\mathcal{A}_2^1$ controls the inputs of x and therefore cannot attack at this point. However, the adversary $\mathcal{A}_1^1$ controls the inputs of y at that node, and can perform the same attack as $\mathcal{A}^1$. Overall, the adversary $\mathcal{A}_2^1$ only attacks the protocol on nodes v which are invocations of f, whereas $\mathcal{A}_1^1$ attacks the protocol on nodes v which are invocations of f^T. The overall bias of the two adversaries ($\mathcal{A}_1^1$,$\mathcal{A}_2^1$) in a protocol where the invocations are of f, f^T is exactly the same as the overall bias of the adversary $\mathcal{A}^1$ in a protocol where all the invocations are of the function f. Similarly, we split $\mathcal{A}^0$ into $(\mathcal{A}_1^0, \mathcal{A}_2^0)$, and the overall bias of $\mathcal{A}^0$ is essentially the same as the overall bias of $(\mathcal{A}_1^0, \mathcal{A}_2^0)$. Since the sum of the biases of $(\mathcal{A}^0, \mathcal{A}^1)$ is non-negligible, the sum of the biases of $(\mathcal{A}_1^0, \mathcal{A}_1^1, \mathcal{A}_2^0, \mathcal{A}_2^1)$ is non-negligible as well.

The second item can be dealt with in a straightforward manner. By the requirement on a secure coin-tossing function, the probability that two honest parties output different bits is at most negligible. In executions that reach leaves for which this event occurs, we can just assume that the adversary fails. Since such executions happen with negligible probability, this reduces the success probability of the adversary by at most a negligible amount.

We now turn to the third item. First we explain in what situations the general case might occur. Consider the case where there are two sets of random tapes that reach the same final node, where on one set the parties output 0 and on the other they output 1. This can happen if the final output is based on parties' full views, including their random tapes, rather than on the public transcript of the computation alone.

We now show that any protocol can be converted into a protocol where the transcript fully determines the output of both parties. Given a coin-tossing protocol π we build a protocol π' as follows: The parties invoke the protocol π and at the end of the computation, each party sends its random tape to the other party. In the protocol π', clearly each leaf fully determines the outputs of the parties, since each parties' entire view is in the public transcript. Furthermore, the requirement that two honest parties output the same bit except with negligible probability clearly holds also for π'. Finally, we claim that if there exists an adversary $\mathcal{A}'$ who succeeds in biasing the output of an honest party with non-negligible probability in π', then we can construct an adversary $\mathcal{A}$

who succeeds in biasing the output of an honest party with non-negligible probability in π. The adversary $\mathcal{A}$ simply invokes $\mathcal{A}'$, and aborts at the point where it is supposed to send its random tape. Since the output of the honest party is determined after the invocation of the protocol π and before the random tapes are sent as part of π', the output of the honest party when interacting with $\mathcal{A}$ is identical to its output when interacting with $\mathcal{A}'$. Thus, $\mathcal{A}$ also succeeds to bias the output of the honest party in π.

Given the above, we can transform a general π into π' for which the third simplifying assumption holds. Then, we can build the adversaries as proven for π', under this assumption. By what we have just shown, this implies the existence of an adversary who can bias the output of the honest party in π, as required. This concludes the proof. ■

Impossibility for 0 or 1-balanced functions. The above theorem proves impossibility for the case that the function is not balanced. As we have mentioned, we must separately deal with the case that the function *is* balanced, but not *strictly* balanced; i.e., the function is either 0-balanced or 1-balanced. The main difference in this case is that not all nodes which have significantly different probabilities in their two children can be used by the adversary to bias the outcome. This is due to the fact that the protocol may specify an input distribution distribution for the honest party at such a node that forces the output to be either 0 or 1 (except with negligible probability), and so the "different child" is only reached with negligible probability. This can happen since the function *is* balanced with $\delta = 0$ or $\delta = 1$. Thus, there may be probability vectors for which $\delta_{\max} - \delta_{\min}$ is 0 or some negligible function. The proof therefore shows that this cannot happen too often, and the adversary can succeed enough to bias the output.

Theorem 5.4 *Let $f : \{x_1, \ldots, x_m\} \times \{y_1, \ldots, y_\ell\} \to \{0, 1\}$ be a 0 or 1-balanced function. Then, f does not information-theoretically imply the coin-tossing protocol.*

Proof: By Claim 3.5, since f is 0 or 1-balanced, it contains the all-zero row and all-zero column, or the all-one row and the all-one column. Thus, if f is 0-balanced, an adversary can "force" the output of any given invocation of f to be 0, and so $\delta^v_{\min}$ (as defined in the proof of Claim 5.3) equals 0. Likewise, if f is 1-balanced, an adversary can force the output to be 1 and thus $\delta^v_{\max} = 1$. We use the same ideas as the proof of Theorem 5.1. We consider the transcript tree with the same simplifications as above (in particular, we consider protocols with invocations of f only, and no invocations of f^T). The transformation to the general case is the same, and we omit this part. In this proof, we set $\epsilon = \epsilon(n) = \frac{1/4 - \mu(n)}{r(n)}$. We start by rewriting the adversaries, and we write the differences in **bold**.

> ***Adversary $\mathcal{A}^1$ controlling P_2 (biasing towards 1):***
> Let v be the current node in the tree $\mathcal{T}$ (starting at the root) and $\boldsymbol{\alpha}^v = (\alpha^v_1, \ldots, \alpha^v_n)$ be the input distribution for P_1 (as explained above, this can be computed). Compute $(\delta^v_1, \ldots, \delta^v_\ell) = \boldsymbol{\alpha}^v \cdot M_f$. Let i and k be indices such that $\delta^v_i = \max_{1 \le j \le \ell}\{\delta^v_j\}$ and $\delta^v_k = \min_{1 \le j \le \ell}\{\delta^v_j\}$ (denote $\delta^v_{\max} = \delta^v_i$ and $\delta^v_{\min} = \delta^v_k$). In the *first* node v in the execution with two children such that $|p_{right(v)} - p_v| \ge \epsilon$ or $|p_{left(v)} - p_v| \ge \epsilon$ **and $\boldsymbol{\epsilon \le \gamma^v \le 1 - \epsilon}$** (checking also that $\epsilon \le \gamma^v \le 1 - \epsilon$ is the only difference from previously), act according to the following:
>
> > If $p_{right(v)} \ge p_v + \epsilon$ or $p_{left(v)} \le p_v - \epsilon$ then send input y_i in this invocation of f (this increases the probability of reaching node $right(v)$ because the probability

of obtaining 1 in this invocation is δ_i which is *maximal*). If $p_{left(v)} \geq p_v + \epsilon$ or $p_{right(v)} \leq p_v - \epsilon$ then send input y_k in this invocation of f (this increases the probability of reaching node $left(v)$ because the probability of obtaining 0 in this invocation is δ_k which is *minimal*).

