Coin Flipping of *Any* Constant Bias Implies One-Way Functions

[Extended Abstract]*

Itay Berman†
School of Computer Science
Tel Aviv University
Israel
itayberm@post.tau.ac.il

Iftach Haitner†
School of Computer Science
Tel Aviv University
Israel
iftachh@cs.tau.ac.il

Aris Tentes†
School of Computer Science
New York University
USA
tentes@cs.nyu.edu

ABSTRACT

We show that the existence of a coin-flipping protocol safe against *any* non-trivial constant bias (e.g., .499) implies the existence of one-way functions. This improves upon a recent result of Haitner and Omri [FOCS '11], who proved this implication for protocols with bias $\frac{\sqrt{2}-1}{2} - o(1) \approx .207$. Unlike the result of Haitner and Omri, our result also holds for *w*eak coin-flipping protocols.

Categories and Subject Descriptors

F.0 [**Theory of Computation**]: General

General Terms

Theory

Keywords

coin-flipping protocols; one-way functions; minimal hardness assumptions

1. INTRODUCTION

A central focus of modern cryptography has been to investigate the weakest possible assumptions under which various cryptographic primitives exist. This direction of research has been quite fruitful, and minimal assumptions are known for a wide variety of primitives. In particular, it has been shown that one-way functions (i.e., easy to compute but hard to invert) imply pseudorandom generators, pseudorandom functions, symmetric-key encryption/message authentication, commitment schemes, and digital signatures [10, 11, 14, 13, 20, 21, 9, 22], where one-way functions were also shown to be implied by each of these primitives [15].

An important exception to the above successful characterization, however, is the case of coin-flipping (-tossing) protocols. A coin-flipping protocol [4] allows the honest parties to jointly flip an unbiased coin, where even a cheating (efficient) party cannot bias the outcome of the protocol by very much. Specifically, a coin-flipping protocol is δ-bias if no efficient cheating party can make the common output to be 1, or to be 0, with probability greater than $\frac{1}{2} + \delta$. While one-way functions are known to imply negligible-bias coin-flipping protocols [4, 20, 14], the other direction is less clear. Impagliazzo and Luby [15] showed that $\Theta\left(1/\sqrt{m}\right)$-bias coin-flipping protocols imply one-way functions, where m is the number of rounds in the protocol.[1] Recently, Maji, Prabhakaran, and Sahai [17] extended the above for $(\frac{1}{2} - 1/\operatorname{poly}(n))$-bias *constant-round* protocols, where n is the security parameter. And more recently, Haitner and Omri [12] have shown the above implication holds for $(\frac{\sqrt{2}-1}{2} - o(1) \approx 0.207)$-bias coin-flipping protocols (of arbitrary round complexity). No such implications were known for any other choice of parameters, and in particular for protocols with bias greater than $\frac{\sqrt{2}-1}{2}$ with super-constant round complexity.

1.1 Our Result

In this work, we make progress towards answering the question of whether coin-flipping protocols also imply one-way functions. We show that (even weak) coin-flipping protocols, safe against any non-trivial bias (e.g., 0.4999), do in fact imply such functions. We note that unlike [12], but like [15, 17], our result also applies to the so-called *weak coin-flipping protocols* (see Section 2.3 for the formal definition of strong and weak coin-flipping protocols). Specifically, we prove the following theorem.

THEOREM 1.1 (INFORMAL). *For any $c > 0$, the existence of a $(\frac{1}{2} - c)$-bias coin-flipping protocol (of any round complexity) implies the existence of one-way functions.*

Note that $\frac{1}{2}$-bias coin-flipping protocol requires no assumption (i.e., one party flips a coin and announces the result to the other party). So our result is tight as long as constant biases (i.e., independent of the security parameter) are concerned.

*Full version can be found at [3].

†Research supported by ISF grant 1076/11, the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11) and US-Israel BSF grant 2010196.

STOC '14, May 31 - June 03 2014, New York, NY, USA

ACM 978-1-4503-2710-7/14/05. . .$15.00.
http://dx.doi.org/10.1145/2591796.2591845.

[1] In the original paper, only $\frac{1}{2} + \operatorname{neg}(m)$ was stated, where the above term follows the proof technique hinted at the original paper and the result by Cleve and Impagliazzo [7].

To prove Theorem 1.1, we observe a connection between the success probability of the best (valid) attacks in a two-party game (i.e., chess) and the success of the biased-continuation attack of [12] in winning this game (see more in Section 1.3). The scope of this interesting connection seems to extend beyond the question in the focus of this paper, and we hope that it will find additional implications.

1.2 Related Results

As mentioned above, Impagliazzo and Luby [15] showed that negligible-bias coin-flipping protocols imply one-way functions. Maji et al. [17] proved the same for $(\frac{1}{2} - o(1))$-bias yet constant-round protocols. Finally, Haitner and Omri [12] showed that the above implication holds for $\frac{\sqrt{2}-1}{2} - o(1) \approx 0.207$)-bias (strong) coin-flipping protocols (of arbitrary round complexity). Results of weaker complexity implications are also known.

Zachos [23] has shown that non-trivial (i.e., $(\frac{1}{2} - o(1))$-bias), constant-round coin-flipping protocols imply that NP $\not\subseteq$ BPP, where Maji et al. [17] proved the same implication for $(\frac{1}{4} - o(1))$-bias coin-flipping protocols of arbitrary round complexity. Finally, it is well known that the existence of non-trivial coin-flipping protocols implies that PSPACE $\not\subseteq$ BPP. Apart from [12], all the above results extend to weak coin-flipping protocols. See Table 1 for a summary of the above results.

Implication	*Protocol type*	*Paper*
OWF	$(\frac{1}{2} - c)$-bias, for arbitrary $c > 0$	**This work**
OWF	$(\frac{\sqrt{2}-1}{2} - o(1))$-bias	[12][2]
OWF	$(\frac{1}{2} - o(1))$-bias, *constant round*	[17]
OWF	Negligible bias	[15]
NP $\not\subseteq$ BPP	$(\frac{1}{4} - o(1))$-bias	[17]
NP $\not\subseteq$ BPP	$(\frac{1}{2} - o(1))$-bias, *constant round*	[23]
PSPACE $\not\subseteq$ BPP	Non-trivial	Folklore

Table 1: Results summary.

Information theoretic coin-flipping protocols (i.e., whose security holds against all-powerful attackers) were shown to exist in the quantum world; Mochon [18] presented an ε-bias quantum weak coin-flipping protocol for any $\varepsilon > 0$. Chailloux and Kerenidis [5] presented a $\left(\frac{\sqrt{2}-1}{2} - \varepsilon\right)$-bias quantum strong coin-flipping protocol for any $\varepsilon > 0$ (this bias was shown in [16] to be tight). A key step in [5] is a reduction from strong to weak coin-flipping protocols, which holds also in the classical world.

A related line of work considers *fair* coin-flipping protocols. In this setting the honest party is required to always output a bit, whatever the other party does. In particular, a cheating party might bias the output coin just by aborting. We know that one-way functions imply fair $(1/\sqrt{m})$-bias coin-flipping protocols [1, 6], where m is the round complexity of the protocol, and this quantity is known to be tight for $O(m/\log m)$-round protocols with fully black-box reductions [8]. Oblivious transfer, on the other hand, implies fair $1/m$-bias protocols [19, 2] (this bias was shown in [6] to be tight).

1.3 Our Techniques

The following is a rather elaborate, high-level description of the ideas underlying our proof.

That the existence of a given (cryptographic) primitive implies the existence of one-way functions is typically proven by looking at the *primitive core function* — an efficiently computable function (not necessarily unique) whose inversion on uniformly chosen outputs implies breaking the security of the primitive.[3] For private-key encryption, for instance, a possible core function is the mapping from the inputs of the encryption algorithm (i.e., message, secret key, and randomness) into the ciphertexts. Assuming that one has defined such a core function for a given primitive, then, by definition, this function should be one-way. So it all boils down to finding, or proving the existence of, such a core function for the primitive under consideration. For a *non-interactive* primitive, finding such a core function is typically easy. In contrast, for an *interactive* primitive, finding such a core function, or functions is, at least in many settings, a much more involved task. The reason is that in order to break an interactive primitive, the attacker typically has to invert a given function on many different outputs, where these outputs are chosen *adaptively* by the attacker, after seeing the answers to the previous queries. As a result, it is very challenging to find a single function, or even finitely many functions, whose output distribution (on uniformly chosen input) matches the distribution of the attacker's queries.[4]

What seems as the only plausible candidate to serve as the core function of a coin-flipping protocol is its *transcript function*: the function that maps the parties' randomness into the resulting protocol transcript (i.e., the transcript produced by executing the protocol with this randomness). In order to bias the output of an m-round coin-flipping protocol by more than $O(\frac{1}{\sqrt{m}})$, a super-constant number of adaptive inversions of the transcript function seems necessary. Yet, we managed to prove that the transcript function is the core function of any (constant-bias) coin-flipping protocol. This is done by designing an adaptive attacker for any such protocol, whose query distribution is "not too far" from the output distribution of the transcript function (when invoked on uniform inputs). Since our attacker, described below, is not only adaptive, but also defined in a recursive manner, proving it possesses the aforementioned property is one of the major challenges we had to deal with.