In all other nodes act honestly.

The adversary $\mathcal{A}^0$ controlling P_2 (biasing towards 0): The adversary is the same as $\mathcal{A}^1$ as defined above with the same differences as in the previous proof.

We now turn to the analysis. Let $\mathcal{V}$ denote the set of all nodes for which the adversaries do not act honestly and attempt to bias the result (i.e., the nodes that fulfill the conditions above). Observe that $\mathcal{V}$ is a subset of the set $\mathcal{V}$ defined in the proof of Theorem 5.1. We compute the sum of differences between the output probability at $v \in \mathcal{V}$ when honest parties run, and when an honest party P_1 runs with dishonest P_2. Like the proof of Theorem 5.1, we show that whenever the adversaries reach a node $v \in \mathcal{V}$, they succeed in biasing the result (the proof of this is almost the same as above). Then, we compute the probability that the execution reaches a node in $\mathcal{V}$. However, here, unlike the proof of Theorem 5.1, the probability of reaching a node $v \in \mathcal{V}$ is not 1.

The adversary's effect conditioned on reaching $\mathcal{V}$. We first obtain a bound on the sum of difference between the output probability at $v \in \mathcal{V}$ when honest parties run, and when an honest party P_1 runs with adversaries, $\mathcal{A}^1, \mathcal{A}^0$ respectively, where in both cases we are conditioning on the event that the execution reaches a node $v \in \mathcal{V}$. Since the attack is exactly the same as in the previous proof (Claim 5.3), we conclude that for any node $v \in \mathcal{V}$:

$$\left(\mathsf{out}^0_{|v}(P_1, \mathcal{A}^0) - \mathsf{out}^0_{|v}(P_1, P_2)\right) + \left(\mathsf{out}^1_{|v}(P_1, \mathcal{A}^1) - \mathsf{out}^1_{|v}(P_1, P_2)\right) \geq (\delta^v_{\max} - \delta^v_{\min}) \cdot \epsilon$$

We now compute a lower bound on the term $\delta^v_{\max} - \delta^v_{\min}$. In the above proof for an unbalanced function this was bounded by a constant c. In this case, where f is 0 or 1-balanced, Lemma 3.4 does not hold, and thus the proof is more involved.

First, consider the case that f is 1-balanced. As we have stated above, this implies that $\delta^v_{\max} = 1$ for every node v. We now show that $\delta^v_{\min} \leq 1 - \epsilon$. In order to see this, we note that $\delta^v_{\min} \leq \gamma^v$, for every node v (since γ^v is the "correct probability" of getting 1 and this is at least the minimum probability). Now, for every $v \in \mathcal{V}$, by the definition of $\mathcal{V}$ for this adversary we have that $\gamma^v \leq 1-\epsilon$, and therefore we can conclude that $\delta^v_{\min} \leq 1-\epsilon$, and so $\delta^v_{\max} - \delta^v_{\min} \geq \epsilon$. Next, consider the case that f is 0-balanced. In this case, $\delta^v_{\min} = 0$, but since $\delta^v_{\max} \geq \gamma^v$ (for the same reason that $\delta^v_{\min} \leq \gamma^v$) and since $\gamma^v \geq \epsilon$ by the definition of the adversary, we have that $\delta^v_{\max} \geq \epsilon$ and so once again $\delta^v_{\max} - \delta^v_{\min} \geq \epsilon$.

Combining the above, we have that for every $v \in \mathcal{V}$ the overall sum of the probability changes of the adversaries is:

$$\begin{aligned}\left(\mathsf{out}^0_{|v}(P_1, \mathcal{A}^0) - \mathsf{out}^0_{|v}(P_1, P_2)\right) + \left(\mathsf{out}^1_{|v}(P_1, \mathcal{A}^1) - \mathsf{out}^1_{|v}(P_1, P_2)\right) \\ \geq (\delta^v_{\max} - \delta^v_{\min}) \cdot \epsilon \geq (1 - (1 - \epsilon)) \cdot \epsilon = \epsilon^2\end{aligned}$$

We remark that for the case where M_f is 0-balanced and not 1-balanced, $\delta^v_{\min} = 0$, but it follows that $\delta^v_{\max} \geq \gamma^v \geq \epsilon$, and so $\delta^v_{\max} - \delta^v_{\min} \geq \epsilon$, as well.

Computing the probability to reach $\mathcal{V}$. We have shown that when the parties reach nodes from $\mathcal{V}$, the adversaries succeed in biasing the result. Now, whenever the adversaries do not reach nodes from $\mathcal{V}$, they play honestly and therefore both parties are expected to output a fair coin. Therefore, if the execution reaches a node in $\mathcal{V}$ with non-negligible probability, this will imply non-negligible bias, as required. We now show that the probability that an execution reaches a node in $\mathcal{V}$ is at least 1/2 (this is the key difference to the unbalanced case where *every* execution reached a node in $\mathcal{V}$).

Every pair of random coins (r_1, r_2) for the parties fully determine a path from the root to one of the leaves, and each leaf defines the output for both parties. For each node v in the tree $\mathcal{T}$, we consider its type according to the following:

(a) $|p_{right(v)} - p_v| < \epsilon$ and $|p_{left(v)} - p_v| < \epsilon$, or v has only one child

(b) $|p_{right(v)} - p_v| \geq \epsilon$ and $\epsilon \leq \gamma^v \leq 1 - \epsilon$

(c) $|p_{right(v)} - p_v| \geq \epsilon$ and $\gamma^v < \epsilon$

(d) $|p_{left(v)} - p_v| \geq \epsilon$ and $\epsilon \leq \gamma^v \leq 1 - \epsilon$

(e) $|p_{left(v)} - p_v| \geq \epsilon$ and $\gamma^v > 1 - \epsilon$

We first prove that these cases cover all possibilities. Specifically, we claim that it cannot hold that $|p_{right(v)} - p_v| \geq \epsilon$ and $\gamma^v > 1 - \epsilon$. Moreover, it cannot hold that $|p_{left(v)} - p_v| \geq \epsilon$ and $\gamma^v < \epsilon$. In order to see this, first note that since $p_v = \gamma^v \cdot p_{right(v)} + (1 - \gamma^v) \cdot p_{left(v)}$, we can write:

$$\left|p_{right(v)} - p_v\right| = \left|p_{right(v)} - \gamma^v \cdot p_{right(v)} - (1 - \gamma^v) \cdot p_{left(v)}\right| = (1 - \gamma^v) \cdot \left|p_{right(v)} - p_{left(v)}\right|, \quad (3)$$

and,

$$\left|p_{left(v)} - p_v\right| = \left|p_{left(v)} - \gamma^v \cdot p_{right(v)} - (1 - \gamma^v) \cdot p_{left(v)}\right| = \gamma^v \cdot \left|p_{left(v)} - p_{right(v)}\right|. \quad (4)$$

Now, $|p_{right(v)} - p_{left(v)}| \leq 1$ because they are both probabilities, and thus $|p_{right(v)} - p_v| \leq 1 - \gamma^v$. If $|p_{right(v)} - p_v| \geq \epsilon$ then this implies that $1 - \gamma^v \geq \epsilon$ and thus $\gamma^v \leq 1 - \epsilon$. Thus, it cannot be that $|p_{right(v)} - p_v| \geq \epsilon$ and $\gamma^v > 1 - \epsilon$, as required. Similarly, if $|p_{left(v)} - p_v| \geq \epsilon$ then $\gamma^v \geq \epsilon$ and thus it cannot be that $|p_{left(v)} - p_v| \geq \epsilon$ and $\gamma^v < \epsilon$.

Next, recall that the nodes in $\mathcal{V}$ are all of type (d) or (b). We now show that for any node of type (e), it must hold that $|p_v - p_{right(v)}| < \epsilon$. Similarly, for any node of type (c), it must hold that $|p_v - p_{left(v)}| < \epsilon$. We explain how we use this, immediately after the proof of the claim.

Claim 5.5 *Let v be a node in the tree $\mathcal{T}$ with two children. Then:*

1. *If $\gamma^v > 1 - \epsilon$, then $|p_{right(v)} - p_v| < \epsilon$.*
2. *If $\gamma^v < \epsilon$, then $|p_{left(v)} - p_v| < \epsilon$.*

Proof: For the first item, by Eq. (3) we have that $|p_{right(v)} - p_v| \leq 1 - \gamma^v$. In addition, if $\gamma^v > 1 - \epsilon$ then $1 - \gamma^v < \epsilon$. We conclude that $\left|p_{right(v)} - p_v\right| \leq 1 - \gamma^v < \epsilon$, as required. The second item is proven analogously using Eq. (4). ■

The claim above shows that for any node v of type (e) the right child has output probability that is close to the probability of the parent, and for any node of type (c) the left child has output

probability that is close to the probability of the parent. This means that little progress towards the output is made in this invocation. Intuitively, it cannot be the case that little progress is made in *all* invocations. We will now show this formally.

Let $u_1, \ldots, u_{r(n)}$ be a path in the tree $\mathcal{T}$ from the root u_1 to some leaf $u_{r(n)}$. Recall that any execution defines a unique path from the root and one of the leaves. The next claim shows that there does not exist a path in the tree (associated with a pair of random tapes (r_1, r_2)) such that all the nodes in the path are of types (a), (e) and (c), and in all the nodes of type (e) the path proceeds to the right son and in all the nodes of type (c) the path proceeds to the left son.

Claim 5.6 *There does not exist a path $u_1, \ldots, u_{r(n)}$ for which for every $i = 1, \ldots, r(n) - 1$ one of the following conditions hold:*

1. $|p_{right(u_i)} - p_{u_i}| < \epsilon$ *and* $|p_{left(u_i)} - p_{u_i}| < \epsilon$*, or u_i has only one child* (i.e., u_i is type (a)).
2. $|p_{left(u_i)} - p_{u_i}| \geq \epsilon$*,* $\gamma^{u_i} > 1 - \epsilon$ *and* $u_{i+1} = right(u_i)$ (that is, u_i is type (e), and the path proceeds to the right son).
3. $|p_{right(u_i)} - p_{u_i}| \geq \epsilon$*,* $\gamma^{u_i} < \epsilon$ *and* $u_{i+1} = left(u_i)$. (that is, u_i is type (c), and the path proceeds to the left son).

Proof: Assume by contradiction that there exists a path $u_1, \ldots, u_{r(n)}$ for which for every $i = 1, \ldots, r(n) - 1$, one of the above conditions holds. Let u_i be a node. If the first item holds for u_i, then $|p_{u_{i+1}} - p_{u_i}| < \epsilon$ (observe that if u_i has just one child then $p_{u_{i+1}} = p_{u_i}$ and so $|p_{u_{i+1}} - p_{u_i}| < \epsilon$). If the second item holds for u_i, then this is a node of type (e) and the path proceeds to the right son. By Claim 5.5, this implies that $|p_{u_{i+1}} - p_{u_i}| < \epsilon$ (observe that the claim can be applied since the node u_i has two children by the fact that it is not of type (a)). Likewise, if the third item holds for u_i, then this is a node of type (c) and the party proceeds to the left son, and so by Claim 5.5 it holds that $|p_{u_{i+1}} - p_{u_i}| < \epsilon$. We conclude that for every $i = 1, \ldots, r(n) - 1$, $|p_{u_{i+1}} - p_{u_i}| < \epsilon$, and so:

$$|p_{u_{r(n)}} - p_{u_1}| \leq \sum_{i=1}^{r(n)} |p_{u_{i+1}} - p_{u_i}| < \sum_{i=1}^{r(n)} \epsilon = r(n) \cdot \epsilon = r(n) \cdot \frac{1/4 - \mu(n)}{r(n)} = 1/4 - \mu(n).$$

Now, recall that $|p_{u_1} - 1/2| < \mu(n)$ for some negligible function $\mu(n)$. This implies that $p_{u_{r(n)}}$ is non-negligibly far from the range $\left[\frac{1}{4}, \frac{3}{4}\right]$, in contradiction to the fact that it equals to 0 or 1. This completes the proof. ■

Recall that nodes in $\mathcal{V}$ are of type (b) and (d) only, and so if they are reached in an execution then the adversary succeeds in biasing the result. In addition, we have shown that it cannot be the case that an entire execution is made up of nodes of type (a) and nodes of type (e) where the execution when to the right and nodes of type (c) in which the execution went to the left. Thus, if no node in $\mathcal{V}$ is reached, it *must be the case* that a node of type (e) was reached and the execution went to the left, meaning that the output of the invocation of f was 0, or the execution reached a node of type (c) and the execution went to the right, meaning that the output of f was 1. We now show that the probability of this occurring is small. That is, we bound the following probability:

$$\Pr\left[\exists j \text{ for which } \mathsf{type}(u_j) = e : u_{j+1} = right(u_j) \quad \vee \quad \exists j \text{ for which } \mathsf{type}(u_j) = c : u_{j+1} = left(u_j)\right].$$

By the union bound, it is enough to compute the probability of each one of the terms. Observe that:

$$\Pr\left[u_{j+1} = left(u_j) \mid \mathsf{type}(u_j) = (e)\right] < \epsilon$$

because by the definition of type (e), $\gamma^{u_j} > 1 - \epsilon$. We therefore have:

$$\begin{aligned}
&\Pr\left[\exists j \text{ for which } \mathsf{type}(u_j) = c : u_{j+1} = left(u_j)\right] \\
&\leq \sum_{j \text{ s.t. } \mathsf{type}(u_j)=c} \Pr\left[u_{j+1} = left(u_j) \mid \mathsf{type}(u_j) = c\right] \\
&< r(n) \cdot \epsilon \\
&= r(n) \cdot \frac{1/4 - \mu(n)}{r(n)} \\
&= 1/4 - \mu(n).
\end{aligned}$$

In a similar way, we conclude that the probability to reach a node of type (c) and to move to the right child is bounded by $1/4 - \mu(n)$. Therefore, the probability of not reaching a node in $\mathcal{V}$ is bounded by $1/2 - 2\mu(n)$.