In what follows, we give a high-level overview of our attacker that ignores computational issues (i.e., assumes it has

[1] Only holds for *strong* coin-flipping protocols.

[3] For the sake of this informal discussion, inverting a function on a given value means returning a *uniformly* chosen preimage of this value.

[4] If the attacker makes *constant* number of queries, one can overcome the above difficulty by defining a set of core functions $f_1, \ldots, f_k$, where f_1 is the function defined by the primitive, f_2 is the function defined by the attacker after making the first inversion call, and so on. Since the evaluation time of f_{i+1} is polynomial in the evaluation time of f_i (since evaluating f_{i+1} requires a call to an inverter of f_i), this approach fails miserably for attackers of super-constant query complexity.

a perfect inverter for any function). We then explain how to adjust this attacker to work with the inverter of the protocol's transcript function.

1.3.1 Optimal Valid Attacks and The Biased-Continuation Attack

The crux of our approach lies in an interesting connection between the optimal attack on a coin-flipping protocol and the more feasible *recursive biased-continuation* attack. The latter attack recursively applies the biased-continuation attack used by Haitner and Omri [12] to achieve their constant-bias attack (called there, the *random-continuation* attack) and is the basis of our efficient attack (assuming one-way functions do not exist) on coin-flipping protocols.

Let $\Pi = (\mathsf{A}, \mathsf{B})$ be a coin-flipping protocol (i.e., the common output of the honest parties is a uniformly chosen bit). In this discussion we restrict ourselves to analyzing attacks that when carried out by the left-hand side party, i.e., A, are used to bias the outcome towards one, and when carried out by the right-hand side party, i.e., B, are used to bias the outcome towards zero. Analogous statements hold for opposite attacks (i.e., attacks carried out by A and used to bias towards zero, and attacks carried out by B and used to bias towards one). The optimal valid attacker $\mathcal{A}$ carries out the *best* attack A can employ (using unbounded power) to bias the protocol towards *one*, while sending *valid* messages — ones that could have been sent by the honest party. The optimal valid attacker $\mathcal{B}$ carries out the best attack B can employ to bias the protocol towards *zero* is analogously defined. Since coin-flipping protocol is a zero-sum game, for any such protocol the expected outcome of $(\mathcal{A}, \mathcal{B})$ is either zero or one. As a first step, we give a lower bound on the success probability of the recursive biased-continuation attack carried out by the party winning the aforementioned zero-sum game. As this lower bound might not be sufficient for our goal (it might be less that constant) — and this is a crucial point in the description below — our analysis takes additional steps to give an arbitrarily-close-to-one lower bound on the success probability of the recursive biased-continuation attack carried out by *some* party, which may or may not be the same party winning the zero-sum game.[5]

Assume that $\mathcal{A}$ is the winning party when playing against $\mathcal{B}$. Since $\mathcal{A}$ sends only valid messages, it follows that the expected outcome of $(\mathsf{A}, \mathcal{B})$, i.e., honest A against the optimal attacker for B, is larger than zero (since A might send the optimal messages "by mistake"). Let $\mathsf{OPT}_{\mathsf{A}}(\Pi)$ be the expected outcome of the protocol $(\mathcal{A}, \mathsf{B})$ and let $\mathsf{OPT}_{\mathsf{B}}(\Pi)$ be 1 minus the expected outcome of the protocol $(\mathsf{A}, \mathcal{B})$. The above observation yields that $\mathsf{OPT}_{\mathsf{A}}(\Pi) = 1$, while $\mathsf{OPT}_{\mathsf{B}}(\Pi) = 1 - \alpha < 1$. This gives rise to the following question: *what gives $\mathcal{A}$ an advantage over $\mathcal{B}$?*

We show that if $\mathsf{OPT}_{\mathsf{B}}(\Pi) = 1 - \alpha$, then there exists an α-dense set $\mathcal{S}^{\mathsf{A}}$ of 1-transcripts, full transcripts in which the parties' common output is 1,[6] that are "dominated by A". The A-dominated set has an important property — its density is "immune" to any action B might take, even if B is employing its optimal attack; specifically, the following holds:

$$\Pr_{\langle \mathsf{A},\mathsf{B}\rangle}\left[\mathcal{S}^{\mathsf{A}}\right] = \Pr_{\langle \mathsf{A},\mathcal{B}\rangle}\left[\mathcal{S}^{\mathsf{A}}\right] = \alpha, \tag{1}$$

where $\langle \Pi' \rangle$ samples a random full transcript of protocol Π'. It is easy to be convinced that the above holds in case A controls the root of the tree and has a 1-transcript as a direct descendant; see Figure 1 for a concrete example. The proof of the general case can be found in [3]. Since the A-dominated set is B-immune, a possible attack for $\mathcal{A}$ is to go towards this set. Hence, what seems like a feasible adversarial attack for A is to mimic $\mathcal{A}$'s attack by hitting the A-dominated set with high probability. It turns out that the biased-continuation attack of [12] does exactly that.

The biased-continuation attacker $\mathsf{A}^{(1)}$, taking the role of A in Π and trying to bias the output of Π towards one, is defined as follows: given that the partial transcript is trans, algorithm $\mathsf{A}^{(1)}$ samples a pair of random coins $(r_{\mathsf{A}}, r_{\mathsf{B}})$ that is consistent with trans and leads to a 1-transcript, and then acts as the honest A on the random coins r_{A}, given the transcript trans. In other words, $\mathsf{A}^{(1)}$ takes the first step of a random continuation of (A, B) leading to a 1-transcript. (The attacker $\mathsf{B}^{(1)}$, taking the role of B and trying to bias the outcome towards zero, is analogously defined.) Haitner and Omri [12] showed that for any coin-flipping protocol, if either A or B carries out the biased-continuation attack towards one, the outcome of the protocol will be biased towards one by $\frac{\sqrt{2}-1}{2}$ (when interacting with the honest party).[7] Our basic attack employs the above biased-continuation attack recursively. Specifically, for $i > 1$ we consider the attacker $\mathsf{A}^{(i)}$ that takes the first step of a random continuation of $(\mathsf{A}^{(i-1)}, \mathsf{B})$ leading to a 1-transcript, letting $\mathsf{A}^{(0)} \equiv \mathsf{A}$. The attacker $\mathsf{B}^{(i)}$ is analogously defined. Our analysis takes a different route from that of [12], whose approach is only applicable for handling bias up to $\frac{\sqrt{2}-1}{2}$ and cannot be applied to weak coin-flipping protocols.[8] Instead, we analyze the probability of the biased-continuation attacker to hit the dominated set we introduced above.

Let trans be a 1-transcript of Π in which all messages are sent by A. Since $\mathsf{A}^{(1)}$ picks a random 1-transcript, and B cannot force $\mathsf{A}^{(1)}$ to diverge from this transcript, the probability to produce trans under an execution of $(\mathsf{A}^{(1)}, \mathsf{B})$ is *doubled* with respect to this probability under an execution of (A, B) (assuming the expected outcome of (A, B) is 1/2). The above property, that B cannot force $\mathsf{A}^{(1)}$ to diverge from a transcript, is in fact the B-immune property of the

[5] That the identity of the winner in $(\mathcal{A}, \mathcal{B})$ cannot be determined by the recursive biased-continuation attack is crucial. Since we show that the latter attack can be efficiently approximated assuming one-way functions do not exist, the consequences of giving up this information would be profound. It would mean that we can estimate the optimal attack (which is implemented in PSPACE) using only the assumption that one-way functions do not exist.

[6] Throughout, we assume without loss of generality that the protocol's transcripts determines the common output of the parties.

[7] They show that the same holds for the analogous attackers carrying out the biased-continuation attack towards zero.