Thus the probability of reaching a node in $\mathcal{V}$ is greater than $1/2$. That is, we have that:

$$\sum_{v \in \mathcal{V}} \mathsf{reach}_v > \frac{1}{2}$$

where for every $v \in \mathcal{V}$, reach_v denotes the probability to reach the node v (which, as discussed in the proof of the unbalanced case, is the same in all executions of $\langle P_1, P_2^* \rangle$, where $P_2^* \in \{P_2, \mathcal{A}^0, \mathcal{A}^1\}$).

Concluding the proof. We have seen that the probability of reaching a node in $\mathcal{V}$ is greater than $1/2$. Moreover, whenever the adversaries reach such a node, the sum of the differences between honest execution and execution with the adversaries is ϵ^2. Combining the above, we conclude that:

$$\begin{aligned}
&\left(\mathsf{out}^1(P_1, \mathcal{A}^1) - \mathsf{out}^1(P_1, P_2)\right) + \left(\mathsf{out}^0(P_1, \mathcal{A}^1) - \mathsf{out}^0(P_1, P_2)\right) \\
&= \sum_{v \in \mathcal{V}} \mathsf{reach}_v \cdot \left[\left(\mathsf{out}^1_{|v}(P_1, \mathcal{A}^1) - \mathsf{out}^1_{|v}(P_1, P_2)\right) + \left(\mathsf{out}^0_{|v}(P_1, \mathcal{A}^0) - \mathsf{out}^0_{|v}(P_1, P_2)\right)\right] \\
&\geq \sum_{v \in \mathcal{V}} \mathsf{reach}_v \cdot \left(\epsilon^2\right) > \frac{1}{2} \cdot \epsilon^2
\end{aligned}$$

Similarly to the proof of Theorem 5.1, this implies that:

$$\left(\mathsf{out}^0(P_1, \mathcal{A}^0) - \frac{1}{2}\right) + \left(\mathsf{out}^1(P_1, \mathcal{A}^1) - \frac{1}{2}\right) > \frac{1}{2} \cdot \epsilon^2 - 2\mu(n).$$

and so one of the adversaries succeeds in biasing the result with non-negligible probability. This concludes the proof. ■

Conclusion: Combining Theorems 4.1, 5.1 and 5.4, we obtain the following corollary:

Corollary 5.7 *Let $f : \{x_1, \ldots, x_m\} \times \{y_1, \ldots, y_\ell\} \to \{0, 1\}$ be a function.*

- *If f is strictly-balanced, then f implies the coin-tossing functionality (computationally and information-theoretically).*
- *If f is not strictly-balanced, then f does not information-theoretically imply the coin-tossing functionality in the presence of malicious adversaries.*

Impossibility in the OT-hybrid model. Our proof of impossibility holds in the information-theoretic setting only since the adversary must carry out computations that do not seem to be computable in polynomial-time. It is natural to ask whether or not the impossibility result still holds in the computational setting. We do not have an answer to this question. However, as a step in this direction, we show that the impossibility still holds if the parties are given access to an ideal oblivious transfer (OT) primitive as well as to the function f. That is, we prove the following:

Theorem 5.8 *Let $f : \{x_1, \ldots, x_m\} \times \{y_1, \ldots, y_\ell\} \to \{0,1\}$ be a function. If f is not strictly-balanced, then the pair of functions (f, OT) do not information-theoretically imply the coin tossing functionality in the presence of malicious adversaries.*

Proof Sketch: In order to see that this is the case, first observe that if f has an embedded-OR then it implies oblivious transfer [9]. Thus, f can be used to obtain OT, and so the question of whether f implies coin tossing or (f, OT) imply coin tossing is the same. It thus remains to consider the case that f does not have an embedded OR but does have an embedded XOR (if it has neither then it is trivial and so clearly cannot imply coin tossing, as we have mentioned). We now show that in such a case f must be strictly balanced, and so this case is not relevant. Let x_1, x_2, y_1, y_2 be an embedded XOR in f; i.e., $f(x_1, y_1) = f(x_2, y_2) \neq f(x_1, y_2) = f(x_2, y_1)$. Now, if there exists a y_3 such that $f(x_1, y_3) = f(x_2, y_3)$ then f has an embedded OR. Thus, x_1 and x_2 must be complementary rows (as in example function (a) in the Introduction). Likewise, if there exists an x_3 such that $f(x_3, y_1) = f(x_3, y_2)$ then f has an embedded OR. Thus, y_1 and y_2 must be complementary columns. We conclude that f has two complementary rows and columns, and as we have shown in the Introduction, this implies that f is strictly balanced with $\delta = \frac{1}{2}$. ■

6 Fairness in the Presence of Fail-Stop Adversaries

In order to study the feasibility of achieving fair secure computation in the fail-stop model, we must first present a definition of security for this model. To the best of our knowledge, there is no simulation-based security definition for the fail-stop model in the literature. As we have mentioned in the introduction, there are two natural ways of defining security in this model, and it is not a priori clear which is the "correct one". We therefore define two models and study feasibility for both. In the first model, the ideal-model adversary/simulator must either send the party's prescribed input to the trusted party computing the function, or a special abort symbol $\perp$, but nothing else. This is similar to the semi-honest model, except that $\perp$ can be sent as well. We note that if $\perp$ is sent, then both parties obtain $\perp$ for output and thus fairness is preserved.[6] This is actually a very strong requirement from the protocol since both parties either learn the prescribed output, or they both output $\perp$. In the second model, the ideal adversary can send any input that it wishes to the trusted party, just like a malicious adversary. We remark that if the real adversary does not abort a real protocol execution, then the result is the same as an execution of two honest parties and thus the output is computed from the prescribed inputs. This implies that the ideal adversary can really only send a different input in the case that the real adversary halts before the protocol is completed. As we have mentioned in the Introduction, the impossibility result of

[6] It is necessary to allow an explicit abort in this model since if the corrupted party does not participate at all then output cannot be computed. The typical solution to this problem, which is to take some default input, is not appropriate here because this means that the simulator can change the input of the corrupted party. Thus, such an early abort must result in output $\perp$.

Cleve [4] for coin-tossing holds in the both models, since the parties have no input, and so for this functionality the models are identical.

6.1 Fail-Stop 1

In this section we define and explore the first fail-stop model. We proceed directly to define the ideal model:

Execution in the ideal world. 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^n 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 its prescribed input or $\perp$.

Trusted party sends outputs: If an input $\perp$ was received, then the trusted party sends $\perp$ to both parties. Otherwise, it computes $f(x, y)$ and sends the result to both parties.

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 $\text{IDEAL}^{\text{f-stop-1}}_{f,\mathcal{S}(z)}(x, y, n)$ 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.

Security. The real model is the same as is defined in Section 2, except that we consider adversaries that are fail-stop only. This means that the adversary must behave exactly like an honest party, except that it can halt whenever it wishes during the protocol. We stress that its decision to halt or not halt, and where, may depend on its view. We are now ready to present the security definition.

Definition 6.1 (Security – fail-stop1) *Protocol π* securely computes *f* with complete fairness in the fail-stop1 model *if for every non-uniform probabilistic polynomial-time fail-stop adversary $\mathcal{A}$ in the real model, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ in the ideal model such that:*

$$\left\{\text{IDEAL}^{\text{f-stop-1}}_{f,\mathcal{S}(z)}(x_1, x_2, n)\right\}_{x\in X, y\in Y, z\in\{0,1\}^*, n\in\mathbb{N}} \stackrel{\text{c}}{\equiv} \left\{\text{REAL}_{\pi,\mathcal{A}(z)}(x, y, n)\right\}_{x\in X, y\in Y, z\in\{0,1\}^*, n\in\mathbb{N}} .$$

Exploring fairness in the fail-stop-1 model. We first observe that if a function contains an embedded XOR, then it cannot be computed fairly in this model.

Theorem 6.2 *Let $f : \{x_1, \ldots, x_m\} \times \{y_1, \ldots, y_\ell\} \to \{0, 1\}$ be a function that contains an embedded XOR. Then, f implies the coin-tossing functionality and thus cannot be computed fairly.*

Proof: Assume that f contains an embedded XOR; i.e., there exist inputs x_1, x_2, y_1, y_2 such that $f(x_1, y_1) = f(x_2, y_2) \neq f(x_1, y_2) = f(x_2, y_1)$. We can easily construct a protocol for coin-tossing using f that is secure in the fail-stop model. Party P_1 chooses input $x \in \{x_1, x_2\}$ uniformly at

random, P_2 chooses $y \in \{y_1, y_2\}$ uniformly at random, and the parties invoke the function f where P_1 inputs x and P_2 inputs y. In case the result of the invocation is $\perp$, the other party chooses its output uniformly at random.

Since the adversary is fail-stop1, it must follow the protocol specification or abort prematurely. In both cases, it is easy to see that the honest party outputs an unbiased coin. Formally, for any given fail-stop adversary $\mathcal{A}$ we can construct a simulator $\mathcal{S}$: $\mathcal{S}$ receives from the coin tossing functionality f^{ct} the bit b, and invokes the adversary $\mathcal{A}$. If $\mathcal{A}$ sends the trusted party computing f the symbol $\perp$, then $\mathcal{S}$ responds with $\perp$. Otherwise, (if $\mathcal{A}$ sends some real value - either x_1, x_2 if it controls P_1, or y_1, y_2 if it controls P_2), then $\mathcal{S}$ responds with the bit b that it received from f^{ct} as if it is the output of the ideal call to f. It is easy to see that the ideal and real distributions are identical. ■

As we have mentioned, if a function does not contain an embedded XOR or an embedded OR then it is trivial and can be computed fairly (because the output depends on only one of the party's inputs). It therefore remains to consider the feasibility of fairly computing functions that have an embedded OR but no embedded XOR. Gordon et. al [8] present a protocol for securely computing any function of this type with complete fairness, in the presence of a malicious adversary. However, the security of their protocol relies inherently on the ability of the simulator to send the trusted party an input that is not the corrupted party's prescribed input. Thus, their protocol seems not to be secure in this model.

The problem of securely computing functions that have an embedded OR but no embedded XOR therefore remains open. We remark that there are very few functions of this type, and any such function has a very specific structure, as discussed in [8].

6.2 Fail-Stop 2

In this section we define and explore the second fail-stop model. In this case, the ideal adversary can send any value it wishes to the trusted party (and the output of the honest party is determined accordingly). It is easy to see that in executions where the real adversary does not abort the output is the same as between two honest parties. Thus, the ideal adversary is forced to send the prescribed input of the party in this case. Observe that the ideal model here is identical to the ideal model for the case of malicious adversaries. Thus, the only difference between this definition and the definition of security for malicious adversaries is the quantification over the real adversary; here we quantify only over fail-stop real adversaries. Otherwise, all is the same.

Definition 6.3 (Security – fail-stop2) *Protocol* π securely computes f with complete fairness in the fail-stop2 model *if for every non-uniform probabilistic polynomial-time fail-stop adversary* $\mathcal{A}$ *in the real world, there exists a non-uniform probabilistic polynomial-time adversary* $\mathcal{S}$ *in the ideal model such that:*

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

In the g-hybrid-model for fail-stop2 adversaries, where the parties have access to a trusted party computing function g for them, a corrupted party may provide an incorrect input to an invocation of g as long as it halts at that point. This may seem arbitrary. However, it follows naturally from the definition since a secure fail-stop2 protocol is used to replace the invocations of g in the real model. Thus, if a fail-stop adversary can change its input as long as it aborts in the real model, then this capability is necessary also for invocations of g in the g-hybrid model.

Exploring fairness in the fail-stop-2 model. In the following we show that the malicious adversaries that we constructed in the proofs of Theorem 5.1 and Theorem 5.4 can be modified to be *fail-stop2*. We remark that the adversaries that we constructed did not abort during the protocol execution, but rather continued after providing a "different" input in one of the f invocations. Thus, they are not fail-stop2 adversaries. In order to prove the impossibility for this case, we need to modify the adversaries so that they *halt* at the node v for which they can bias the outcome of the invocation (i.e., a node v for which v's children in the execution tree have significantly different probabilities for the output of the entire execution equalling 1). Recall that in this fail-stop model, the adversary is allowed to send a different input than prescribed in the invocation at which it halts; thus, this is a valid attack strategy.