[8] A key step in the analysis of Haitner and Omri [12] is to consider the "all-cheating protocol" $(\mathsf{A}^{(1),1}, \mathsf{B}^{(1),1})$, where $\mathsf{A}^{(1),1}$ plays against $\mathsf{B}^{(1),1}$ and they both carry out the biased-continuation attack trying to bias the outcome towards one. Since, and this is easy to versify, the expected outcome of $(\mathsf{A}^{(1),1}, \mathsf{B}^{(1),1})$ is one, using symmetry one can show that the expected outcome of either $(\mathsf{A}^{(1),1}, \mathsf{B})$ or $(\mathsf{A}, \mathsf{B}^{(1),1})$ is at least $\frac{1}{\sqrt{2}}$, yielding a bias of $\frac{1}{\sqrt{2}} - \frac{1}{2}$. As mentioned in [12], symmetry cannot be used to prove a bias larger than $\frac{1}{\sqrt{2}} - \frac{1}{2}$.

A-dominated set. A key point we make is to generalize the above argument to show that for the α-dense A-dominated set $\mathcal{S}^{\mathsf{A}}$ (exists assuming that $\mathsf{OPT}_{\mathsf{B}}(\Pi) = 1-\alpha < 1$), it holds that:

$$\Pr_{\langle \mathsf{A}^{(1)},\mathsf{B}\rangle}\left[\mathcal{S}^{\mathsf{A}}\right] \geq \frac{\alpha}{\mathsf{val}(\Pi)}, \tag{2}$$

where $\mathsf{val}(\Pi')$ is the expected outcome of Π'. Namely, in $(\mathsf{A}^{(1)},\mathsf{B})$ the probability of hitting the set $\mathcal{S}^{\mathsf{A}}$ of 1-transcripts is larger by a factor of at least $\frac{1}{\mathsf{val}(\Pi)}$ than the probability of hitting this set in the original protocol Π. Again, it is easy to be convinced that the above holds in case A controls the root of the tree and has a 1-transcript as a direct descendant; see Figure 1 for a concrete example. The proof of the general case can be found in [3].

Consider now the protocol $(\mathsf{A}^{(1)},\mathsf{B})$. In this protocol, the probability of hitting the set $\mathcal{S}^{\mathsf{A}}$ is at least $\frac{\alpha}{\mathsf{val}(\Pi)}$, and clearly the set $\mathcal{S}^{\mathsf{A}}$ remains B-immune. Hence, we can apply Equation (2) again, to deduce that

$$\begin{aligned}\Pr_{\langle \mathsf{A}^{(2)},\mathsf{B}\rangle}\left[\mathcal{S}^{\mathsf{A}}\right] = \Pr_{\langle (\mathsf{A}^{(1)})^{(1)},\mathsf{B}\rangle}\left[\mathcal{S}^{\mathsf{A}}\right] &\geq \frac{\Pr_{\langle \mathsf{A}^{(1)},\mathsf{B}\rangle}\left[\mathcal{S}^{\mathsf{A}}\right]}{\mathsf{val}(\mathsf{A}^{(1)},\mathsf{B})} \\ &\geq \frac{\alpha}{\mathsf{val}(\Pi)\cdot\mathsf{val}(\mathsf{A}^{(1)},\mathsf{B})}.\end{aligned} \tag{3}$$

Continuing it for κ iterations yields that

$$\mathsf{val}(\mathsf{A}^{(\kappa)},\mathsf{B}) \geq \Pr_{\langle \mathsf{A}^{(\kappa)},\mathsf{B}\rangle}\left[\mathcal{S}^{\mathsf{A}}\right] \geq \frac{\alpha}{\prod_{i=0}^{\kappa-1}\mathsf{val}(\mathsf{A}^{(i)},\mathsf{B})}. \tag{4}$$

So, modulo some cheating,[9] it seems that we are in good shape. Taking, for example, $\kappa = \log(\frac{1}{\alpha})/\log(\frac{1}{0.9})$, Equation (4) yields that $\mathsf{val}(\mathsf{A}^{(\kappa)},\mathsf{B}) > 0.9$. Namely, if we assume that $\mathcal{A}$ has an advantage over $\mathcal{B}$, then by recursively applying the biased-continuation attack for A enough times, we arbitrarily bias the expected output of the protocol towards one. Unfortunately, if this advantage (i.e., $\alpha = (1-\mathsf{OPT}_{\mathsf{B}}(\Pi))$) is very small, which is the case in typical examples, the number of recursions required might be linear in the protocol depth (or even larger). Given the recursive nature of the above attack, the running time of the described attacker is *exponential*. To overcome this obstacle, we consider not only the dominated set, but additional sets that are "close to" being dominated. Informally speaking, a 1-transcript belongs to the A-dominated set if it can be generated by an execution of $(\mathcal{A},\mathsf{B})$. In other words, the probability, over B's coins, that a transcript generated by a random execution of $(\mathcal{A},\mathsf{B})$ belongs to the A-dominated set is one. We define a set of 1-transcripts (that does not belong to the A-dominated set) to be "close to" A-dominated if there is an (unbounded) attacker $\widehat{\mathcal{A}}$, such that the probability, over B's coins, that a transcript generated by a random execution of $(\widehat{\mathcal{A}},\mathsf{B})$ belongs to the set is close to one. These sets are formally defined via the notion of conditional protocols, discussed next.

[9] The actual argument is somewhat more complicated than the one given above. To ensure the above argument holds we need to consider measures over the 1-transcripts (and not sets). In addition, while (the measure variant of) Equation (3) is correct, deriving it from Equation (2) takes some additional steps.

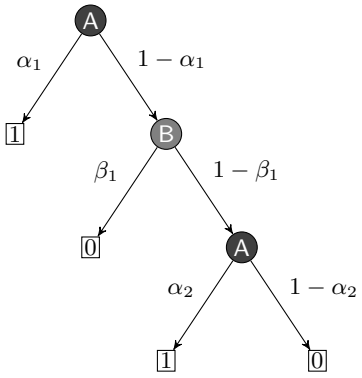

Figure 1: Coin-flipping protocol Π. The label of an internal node (i.e., partial transcript) denotes the name of the party controlling it (i.e., the party that sends the next message given this partial transcript), and that of a leaf (i.e., full transcript) denotes its value — the parties' common output once reaching this leaf. Finally, the label on an edge leaving a node u to node u' denotes the probability that a random execution of Π visits u' once in u.
Note that $\mathsf{OPT}_{\mathsf{A}}(\Pi) = 1$ and $\mathsf{OPT}_{\mathsf{B}}(\Pi) = 1-\alpha_1$. The A-dominated set $\mathcal{S}^{\mathsf{A}}$ in this case consists of the single 1-leaf to the left of the root. The conditional protocol Π' is the protocol rooted in the node to the right of the root (of Π), and the B'-dominated set $\mathcal{S}^{\mathsf{B}}$ consists of the single 0-leaf to the left of the root of Π'.

1.3.1.1 *Conditional Protocols.*

Let $\Pi = (\mathsf{A},\mathsf{B})$ be a coin-flipping protocol in which there exists an A-dominated set $\mathcal{S}^{\mathsf{A}}$ of density $\alpha > 0$. Consider the "conditional" protocol $\Pi' = (\mathsf{A}',\mathsf{B}')$, resulting from conditioning on not hitting the set $\mathcal{S}^{\mathsf{A}}$. Namely, the message distribution of Π' is that induced by a random execution of Π that does not generate transcripts in $\mathcal{S}^{\mathsf{A}}$. See Figure 1 for a concrete example. We note that the protocol Π' might not be efficiently computable (even if Π is), but this does not bother us, since we only use it as a thought experiment.

We have effectively removed all the 1-transcripts dominated by A (the set $\mathcal{S}^{\mathsf{A}}$ must contain all such transcripts; otherwise $\mathsf{OPT}_{\mathsf{B}}(\Pi)$ would be smaller than $1-\alpha$). Thus, the expected outcome of $(\mathcal{A}',\mathcal{B}')$ is zero. Therefore, $\mathsf{OPT}_{\mathsf{B}'}(\Pi') = 1$ and $\mathsf{OPT}_{\mathsf{A}'}(\Pi') = 1-\beta < 1$. It follows from this crucial observation that there exists a B'-dominated $\mathcal{S}^{\mathsf{B}}$ of density β, over the 0-transcripts of Π'. Applying a similar argument to that used for Equation (4) yields that for large enough κ, the biased-continuation attacker $\mathsf{B}'^{(\kappa)}$, playing the role of B', succeeds in biasing the outcome of Π' toward zero, where κ is proportional to $\log(\frac{1}{\beta})$. Moreover, if α is small, the above yields that $\mathsf{B}^{(\kappa)}$ is doing almost equally well in the original protocol Π. If β is also small, we can now consider the conditional protocol Π'', obtained by conditioning Π' on not hitting the B'-dominated set, and so on.