We prove impossibility in this case by considering two possible cases, relating to the potential difference between the honest party outputting 1 at a node when the other party aborts at that node but until then was fully honest, or when the other party continues honestly from that node (to be more exact, we consider the average over all nodes).

1. First, assume that there is a noticeable difference between an abort after fully honest behavior and a fully honest execution. In this case, we construct a fail-stop adversary who plays fully honestly until an appropriate node where such a difference occurs and then halts. (In fact, such an adversary is even of the fail-stop1 type).

2. Next, assume that there is *no noticeable difference* between an abort after fully honest behavior and a fully honest execution. Intuitively, this means that continuing honestly or halting makes no difference. Thus, if we take the malicious adversaries from Section **??** and modify them so that they halt immediately after providing malicious input (as required in the fail-stop2 model), then we obtain that there is no noticeable difference between the original malicious adversary and the fail-stop2 modified adversary. We remark that this is not immediate since the difference in this case is between aborting and not aborting without giving any malicious input. However, as we show, if there is no difference when honest inputs are used throughout, then this is also no difference when a malicious input is used.

We conclude that one of the two types of fail-stop2 adversaries described above can bias any protocol.

Theorem 6.4 *Let $f:\{x_1,\ldots,x_m\}\times\{y_1,\ldots,y_\ell\}\to\{0,1\}$ be a function that is* not *δ-balanced, for any $0<\delta<1$. Then, f does not information-theoretically imply the coin-tossing protocol in the fail-stop2 model.*

Proof: We follow the proofs of Theorem 5.4 and use the same ideas. Recall that the adversaries play honestly until they reach a node in $\mathcal{V}$. Then, the adversaries send an input value to the invocation of f not according to the protocol, but continue to play honestly afterward. In our case, we define similar adversaries: the adversaries play honestly until they reach a node in $\mathcal{V}$. Then, they send an input value not according to the protocol and abort. This is allowed in our model: the adversaries play honestly and according to the protocol until they reach a node for which they quit. In the single invocation of f for which they abort they send a different value to that specified by the protocol. This is allowed since in the fail-stop2 model the ideal simulator is allowed to send any input it desires in the case of an early abort.

We recall that the adversaries reach a node in the set $\mathcal{V}$ with probability greater than $1/2$. We now analyze whether they succeed to bias the result when the execution reaches a node in $\mathcal{V}$.

Aborting vs. continuing honestly. At each node v of the transcript tree $\mathcal{T}$, we write in addition to the probability of outputting 1 when both parties act honestly (i.e., p_v), the probability that the output of the honest party is 1 if the execution is terminated in the node v (i.e., the "default output" upon abort). We denote this probability by q_v. Now, recall the definition of the set $\mathcal{V}$: the set of nodes for which the adversary does not act honestly and changes the input. As mentioned above, the adversary act honestly until it reaches a node in $\mathcal{V}$. Then, in this node it aborts and so may send an incorrect value to the invocation of f, and the honest party receives 0 or 1 from the invocation according the the inputs that were sent. In all the following invocations of f, the honest party receives $\perp$ and thus it chooses its output according to the probability $q_{right(v)}$ (if the execution proceeded to the right child of v in the last invocation of f) or $q_{left(v)}$ (if the execution proceeded to the left child). Recall that γ^v denotes the probability of receiving output 1 from the invocation of f at the node v in an honest execution, and thus this is the probability that the execution proceeds to the right child of v. Moreover, recall that in both proofs, each node $v \in \mathcal{V}$ has two children. We have the following claim:

Claim 6.5 *Let $\mathcal{V}$ be as above. Then, there exists a negligible function $\mu'(\cdot)$ such that:*

$$\sum_{v \in \mathcal{V}} \left(|p_{left(v)} - q_{left(v)}| + |p_{right(v)} - q_{right(v)}| \right) \cdot \mathsf{reach}_v \leq \mu'(n)$$

Proof: Assume in contradiction that the claim does not hold. Then, there exists a non-negligible function $\omega(n)$ such that:

$$\sum_{v \in \mathcal{V}} \left(|p_{left(v)} - q_{left(v)}| + |p_{right(v)} - q_{right(v)}| \right) \cdot \mathsf{reach}_v \geq \omega(n)$$

We construct an adversary that succeed to bias the coin with some non-negligible probability.

Let $\mathcal{V}' = \{left(v), right(v) \mid v \in \mathcal{V}\}$, i.e., all the children of nodes in $\mathcal{V}$, and define the sets $\mathcal{V}'_p = \{v' \in \mathcal{V}' \mid p_{v'} \geq q_{v'}\}$ and $\mathcal{V}'_q = \{v' \in \mathcal{V}' \mid p_{v'} < q_{v'}\}$. Note that $\mathcal{V}'_p$ and $\mathcal{V}'_q$ constitute a partitioning of the set $\mathcal{V}'$. We now consider two adversaries $\mathcal{B}^0$ and $\mathcal{B}^1$ that work as follows: the adversary $\mathcal{B}^0$ aborts at each node $v' \in \mathcal{V}_p$, and $\mathcal{B}^1$ aborts at each node for which $v' \in \mathcal{V}_q$ (that is, both adversaries send $\perp$ to the invocation of f in the node v'), but play completely honestly until that point (these adversaries never send "incorrect" input in an invocation of f).

Let $v' \in \mathcal{V}_p$. The difference in the probability that an honest party outputs 0 in an honest execution and when interacting with the adversary $\mathcal{B}^0$, conditioned on the event that the node v' is reached, is:

$$\begin{aligned}
\mathsf{out}^0_{|v'}(P_1, \mathcal{B}^0) - \mathsf{out}^0_{|v'}(P_1, P_2) &= \Pr_{(r_1,r_2) \leftarrow \mathsf{Uni}}\left[\mathsf{out}\langle P_1(r_1), \mathcal{B}^0(r_2)\rangle = 0 \;\middle|\; v' \text{ is reached}\right] \\
&\quad - \Pr_{(r_1,r_2) \leftarrow \mathsf{Uni}}\left[\mathsf{out}\langle P_1(r_1), P_2(r_2)\rangle = 0 \;\middle|\; v' \text{ is reached}\right] \\
&= (1 - q_{v'}) - (1 - p_{v'}) = p_{v'} - q_{v'} \ .
\end{aligned}$$

Recall that the adversary $\mathcal{B}^1$ plays honestly when it reaches a node $v' \in \mathcal{V}'_p$, and so for such nodes $v' \in \mathcal{V}'_p$,

$$\mathsf{out}^0_{|v'}(P_1, \mathcal{B}^1) - \mathsf{out}^0_{|v'}(P_1, P_2) = 0.$$

Next, consider a node $v' \in \mathcal{V}'_q$. The difference in the probability that an honest party outputs 1 in an honest execution and when interacting with adversary $\mathcal{B}^1$, conditioned on the event that v' is reached, is:

$$\begin{aligned}\mathsf{out}^1_{|v'}(P_1,\mathcal{B}^1) - \mathsf{out}^1_{|v'}(P_1,P_2) &= \Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}\Big[\mathsf{out}\langle P_1(r_1),\mathcal{B}^1(r_2)\rangle = 1 \;\Big|\; v' \text{ is reached}\Big] \\ &\quad - \Pr_{(r_1,r_2)\leftarrow\mathsf{Uni}}\Big[\mathsf{out}\langle P_1(r_1),P_2(r_2)\rangle = 1 \;\Big|\; v' \text{ is reached}\Big] \\ &= q_{v'} - p_{v'}\end{aligned}$$

On the other hand, when $v' \in \mathcal{V}'_q$, the adversary $\mathcal{B}^0$ plays honestly, and so for such $v' \in \mathcal{V}'_q$:

$$\mathsf{out}^1_{|v'}(P_1,\mathcal{B}^1) - \mathsf{out}^1_{|v'}(P_1,P_2) = 0.$$

Now, recall that $\mathcal{V}' = \{left(v), right(v) \mid v \in \mathcal{V}\}$. At a node $v \in \mathcal{V}$, since both adversaries behave honestly, we move to the right child with probability γ^v and to the left child with probability $1-\gamma^v$. Therefore, for every $v \in \mathcal{V}$, for every $P^*_2 \in \{P_2, \mathcal{B}^0, \mathcal{B}^1\}$ and for every $b \in \{0,1\}$ it holds that:

$$\mathsf{out}^b_{|v}(P_1,P^*_2) = \gamma^v \cdot \mathsf{out}^b_{|right(v)}(P_1,P^*_2) + (1-\gamma^v)\cdot \mathsf{out}^b_{|left(v)}(P_1,P^*_2),$$

and so, for each one of the adversaries $\mathcal{B} \in \{\mathcal{B}^0,\mathcal{B}^1\}$:

$$\begin{aligned}&\mathsf{out}^b_{|v}(P_1,\mathcal{B}) - \mathsf{out}^b_{|v}(P_1,P_2) \\ &= \gamma^v \cdot \Big(\mathsf{out}^b_{|right(v)}(P_1,\mathcal{B}) - \mathsf{out}^b_{|right(v)}(P_1,P_2)\Big) + (1-\gamma^v)\cdot\Big(\mathsf{out}^b_{|left(v)}(P_1,\mathcal{B}) - \mathsf{out}^b_{|left(v)}(P_1,P_2)\Big)\end{aligned}$$

Recall that for every $v \in \mathcal{V}$, by the definition of $\mathcal{V}$ it holds that $\epsilon \le \gamma^v \le 1-\epsilon$, and thus $\gamma^v \ge \epsilon$ and $1-\gamma^v \ge \epsilon$. Combining the above, for every $v \in \mathcal{V}$ it holds that:

$$\begin{aligned}&\Big(\mathsf{out}^1_{|v}(P_1,\mathcal{B}^1) - \mathsf{out}^1_{|v}(P_1,P_2)\Big) + \Big(\mathsf{out}^0_{|v}(P_1,\mathcal{B}^0) - \mathsf{out}^0_{|v}(P_1,P_2)\Big) \\ &\qquad = \gamma^v \cdot \big|p_{right(v)} - q_{right(v)}\big| + (1-\gamma^v)\cdot\big|p_{left(v)} - q_{left(v)}\big| \\ &\qquad \ge \epsilon \cdot \big(\big|p_{right(v)} - q_{right(v)}\big| + \big|p_{left(v)} - q_{left(v)}\big|\big)\end{aligned}$$

and therefore we conclude:

$$\begin{aligned}&\big(\mathsf{out}^1(P_1,\mathcal{B}^1) - \mathsf{out}^1(P_1,P_2)\big) + \big(\mathsf{out}^0(P_1,\mathcal{B}^0) - \mathsf{out}^0(P_1,P_2)\big) \\ &= \sum_{v\in\mathcal{V}}\Big[\Big(\mathsf{out}^1_{|v}(P_1,\mathcal{B}^1) - \mathsf{out}^1_{|v}(P_1,P_2)\Big) + \Big(\mathsf{out}^0_{|v}(P_1,\mathcal{B}^0) - \mathsf{out}^0_{|v}(P_1,P_2)\Big)\Big]\cdot \mathsf{reach}_v \\ &\ge \epsilon \cdot \sum_{v\in\mathcal{V}}\big(\big|p_{right(v)} - q_{right(v)}\big| + \big|p_{left(v)} - q_{left(v)}\big|\big)\cdot \mathsf{reach}_v \ge \epsilon\cdot\omega(n)\end{aligned}$$

which is non-negligible, in contradiction to out assumption that the coin-tossing protocol is secure. We therefore conclude that there exists a negligible function $\mu'(\cdot)$ such that:

$$\sum_{v\in\mathcal{V}}\big(\big|p_{right(v)} - q_{right(v)}\big| + \big|p_{left(v)} - q_{left(v)}\big|\big)\cdot \Pr[v \text{ is reached}] \le \mu'(n).$$

■

We now consider the bias of the modified fail-stop adversaries $\mathcal{A}^0, \mathcal{A}^1$ as in the proof of Theorem 5.1, conditioned on the event that the adversaries have reached a node $v \in \mathcal{V}$. We have 2 cases:

Case 1: $p_{right(v)} \geq p_v + \epsilon$ or $p_{left(v)} \leq p_v - \epsilon$: Observe that in the fail-stop case, since the adversary aborts in some node $v \in \mathcal{V}$, the honest party outputs 1 with probability $q_{left(v)}$ or $q_{right(v)}$ and not $p_{left(v)}$ or $p_{right(v)}$. We have that:

$$\begin{aligned}
&\mathsf{out}^1_{|v}(P_1, \mathcal{A}^1) - \mathsf{out}^1_{|v}(P_1, P_2) = (\delta^v_{\max} - \gamma^v) \cdot \left(q_{right(v)} - q_{left(v)}\right) \\
&\quad = (\delta^v_{\max} - \gamma^v) \cdot \left(q_{right(v)} - p_{right(v)} + p_{right(v)} - q_{left(v)} + p_{left(v)} - p_{left(v)}\right) \\
&\quad = (\delta^v_{\max} - \gamma^v) \cdot \left((p_{right(v)} - p_{left(v)}) + (q_{right(v)} - p_{right(v)}) + (p_{left(v)} - q_{left(v)})\right) \\
&\quad \geq (\delta^v_{\max} - \gamma^v) \cdot \left((p_{right(v)} - p_{left(v)}) - |q_{right(v)} - p_{right(v)}| - |p_{left(v)} - q_{left(v)}|\right).
\end{aligned}$$