By iterating the above process enough times, the A-dominated sets cover all the 1-transcripts, and the B-dominated sets cover all the 0-transcripts.[10] Assume that in the above iterated process, the density of the A-dominated sets is the first to go beyond $\varepsilon > 0$. It can be shown — and

[10] When considering measures and not sets, as done in the actual proof, this covering property is not trivial.

this a key technical contribution of this paper — that it is almost as good as if the density of the *initial* set $\mathcal{S}^{\mathsf{A}}$ was ε.[11] We conclude that for any $\varepsilon > 0$, there exists a constant κ such that $\mathsf{val}(\mathsf{A}^{(\kappa)}, \mathsf{B}) > 1 - \varepsilon$.[12]

1.3.2 Using the Transcript Inverter

We have seen above that for any constant ε, by recursively applying the biased-continuation attack for constantly many times, we get an attack that biases the outcome of the protocol by $\frac{1}{2} - \varepsilon$. The next thing is to implement the above attack *efficiently*, under the assumption that one-way functions do not exist. Given a partial transcript u of protocol Π, we wish to return a uniformly chosen full transcript of Π that is consistent with u and the common outcome it induces is one. Biased continuation can be reduced to the task of finding *honest continuation*: returning a uniformly chosen full transcript of Π that is consistent with u. Assuming honest continuations can be done for the protocol, biased-continuation can also be done by calling the honest continuation many times, until transcript whose output is one is obtained. The latter can be done efficiently, as long as the value of the partial transcript u — the expected outcome of the protocol conditioned on u, is not too low. (If it is too low, too much time might pass before a full transcript leading to one is obtained.) Ignoring this low value problem, and noting that honest continuation of a protocol can be reduced to inverting the protocol's transcript function, all we need to do to implement $\mathsf{A}^{(i)}$ is to invert the transcript functions of the protocols $(\mathsf{A}, \mathsf{B}), (\mathsf{A}^{(1)}, \mathsf{B}), \ldots, (\mathsf{A}^{(i-1)}, \mathsf{B})$. Furthermore, noting that the attackers $\mathsf{A}^{(1)}, \ldots, \mathsf{A}^{(i-1)}$ are *stateless*, it suffices to have the ability to invert *only* the transcript function of (A, B).

So attacking a coin-flipping protocol Π boils down to inverting the transcript function f_Π of Π, and making sure we are not doing that on low value transcripts. Assuming one-way functions do not exist, there exists an efficient inverter Inv for f_Π that is guaranteed to work well when invoked on random outputs of f_Π (i.e., when f_Π is invoked on the uniform distribution. Nothing is guaranteed for distributions far from uniform). By the above discussion, algorithm Inv implies an efficient approximation of $\mathsf{A}^{(i)}$, as long as the partial transcripts attacked by $\mathsf{A}^{(i)}$ are neither *low-value* nor *unbalanced* (by low-value transcript we mean that the expected outcome of the protocol conditioned on the transcript is low; by unbalanced transcript we mean that its density with respect to $(\mathsf{A}^{(i)}, \mathsf{B})$ is not too far from its density with respect to (A, B)). Unlike [12], we failed to prove (and we believe that it is untrue) that the queries of $\mathsf{A}^{(i)}$ obey these two conditions with sufficiently high probability, and thus we cannot simply argue that $\mathsf{A}^{(i)}$ has an efficient approximation, assuming one-way functions do not exist. Fortunately, we managed to prove the above for the "pruned" variant of $\mathsf{A}^{(i)}$, defined below.

[11] More accurately, let $\widetilde{\mathcal{S}}^{\mathsf{A}}$ be the union of these 1-transcript sets and let $\widetilde{\alpha}$ be the density of $\widetilde{\mathcal{S}}^{\mathsf{A}}$ in Π. Then $\mathsf{val}(\mathsf{A}^{(\kappa)}, \mathsf{B}) \geq \Pr_{\langle \mathsf{A}^{(\kappa)}, \mathsf{B} \rangle}\left[\widetilde{\mathcal{S}}^{\mathsf{A}}\right] \geq \frac{\widetilde{\alpha}}{\prod_{i=0}^{\kappa-1} \mathsf{val}(\mathsf{A}^{(i)}, \mathsf{B})}$.

[12] The assumption that the density of the A-dominated sets is the first to go beyond $\varepsilon > 0$ is independent of the assumption that $\mathcal{A}$ wins in the zero-sum game $(\mathcal{A}, \mathcal{B})$. Specifically, the fact that $\mathsf{A}^{(\kappa)}$ succeeds in biasing the protocol does not guarantee that $\mathcal{A}$ is the winner of $(\mathcal{A}, \mathcal{B})$.

1.3.2.1 Unbalanced and low value transcripts.

Before defining our final attacker, we relate the problem of unbalanced transcripts to that of low-value transcripts. We say that a (partial) transcript u is γ-*unbalanced*, if the probability that u is visited with respect to a random execution of $(\mathsf{A}^{(1)}, \mathsf{B})$, is at least γ times larger than with respect to a random execution of (A, B). Furthermore, we say that a (partial) transcript u is δ-*small*, if the expected outcome of (A, B), conditioned on visiting u, is at most δ. We prove (a variant of) the following statement. For any $\delta > 0$ and $\gamma > 1$, there exists c that depends on δ, such that

$$\Pr_{\ell \leftarrow \langle \mathsf{A}^{(1)}, \mathsf{B} \rangle}[\ell \text{ has a } \gamma\text{-}\mathsf{unbalanced} \text{ prefix but no } \delta\text{-}\mathsf{small} \text{ prefix}] \leq \frac{1}{\gamma^c}. \quad (5)$$

Namely, as long as $(\mathsf{A}^{(1)}, \mathsf{B})$ does not visit low-value transcripts, it is only at low risk to significantly deviate (in a multiplicative sense) from the distribution induced by (A, B). Equation (5) naturally extends to recursive biased-continuation attacks (i.e., $\mathsf{A}^{(1)}$, for $i > 1$). It also has an equivalent form for the attacker $\mathsf{B}^{(1)}$, trying to bias the protocol towards zero, with respect to δ-high transcripts — the expected outcome of Π, conditioned on visiting the transcript, is at least $1 - \delta$.

1.3.2.2 The pruning attacker.

At last we are ready to define our final attacker. To this end, for protocol $\Pi = (\mathsf{A}, \mathsf{B})$ we define its δ-*pruned variant* $\Pi_\delta = (\mathsf{A}_\delta, \mathsf{B}_\delta)$, where $\delta \in (0, \frac{1}{2})$, as follows. As long as the execution does not visit a δ-low or δ-high transcripts, the parties act as in Π. Once a δ-low transcript is visited, only the party B sends messages, and it does so according to the distribution induced by Π. If a δ-high transcript is visited (and has no δ-low prefix), only the party A sends messages, and again it does so according to the distribution induced by Π.

Since the transcript distribution induced by Π_δ is the same as of Π, protocol Π_δ is also a coin-flipping protocol. We also note that Π_δ can be implemented efficiently assuming one-way functions do not exist (simply use the inverter of Π's transcript function to estimate the value of a given transcript). Finally, by Equation (5), $\mathsf{A}_\delta^{(i)}$ (i.e., recursive biased-continuation attacks for Π_δ) can be efficiently implemented, since there are *no* low-value transcripts where A needs to send the next message. (Similarly, $\mathsf{B}_\delta^{(i)}$ can be efficiently implemented since there are no high-value transcripts where B needs to send the next message.)

It follows that for any constant $\varepsilon > 0$, there exists constant κ such that either the expected outcome of $(\mathsf{A}_\delta^{(\kappa)}, \mathsf{B}_\delta)$ is a least $1 - \varepsilon$, or the expected outcome of $(\mathsf{A}_\delta, \mathsf{B}_\delta^{(\kappa)})$ is at most ε. Assume for concreteness that it is the former case. We define our pruning attacker $\mathsf{A}^{(\kappa,\delta)}$ as follows. When playing against B, the attacker $\mathsf{A}^{(\kappa,\delta)}$ acts like $\mathsf{A}_\delta^{(\kappa)}$ would when playing against B_δ. Namely, the attacker pretends that it is in the δ-pruned protocol Π_δ. But once a low or high value transcript is reached, $\mathsf{A}^{(\kappa,\delta)}$ acts *honestly* in the rest of the execution (like A would).