In a similar way:

$$\begin{aligned}
&\mathsf{out}^0_{|v}(P_1, \mathcal{A}^0) - \mathsf{out}^0_{|v}(P_1, P_2) = (\gamma^v - \delta^v_{\min}) \cdot \left(q_{right(v)} - q_{left(v)}\right) \\
&\quad \geq (\gamma^v - \delta^v_{\min}) \cdot \left((p_{right(v)} - p_{left(v)}) - |q_{right(v)} - p_{right(v)}| - |p_{left(v)} - q_{left(v)}|\right).
\end{aligned}$$

Summing up the differences in probabilities we derive:

$$\begin{aligned}
&\left(\mathsf{out}^1_{|v}(P_1, \mathcal{A}^1) - \mathsf{out}^1_{|v}(P_1, P_2)\right) + \left(\mathsf{out}^0_{|v}(P_1, \mathcal{A}^0) - \mathsf{out}^0_{|v}(P_1, P_2)\right) \\
&\geq (\delta^v_{\max} - \delta^v_{\min}) \cdot \left(p_{right(v)} - p_{left(v)}\right) - (\delta^v_{\max} - \delta^v_{\min}) \cdot \left(|q_{right(v)} - p_{right(v)}| + |p_{left(v)} - q_{left(v)}|\right)
\end{aligned}$$

Now, recall that in the proof of Theorem 5.1, we showed that $\left(p_{right(v)} - p_{left(v)}\right) \geq \epsilon$. In addition, in the case that f is not balanced, there exists a constant $c > 0$ such that $\delta^v_{\max} - \delta^v_{\min} \geq c$, and in the case that f is 1 or 0-balanced it holds that $\delta^v_{\max} - \delta^v_{\min} \geq \epsilon$ (as shown in the proof of Theorem 5.4). In all cases, there exists a non-negligible value Δ such that $\delta^v_{\max} - \delta^v_{\min} \geq \Delta$. Moreover, since $\delta^v_{\max}, \delta^v_{\min}$ are both probabilities it trivially holds that $\delta^v_{\max} - \delta^v_{\min} \leq 1$. Combining all the above, we have:

$$\begin{aligned}
&\left(\mathsf{out}^1_{|v}(P_1, \mathcal{A}^1) - \mathsf{out}^1_{|v}(P_1, P_2)\right) + \left(\mathsf{out}^0_{|v}(P_1, \mathcal{A}^0) - \mathsf{out}^0_{|v}(P_1, P_2)\right) \\
&\qquad \geq \Delta \cdot \epsilon - \left(|q_{right(v)} - p_{right(v)}| + |p_{left(v)} - q_{left(v)}|\right)
\end{aligned}$$

Case 2: $p_{left(v)} \geq p_v + \epsilon$ or $p_{right(v)} \leq p_v - \epsilon$: With similar calculations we derive:

$$\begin{aligned}
&\left(\mathsf{out}^1_{|v}(P_1, \mathcal{A}^1) - \mathsf{out}^1_{|v}(P_1, P_2)\right) + \left(\mathsf{out}^0_{|v}(P_1, \mathcal{A}^0) - \mathsf{out}^0_{|v}(P_1, P_2)\right) \\
&\qquad \geq \Delta \cdot \epsilon - \left(|p_{left(v)} - q_{left(v)}| + |p_{right(v)} - q_{right(v)}|\right)
\end{aligned}$$

where, Δ is the same as above.

Overall bias. We have computed the differences in the probability that an honest party outputs 1 when interacting with an honest P_2 or with one of the adversaries, conditioned on the event that a node $v \in \mathcal{V}$ is reached. We are now ready to compute the overall bias.

$$\begin{aligned}
&\left(\mathsf{out}^1(P_1, \mathcal{A}^1) - \mathsf{out}^1(P_1, P_2)\right) + \left(\mathsf{out}^0(P_1, \mathcal{A}^0) - \mathsf{out}^0(P_1, P_2)\right) \\
&\quad = \sum_{v \in \mathcal{V}} \mathsf{reach}_v \cdot \left[\left(\mathsf{out}^1_{|v}(P_1, \mathcal{A}^1) - \mathsf{out}^1_{|v}(P_1, P_2)\right) + \left(\mathsf{out}^0_{|v}(P_1, \mathcal{A}^1) - \mathsf{out}^0_{|v}(P_1, P_2)\right)\right] \\
&\quad \geq \sum_{v \in \mathcal{V}} \mathsf{reach}_v \cdot \left[\Delta \cdot \epsilon - \left(|p_{left(v)} - q_{left(v)}| + |p_{right(v)} - q_{right(v)}|\right)\right]
\end{aligned}$$

$$\geq \sum_{v \in \mathcal{V}} \mathsf{reach}_v \cdot \Delta \cdot \epsilon - \sum_{v \in \mathcal{V}} \left(|p_{left(v)} - q_{left(v)}| + |p_{right(v)} - q_{right(v)}| \right) \cdot \mathsf{reach}_v$$
$$\geq \frac{1}{2} \cdot \Delta \cdot \epsilon - \mu'(n)$$

where the last inequality is true from Claim 6.5. We therefore conclude that the sum of the biases:

$$\left(\mathsf{out}^1(P_1, \mathcal{A}^1) - \frac{1}{2} \right) + \left(\mathsf{out}^0(P_1, \mathcal{A}^0) - \frac{1}{2} \right)$$

is non-negligible, as required. ■

Acknowledgement

We thank Gene Kopp and John Wiltshire-Gordon for helpful discussions.

References

[1] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In the 20*th STOC*, pages 1–10, 1988.

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

[3] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In the 20*th STOC*, pages 1–10, 1988.

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

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

[6] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *JACM*, 38(1):691–729, 1991.

[7] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In the 19*th STOC*, pages 218–229, 1987.

[8] S.D. Gordon, C. Hazay, J. Katz, and Y. Lindell. Complete fairness in secure two-party computation. *JACM*, 58(6):24, 2011.

[9] J. Kilian. A general completeness theorem for two-party games. In 23*rd STOC*, pages 553–560, 1991.

[10] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In the 21*st STOC*, pages 73–85, 1989.

[11] J. von Neumann. Various Techniques Used in Connection with Random Digits. *Journal of Research of the National Bureau of Standards*, 12:36–38, 1951.

[12] A. Yao. How to generate and exchange secrets (extended abstract). In the27*th FOCS*, pages 162–167, 1986.