It follows that until a low or high value transcript has been reached for the first time, the distribution of $(\mathsf{A}^{(\kappa,\delta)}, \mathsf{B})$ is the same as that of $(\mathsf{A}_\delta^{(\kappa)}, \mathsf{B}_\delta)$. Once a δ-low transcript is reached, the expected outcome of both $(\mathsf{A}^{(\kappa,\delta)}, \mathsf{B})$ and

$(\mathsf{A}_\delta^{(\kappa)}, \mathsf{B}_\delta)$ is δ, but when a δ-high transcript is reached, the expected outcome of $(\mathsf{A}^{(\kappa,\delta)}, \mathsf{B})$ is $(1-\delta)$ (since it plays like A would), where the expected outcome of $(\mathsf{A}_\delta^{(\kappa)}, \mathsf{B}_\delta)$ is at most one. All in all, the expected outcome of $(\mathsf{A}^{(\kappa,\delta)}, \mathsf{B})$ is δ-close to that of $(\mathsf{A}_\delta^{(\kappa)}, \mathsf{B}_\delta)$, and thus the expected outcome of $(\mathsf{A}^{(\kappa,\delta)}, \mathsf{B})$ is at least $1-\varepsilon-\delta$. Since ε and δ are arbitrary constants, we have established an efficient attacker to bias the outcome of Π by a value that is an arbitrary constant close to one.

Paper Organization

General notations and definitions used throughout the paper are given in Section 2. Our ideal attacker to bias any coin-flipping protocol (i.e., the recursive biased-continuation attacker, which has access to a perfect sampler) is presented in Section 3, where in Section 4 we present the property of the above attacker that is useful when the sampler is implemented using one-way function inverter. In Section 4.3 we conclude with the statement of our main theorem.

2. PRELIMINARIES

2.1 Notations

We use calligraphic letters to denote sets, uppercase for random variables and functions, lowercase for values, boldface for vectors, and sans-serif (e.g., A) for algorithms (i.e., Turing Machines). All logarithms considered here are in base two, where $\circ$ denotes string concatenation. Let $\mathbb{N}$ denote the set of natural numbers, where 0 is considered as a natural number, i.e., $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$. For $n \in \mathbb{N}$, if n is positive let $[n] = \{1, \cdots, n\}$, where $[0] = \emptyset$.For a non-empty string $t \in \{0,1\}^*$ and $i \in [|t|]$, let t_i be the i'th bit of t, and for $i, j \in [|t|]$ such that $i < j$, let $t_{i,\ldots,j} = t_i \circ t_{i+1} \circ \ldots \circ t_j$. We let poly denote the set all polynomials and let PPTM denote a probabilistic algorithm that runs in *strictly* polynomial time. Give a PPTM algorithm A we let $\mathsf{A}(u; r)$ be an execution of A on input u given randomness r. A function $\nu\colon \mathbb{N} \mapsto [0,1]$ is *negligible*, denoted $\nu(n) = \text{neg}(n)$, if $\nu(n) < 1/p(n)$ for every $p \in$ poly and large enough n.

Given a random variable X, we write $x \leftarrow X$ to indicate that x is selected according to X. Similarly, given a finite set $\mathcal{S}$, we let $s \leftarrow \mathcal{S}$ denote that s is selected according to the uniform distribution on $\mathcal{S}$. We adopt the convention that when the same random variable occurs several times in an expression, all occurrences refer to a single sample. For example, $\Pr[f(X) = X]$ is defined to be the probability that when $x \leftarrow X$, we have $f(x) = x$. We write U_n to denote the random variable distributed uniformly over $\{0,1\}^n$. The support of a distribution D over a finite set $\mathcal{U}$, denoted $\text{Supp}(D)$, is defined as $\{u \in \mathcal{U} : D(u) > 0\}$. The *statistical distance* of two distributions P and Q over a finite set $\mathcal{U}$, denoted as $\text{SD}(P, Q)$, is defined as $\max_{\mathcal{S}\subseteq\mathcal{U}} |P(\mathcal{S}) - Q(\mathcal{S})| = \frac{1}{2}\sum_{u\in\mathcal{U}} |P(u) - Q(u)|$.

2.2 Two-Party Protocols

The following discussion is restricted to no-private input (there might only be a common input, e.g., the security parameter), possibly randomized, two-party protocols, where each message consists of a *single* bit. We do not assume, however, that the parties play in turns (i.e., the same party might send two consecutive messages), but only that the protocol's transcript uniquely determines which party is playing next (i.e., the protocol is well defined). In an m-round protocol, the parties interact for exactly m rounds. The tuple of the messages sent so far in any partial execution of a protocol is called the *(communication) transcript* of this execution.

A protocol Π is defined by a pair of interactive Turing Machines (A, B), which interact given a common input 1^n, where n is called the security parameter. For a fixed security parameter we denote the induced protocol as $\Pi(1^n) = (\mathsf{A}, \mathsf{B})(1^n)$, however, when clear from the context the security parameter is omitted.

If A, B are deterministic, then by $\text{trans}(\mathsf{A}, \mathsf{B})$, we denote the uniquely defined transcript, namely the bits sent by both parties in the order of appearance, when these parties run the protocol (for a fixed security parameter).

2.2.1 Binary Trees

DEFINITION 2.1 (BINARY TREES). *For $m \in \mathbb{N}$, let $\mathcal{T}^m$ be the complete directed binary tree of height m. We naturally identify the vertices of $\mathcal{T}^m$ with binary strings: the root is denoted by the empty string λ, and the the left-hand side and right-hand side children of a non-leaf node u, are denoted by $u0$ and $u1$ respectively.*

- *Let $\mathcal{V}(\mathcal{T}^m)$, $\mathcal{E}(\mathcal{T}^m)$, $\mathsf{root}(\mathcal{T}^m)$ and $\mathcal{L}(\mathcal{T}^m)$ denote the vertices, edges, root and leaves of $\mathcal{T}^m$ respectively.*
- *For $u \in \mathcal{V}(\mathcal{T}^m) \setminus \mathcal{L}(\mathcal{T}^m)$, let $\mathcal{T}^m_u$ be the subtree of $\mathcal{T}^m$ rooted at u.*
- *For $u \in \mathcal{V}(\mathcal{T}^m)$, let $\mathsf{desc}_m(u)$ [resp., $\overline{\mathsf{desc}}_m(u)$] be the* descendants *of u in $\mathcal{T}^m$ including u [resp., excluding u], and for $\mathcal{U} \subseteq \mathcal{V}(\mathcal{T}^m)$ let $\mathsf{desc}_m(\mathcal{U}) = \bigcup_{u\in\mathcal{U}} \mathsf{desc}_m(u)$ and $\overline{\mathsf{desc}}_m(\mathcal{U}) = \bigcup_{u\in\mathcal{U}} \overline{\mathsf{desc}}_m(u)$.*

When m is clear from the context, it is typically omitted from the above notation.

2.2.2 Protocol Trees

We naturally identify a (possibly partial) transcript of a m-round, single-bit message protocol with a rooted path in $\mathcal{T}^m$. That is, the transcript $t \in \{0,1\}^m$ is identified with the path $\lambda, t_1, t_{1,2}, \ldots, t$.

DEFINITION 2.2 (TREE REPRESENTATION OF A PROTOCOL). *We make use of the following definitions with respect to an m-round protocol $\Pi = (\mathsf{A}, \mathsf{B})$, and $\mathsf{C} \in \{\mathsf{A}, \mathsf{B}\}$.*

- *Let $\text{round}(\Pi) = m$, let $\mathcal{T}(\Pi) = \mathcal{T}^m$ and for $X \in \{\mathcal{V}, \mathcal{E}, \mathsf{root}, \mathcal{L}\}$ let $X(\Pi) = X(\mathcal{T}(\Pi))$.*
- *The* edge distribution *induced by a protocol Π, is the function $e_\Pi\colon \mathcal{E}(\Pi) \mapsto [0,1]$ defined as $e_\Pi(u, v)$ being the probability that the transcript of a random execution of Π visits v, conditioned that it visits u.*
- *For $u \in \mathcal{V}(\Pi)$, let $\mathsf{v}_\Pi(u) = e_\Pi(\lambda, u_1) \cdot e_\Pi(u_1, u_{1,2}) \cdot \ldots \cdot e_\Pi(u_{1,\ldots,|u|-1}, u)$, and let the* leaves distribution *induced by Π be the distribution $\langle\Pi\rangle$ over $\mathcal{L}(\Pi)$, defined by $\langle\Pi\rangle(u) = \mathsf{v}_\Pi(u)$.*
- *The party that sends the next message on transcript u, is said to* control *u, and we denote this party by $\mathsf{cntrl}_\Pi(u)$. Let $\mathcal{C}\mathsf{trl}^{\mathsf{C}}_\Pi = \{u \in \mathcal{V}(\Pi)\colon \mathsf{cntrl}_\Pi(u) = \mathsf{C}\}$.*

Note that every function $e\colon \mathcal{E}(\mathcal{T}^m) \mapsto [0,1]$ with $e(u,u0)+e(u,u1)=1$ for every $u \in \mathcal{V}(\mathcal{T}^m) \setminus \mathcal{L}(\mathcal{T}^m)$ with $\mathsf{v}(u)>0$, along with a controlling scheme (who is active in each node), defines a two party, m-round, single-bit message protocol (the resulting protocol might be inefficient). This observation allows us to consider the protocols induced by subtrees of $\mathcal{T}(\Pi)$.

The analysis done in Section 3 naturally gives rise to functions over binary trees, that do not corresponds to any two parties execution. We identify the "protocols" induced by such functions by the special symbol $\perp$. We let $\mathrm{E}_{\langle\perp\rangle}[f]=0$, for any real-value function f.

DEFINITION 2.3 (SUB-PROTOCOLS). *Let Π be a protocol and let $u \in \mathcal{V}(\Pi)$. Let $(\Pi)_u$ denotes the the protocol induced by the function e_Π on the subtree of $\mathcal{T}(\Pi)$ rooted at u, in case such protocol exists,*[13] *and let $(\Pi)_u = \perp$, otherwise.*

When convenient, we remove the parentheses from notation, and simply write Π_u. Two sub-protocols of interest are Π_0 and Π_1, induced by e_Π and the trees rooted at the left-hand side and right-hand side descendants of $\mathsf{root}(\mathcal{T})$.

2.2.3 Tree Value

DEFINITION 2.4 (TREE VALUE). *Let Π a two-party protocol, in which at the end of any of its executions the parties output the* same *real value. Let $\chi_\Pi\colon \mathcal{L}(\Pi) \mapsto \mathbb{R}$ be the* common output *function of Π, where $\chi_\Pi(\ell)$ being the* common *output of the parties in an execution ending in ℓ.*[14] *Let $\mathsf{val}(\Pi) = \mathrm{E}_{\langle\Pi\rangle}[\chi_\Pi]$, and for $x \in \mathbb{R}$ let $\mathcal{L}_x(\Pi) = \{\ell \in \mathcal{L}(\Pi)\colon \chi_\Pi(\ell)=x\}$.*

2.3 Coin-Flipping Protocols

In a coin-flipping protocol two parties interact and in the end they have a common output bit. Ideally, this bit should be random and no cheating party should be able to bias its outcome to neither direction (if the other party remains honest). For interactive, probabilistic algorithms A and B, and $x \in \{0,1\}^*$, let $\mathrm{out}(\mathsf{A},\mathsf{B})(x)$ denotes parties' output, on common input x.

DEFINITION 2.5 ((STRONG) COIN-FLIPPING). *A* PPT *protocol (A,B) is a δ-*bias coin-flipping protocol*, if the following holds.*

Correctness: $\Pr[\mathrm{out}(\mathsf{A},\mathsf{B})(1^n) = (0,0)] = \Pr[\mathrm{out}(\mathsf{A},\mathsf{B})(1^n)=(1,1)] = \frac{1}{2}$.

Security: $\Pr[\mathrm{out}(\mathsf{A}^*,\mathsf{B})(1^n) = (*,c)], \Pr[\mathrm{out}(\mathsf{A},\mathsf{B}^*)(1^n) = (c,*)] \leq \frac{1}{2}+\delta(n)$, *for any* PPTM*'s A^* and B^*, bit $c \in \{0,1\}$ and large enough n.*

Sometimes, e.g., if the parties have (a priori known) opposite preferences, an even weaker definition of coin-flipping protocols is of interest.

DEFINITION 2.6 (WEAK COIN-FLIPPING). *A* PPT *protocol (A,B) is a* weak *δ-*bias coin-flipping protocol*, if the following holds.*

Correctness: Same as in Definition 2.5.

Security: There exist bits $c_\mathsf{A} \neq c_\mathsf{B} \in \{0,1\}$ such that

$$\Pr[\mathrm{out}(\mathsf{A}^*,\mathsf{B})(1^n)=c_\mathsf{A}], \Pr[\mathrm{out}(\mathsf{A},\mathsf{B}^*)(1^n)=c_\mathsf{B}] \leq \frac{1}{2}+\delta(n)$$

for any PPTM*'s A^* and B^*, and large enough n.*

REMARK 2.7. *Our result still holds when replacing the value $\frac{1}{2}$ in the correctness requirement above, with any constant in $(0,1)$. It also holds for protocols in which, with some small probability, the parties are not in agreement regarding the protocol's outcome, or even might output values that are not bits.*

In the the rest of the paper we restrict our attention to m-round single-bit message coin-flipping protocols, where $m=m(n)$ is a function of the protocol's security parameter. Given such protocol $\Pi=(\mathsf{A},\mathsf{B})$, we assume that the common output of the protocol (i.e., the coin), is efficiently computable from a (full) transcript of the protocol. (It is easy to see that these assumptions are without loss of generality).

2.4 One-Way Functions and Distributional One-Way Functions

A one-way function (OWF) is an efficiently computable function whose inverse cannot be computed on average by any PPTM.

DEFINITION 2.8. *A polynomial-time computable function $f\colon \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ is* one-way*, if*

$$\Pr_{x\leftarrow\{0,1\}^n;y=f(x)}\left[\mathsf{A}(1^n,y) \in f^{-1}(y)\right] = \mathrm{neg}(n)$$

for any PPTM A.

A seemingly weaker definition is that of a distributional OWF. Such a function is easy to compute, but, roughly speaking, it is hard to compute uniformly random preimages of random images.

DEFINITION 2.9. *A polynomial-time computable $f\colon \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ is* distributional one-way*, if $\exists p \in \mathrm{poly}$ such that*

$$\mathrm{SD}\left((x,f(x))_{x\leftarrow\{0,1\}^n}, (\mathsf{A}(f(x)),f(x))_{x\leftarrow\{0,1\}^n}\right) \geq \frac{1}{p(n)}$$

for any PPTM A *and large enough n.*

Clearly, any one-way function is also a distributional one-way function. While the other implication is not necessarily always true, Impagliazzo and Luby [15] showed that the existence of distributional one-way functions implies that of (standard) one-way functions. In particular, [15] proved that if one-way functions do not exist, then any efficiently computable function has an inverter of the following form.

DEFINITION 2.10 (γ-INVERTER). *An algorithm* Inv *is an γ-*inverter *of $f\colon \mathcal{D} \mapsto \mathcal{R}$, if the following holds.*

$$\Pr_{x\leftarrow\mathcal{D};y=f(x)}\left[\mathrm{SD}\left((y,x')_{x'\leftarrow f^{-1}(y)}, (y,\mathsf{Inv}(y))\right) \geq \gamma\right] \leq \gamma.$$

LEMMA 2.11 ([15, LEMMA 1]). *Assume one-way functions do not exist, then for any polynomial-time computable function $f\colon \{0,1\}^n \mapsto \{0,1\}^{\ell(n)}$ and $p \in \mathrm{poly}$, there exists a* PPTM *algorithm* Inv *such that the following holds for infinitely many n's. On security parameter 1^n, algorithm* Inv *is a $1/p(n)$-inverter of f_n (i.e., f restricted to $\{0,1\}^n$).*

[13] Namely, the protocol Π_u, is the protocol Π conditioned on u being the transcript of the first $|u|$ rounds.

[14] Since condition on u, the random coins of the parties are in a product distribution, under the above assumption the common output is indeed a function of u.

Note that nothing is guaranteed when invoking a good inverter (i.e., a γ-inverter for some small γ) on an *arbitrary distribution*. Yet, the following lemma shows that if the distribution in consideration is "not too different" from the output distribution of f, then such good inverters are useful.

LEMMA 2.12. *Let f and g be two randomized functions over the same domain $\mathcal{D}$, and let $\{D_i\}_{i\in[k]}$ be a set of distributions over $\mathcal{D}$ such that for some $a \geq 0$ it holds that $\mathrm{E}_{d\leftarrow D_i}[\mathrm{SD}(f(d), g(d))] \leq a$ for every $i \in [k]$. Let A be a k-query oracle-aided algorithm that only makes queries in $\mathcal{D} \cup \{\perp\}$. For $i \in [k]$, let F_i be the probability distribution of the i'th query to f in a random execution of A^f, and let $Q = (Q_1, \ldots, Q_k)$ be the random variable of the queries of A^f in such a random execution (in case the i'th query was $\perp$, we also set its reply to $\perp$).*

Assume $\Pr_{(q_1,\ldots,q_k)\leftarrow Q}[\exists i \in [k] \colon q_i \neq \perp \wedge F_i(q_i) > \lambda \cdot D_i(q_i)] \leq b$ for some $\lambda, b \geq 0$, then $\mathrm{SD}\left(\mathsf{A}^f, \mathsf{A}^g\right) \leq b + ka\lambda$.

3. THE BIASED-CONTINUATION ATTACK

In this section we describe an attack to bias any (coin-flipping) protocol (in the following we typically omit the term "coin-flipping", since we only consider such protocols). The described attack, however, might be impossible to implement efficiently (even when assuming one-way functions do not exist). Specifically, we assume access to an ideal sampling algorithm to sample a *uniform* preimage of *any* output of the functions in consideration. Our actual attack tries to mimic the behaviour of this attack while being efficiently implemented (assuming one-way functions do not exist).

The following discussion is restricted to (coin-flipping) protocols whose parties always output the same bit as their common output, and this bit is determined by the protocol's transcript. In all protocols considered in this section, the messages are bits. In addition, the protocols in consideration have no inputs (neither private nor common), and in particular no security parameter is involved.

For concreteness, the attackers described below taking the left-hand side party of the protocol (i.e., A), are trying to bias the common output of the protocol towards one where the attackers taking the right-hand side party (i.e., B) are trying to bias the common output towards zero. All statements have analogue ones with respect to the opposite attack goals.

Let $\Pi = (\mathsf{A}, \mathsf{B})$ be a protocol. The *recursive biased-continuation attack* described below applies recursively the *biased-continuation attack* introduced by Haitner and Omri [12].[15] The biased-continuation attacker $\mathsf{A}_\Pi^{(1)}$ – playing the role of A – works as follows: in each of A's turns, $\mathsf{A}_\Pi^{(1)}$ picks a random continuation of Π, whose output it induces is equal one, and plays the current turn accordingly. The i'th biased-continuation attacker $\mathsf{A}_\Pi^{(i)}$, formally described below, uses the same strategy but the random continuation taken is of the protocol $(\mathsf{A}_\Pi^{(i-1)}, \mathsf{B})$.

Moving to the formal discussion, for a protocol $\Pi = (\mathsf{A}, \mathsf{B})$, let $\mathsf{BiasedCont}_\Pi$ be the following algorithm.

ALGORITHM 3.1 ($\mathsf{BiasedCont}_\Pi$).

Input: $u \in \mathcal{V}(\Pi) \setminus \mathcal{L}(\Pi)$ *and a bit* $b \in \{0, 1\}$

Operation:

1. *Choose* $\ell \leftarrow \langle \Pi \rangle$ *conditioned that*
   (a) $\ell \in \mathsf{desc}(u)$, *and*
   (b) $\chi_\Pi(\ell) = b$.[16]
2. *Return* $\ell_{|u|+1}$.

Let $\mathsf{A}_\Pi^{(0)} \equiv \mathsf{A}$, and for integer $i > 0$ define:

ALGORITHM 3.2 ($\mathsf{A}_\Pi^{(i)}$).

Oracle: $\mathsf{BiasedCont}_{(\mathsf{A}^{(i-1)},\mathsf{B})}$

Input: transcript $u \in \{0, 1\}^*$.

Operation:

1. *If* $u \in \mathcal{L}(\Pi)$, *output* $\chi_\Pi(u)$ *and halt.*
2. *Set* $\mathsf{msg} = \mathsf{BiasedCont}_{(\mathsf{A}_\Pi^{(i-1)},\mathsf{B})}(u, 1)$.
3. *Send* msg *to* B.
4. *If* $u' = u \circ \mathsf{msg} \in \mathcal{L}(\Pi)$, *output* $\chi_\Pi(u')$.[17]

Adversary $\mathsf{B}_\Pi^{(i)}$ attacking towards zero is analogously defined. Specifically, changing the call $\mathsf{BiasedCont}_{(\mathsf{A}_\Pi^{(i-1)},\mathsf{B})}(u, 1)$ in Algorithm 3.2 to $\mathsf{BiasedCont}_{(\mathsf{A},\mathsf{B}_\Pi^{(i-1)})}(u, 0)$.[18]

One can show that the more recursions $\mathsf{A}_\Pi^{(i)}$ and $\mathsf{B}_\Pi^{(i)}$ do, the closer their success probability to that of an all powerful adversary, who can either bias the outcome to zero or to one. The important point of the following theorem is that for any $\varepsilon > 0$ there exists a *global* constant $\kappa = \kappa(\varepsilon)$ (i.e., independent of the underlying protocol), for which either $\mathsf{A}_\Pi^{(\kappa)}$ or $\mathsf{B}_\Pi^{(\kappa)}$ succeeds in its attack with probability at least $1 - \varepsilon$. This fact gets crucial when trying to efficiently implement these adversaries, as each recursion call might induce a polynomial blowup in the running time of the adversary. Since κ is constant (for a constant ε), the recursive attacker is still efficient.

THEOREM 3.3 (MAIN THEOREM, IDEAL VERSION). *For every $\varepsilon \in (0, \frac{1}{2}]$ there exists an integer $\kappa = \kappa(\varepsilon) \geq 0$ such that for every protocol $\Pi = (\mathsf{A}, \mathsf{B})$, either $\mathsf{val}(\mathsf{A}_\Pi^{(\kappa)}, \mathsf{B}) > 1 - \varepsilon$ or $\mathsf{val}(\mathsf{A}, \mathsf{B}_\Pi^{(\kappa)}) < \varepsilon$.*

In the following we state a basic property of the recursive biased-continuation attacker, that is crucial in order to analyze our efficient attacker (assuming one-way functions do not exist) defined in the full version [3].

[15] Called the "random continuation attack" in [12].

[16] In case no such ℓ exists, the algorithm returns an arbitrary leaf in $\mathsf{desc}(u)$.

[17] For the mere purpose of biassing B's output, there is no need for $\mathsf{A}^{(i)}$ to output anything. Yet, doing that helps us to simplify our recursion definitions (specifically, we use the fact that in $(\mathsf{A}^{(i)}, \mathsf{B})$ the parties always have the same output).

[18] The subscript Π is added to the notation (i.e., $\mathsf{A}_\Pi^{(i)}$), since the biased-continuation attack for A depends not only on the definition of the party A, but also on the definition of B, the other party in the protocol.

4. EFFICIENT ATTACK

In this section we first state a basic property of biased-continuation attack, then discuss how this can be used to implement this attack efficiently, assuming OWFs do not exists and finally state our main theorem.

4.1 Visiting Unbalanced Nodes is Unlikely

We want to compare the leaves distribution of the protocol $\left(\mathsf{A}^{(1)},\mathsf{B}\right)$ to that of $\Pi=(\mathsf{A},\mathsf{B})$. Let u be a node that random execution of the former protocol reaches with probability much higher than that of the latter protocol (we call such node *unbalanced*). For every $i\in[|u|]$, such that $u_{1,\ldots,i}$ is controlled by A, $\mathsf{A}^{(1)}$ puts $\frac{\mathsf{val}(\Pi_{u_{1,\ldots,i+1}})}{\mathsf{val}(\Pi_{u_{1,\ldots,i}})}$ times extra weight on the path leading to u than what A puts on the same path. Thus, if none of the nodes on the path to u, which are controlled by A, has low value, u cannot be unbalanced by much.

The following key lemma formulates the above intuition, and shows that biased-continuation attacker does not biased the initial distribution of the attacked protocol by too much, unless it has previously visited a low value node.

DEFINITION 4.1. *For a protocol $\Pi=(\mathsf{A},\mathsf{B})$ and $\delta\in[0,1]$, let*

- $\mathcal{S}\mathsf{mall}^{\delta}_{\Pi}=\{u\in\mathcal{V}(\Pi)\setminus\mathcal{L}(\Pi):\mathsf{val}(\Pi_u)\leq\delta\}$, *and*
- $\mathcal{L}\mathsf{arge}^{\delta}_{\Pi}=\{u\in\mathcal{V}(\Pi)\setminus\mathcal{L}(\Pi):\mathsf{val}(\Pi_u)\geq 1-\delta\}$.

For $\mathsf{C}\in\{\mathsf{A},\mathsf{B}\}$, let $\mathcal{S}\mathsf{mall}^{\delta,\mathsf{C}}_{\Pi}=\mathcal{S}\mathsf{mall}^{\delta}_{\Pi_1}\cap\mathcal{C}\mathsf{trl}^{\mathsf{C}}_{\Pi}$ and similarly $\mathcal{L}\mathsf{arge}^{\delta,\mathsf{C}}_{\Pi}=\mathcal{L}\mathsf{arge}^{\delta}_{\Pi_1}\cap\mathcal{C}\mathsf{trl}^{\mathsf{C}}_{\Pi}$.[19]

DEFINITION 4.2. *(unbalanced nodes) For a protocol $\Pi=(\mathsf{A},\mathsf{B})$ and $\gamma\geq 1$, let $\mathsf{UnBal}^{\gamma}_{\Pi}=\left\{u\in\mathcal{V}(\Pi):\mathsf{v}_{(\mathsf{A}^{(1)},\mathsf{B})}(u)\geq\gamma\cdot\mathsf{v}_{(\mathsf{A},\mathsf{B})}(u)\right\}$, where $\mathsf{A}^{(1)}$ is as in Algorithm 3.2 and v as in Definition 2.2.*

LEMMA 4.3. *Let $\Pi=(\mathsf{A},\mathsf{B})$ be a protocol and let $\mathsf{A}^{(1)}$ be as in Algorithm 3.2. Then for every $\delta\in(0,\frac{1}{2}]$, there exists a constant $c=c(\delta)>0$ such that for every $\delta'\geq\delta$ and every $\gamma>1$.*

$$\Pr_{\langle\mathsf{A}^{(1)},\mathsf{B}\rangle}\left[\mathsf{desc}\left(\mathsf{UnBal}^{\gamma}_{\Pi}\setminus\overline{\mathsf{desc}}(\mathcal{S}\mathsf{mall}^{\delta',\mathsf{A}}_{\Pi})\right)\right]\leq\frac{2}{\gamma^c}.$$[20]

4.2 Efficiently Approximating Biased-Continuation Attack

For any protocol $\Pi=(\mathsf{A},\mathsf{B})$ let f_Π be defined over $1^*\times\{0,1\}^{\rho_\mathsf{A}(n)}\times\{0,1\}^{\rho_\mathsf{B}(n)}\times[m(n)]$ as follows

$$f_\Pi(1^n,r_\mathsf{A},r_\mathsf{B},i)=1^n,\mathrm{trans}((\mathsf{A}(r_\mathsf{A}),\mathsf{B}(r_\mathsf{B}))(1^n))_{1,\ldots,i}\quad(6)$$

Assuming that f_Π is not distributional one-way, and that in Π there are no low value nodes controlled by A, the biased-continuation attacker $\mathsf{A}^{(1)}$ can be efficiently approximated: when it is its turn, $\mathsf{A}^{(1)}$ calls some γ-inverter of f_Π (see Definition 2.10), until getting randomness for both parties that implies output 1. Lemma 4.3 shows that the inverter will not be called on unbalanced nodes, and thus this implementation is indeed an approximation of $\mathsf{A}^{(1)}$. (An approximation of $\mathsf{B}^{(1)}$ is defined analogously.)

In order, however, to attack a general protocol we have to modify the biased-continuation attack, in a way that it somehow ignores low value nodes. For a full description and analysis of this attack we refer the reader to the full version [3].

4.3 Main Theorem

After modifying the biased-continuation attack we can prove our main theorem stated below:

THEOREM 4.4 (RESTATEMENT OF THEOREM 1.1). *Assume one-way functions do not exist. Then for any* PPT *coin-flipping protocol $\Pi=(\mathsf{A},\mathsf{B})$ and $\varepsilon>0$, there exist* PPTM*'s $\mathcal{A}$ and $\mathcal{B}$ such that the following holds for infinitely many n's.*

1. $\mathsf{val}(\mathcal{A}(1),\mathsf{B})(1^n)\geq 1-\varepsilon$ *or* $\mathsf{val}(\mathsf{A},\mathcal{B}(0))(1^n)\leq\varepsilon$, *and*
2. $\mathsf{val}(\mathcal{A}(0),\mathsf{B})(1^n)\leq\varepsilon$ *or* $\mathsf{val}(\mathsf{A},\mathcal{B}(1))(1^n)\geq 1-\varepsilon$.[21]

5. ACKNOWLEDGMENTS

We are very grateful to Hemanta Maji, Yishay Mansour, Eran Omri and Alex Samorodnitsky for useful discussions.

References

[1] B. Averbuch, M. Blum, B. Chor, S. Goldwasser, and S. Micali. How to implement Bracha's $O(\log n)$ Byzantine agreement algorithm, 1985. Unpublishe manuscript.

[2] A. Beimel, E. Omri, and I. Orlov. Protocols for multiparty coin toss with dishonest majority. In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 538–557. Springer, 2010.

[3] I. Berman, I. Haitner, and A. Tentes. Coin flipping of any constant bias implies one-way functions. www.cs.tau.ac.il/~iftachh/papers/TightCF/TCoingFlip_F.p 2014. Manuscript.

[4] M. Blum. Coin flipping by telephone. pages 11–15, 1981.

[5] A. Chailloux and I. Kerenidis. Optimal quantum strong coin flipping. In *Proceedings of the 50th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 527–533, 2009.

[6] R. Cleve. Limits on the security of coin flips when half the processors are faulty. pages 364–369, 1986.

[7] R. Cleve and R. Impagliazzo. Martingales, collective coin flipping and discrete control processes (extended abstract), 1993. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.1797.

[19] Recall that $\mathcal{C}\mathsf{trl}^{\mathsf{C}}_{\Pi}$ denotes the nodes in $\mathcal{T}(\Pi)$ controlled by the party C.

[20] Recall that $\langle\Pi\rangle$, is the transcript induced by a random execution of Π, where $\mathsf{desc}(u)$ and $\overline{\mathsf{desc}}(u)$ are the descendants and the proper descendants of u as defined in Definition 2.1.

[21] The parameter of the attackers $\mathcal{A}$ and $\mathcal{B}$ indicate the desired output of the attacker.

[8] D. Dachman-Soled, Y. Lindell, M. Mahmoody, and T. Malkin. On the black-box complexity of optimally-fair coin tossing. volume 6597, pages 450–467, 2011.

[9] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 25–32, 1989.

[10] O. Goldreich, S. Goldwasser, and S. Micali. On the cryptographic applications of random functions. In *Advances in Cryptology – CRYPTO '84*, pages 276–288, 1984.

[11] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, pages 792–807, 1986.

[12] I. Haitner and E. Omri. Coin Flipping with Constant Bias Implies One-Way Functions. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 110–119, 2011.

[13] I. Haitner, M. Nguyen, S. J. Ong, O. Reingold, and S. Vadhan. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM Journal on Computing*, 39(3):1153–1218, 2009.

[14] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, pages 1364–1396, 1999.

[15] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 230–235, 1989.

[16] A. Y. Kitaev. Quantum coin-flipping. Presentation at the 6th workshop on quantum information processing (qip 2003), 2003.

[17] H. K. Maji, M. Prabhakaran, and A. Sahai. On the Computational Complexity of Coin Flipping. In *Proceedings of the 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 613–622, 2010.

[18] C. Mochon. Quantum weak coin flipping with arbitrarily small bias. arXiv:0711.4114, 2007.

[19] T. Moran, M. Naor, and G. Segev. An optimally fair coin toss. pages 1–18, 2009.

[20] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, pages 151–158, 1991.

[21] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*, pages 33–43, 1989.

[22] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 387–394, 1990.

[23] S. Zachos. Probabilistic quantifiers, adversaries, and complexity classes: An overview. In *Proceedings of the First Annual IEEE Conference on Computational Complexity.*