A Tight Lower Bound on Adaptively Secure Full-Information Coin Flip

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

Yonatan Karidi-Heller
School of Computer Science
Tel Aviv University, TAU
Tel Aviv, Israel
karidiheller@mail.tau.ac.il

***Abstract*—In a distributed *coin-flipping* protocol, Blum [ACM Transactions on Computer Systems '83], the parties try to output a common (close to) uniform bit, even when some adversarially chosen parties try to bias the common output. In an *adaptively secure full-information coin flip*, Ben-Or and Linial [FOCS '85], the parties communicate over a broadcast channel and a computationally unbounded adversary can choose which parties to corrupt *along* the protocol execution. Ben-Or and Linial proved that the n-party majority protocol is resilient to $O(\sqrt{n})$ corruptions (ignoring poly-logarithmic factors), and conjectured this is a tight upper bound for any n-party protocol (of any round complexity). Their conjecture was proved to be correct for *single-turn* (each party sends a single message) *single-bit* (a message is one bit) protocols Lichtenstein, Linial, and Saks [Combinatorica '89], *symmetric* protocols Goldwasser, Tauman Kalai, and Park [ICALP '15], and recently for (arbitrary message length) single-turn protocols Tauman Kalai, Komargodski, and Raz [DISC '18]. Yet, the question for many-turn protocols was left completely open.**

In this work we close the above gap, proving that *no* n-party protocol (of any round complexity) is resilient to $\omega(\sqrt{n})$ (adaptive) corruptions.

***Index Terms*—adaptive adversaries; coin flipping; lower bound;**

I. Introduction

In a distributed (also known as, collective) *coin-flipping* protocol, Blum [7], the parties try to output a common (close to) uniform bit, even when some adversarially chosen parties try to bias the output. Coin-flipping protocols are fundamental primitives in cryptography and distributed computation, allowing distrustful parties to agree on a common random string (e.g., public randomness) to be used in their joint computation. More generally, almost any random process/protocol/algorithm hides (some form of) a coin-flipping protocol. Consider a random process whose Boolean output is far from being fixed (e.g., has noticeable variance). Such a process can be thought of as a coin-flipping protocol: the common coin is the output, and the the parties' messages serve as the process's randomness. Thus, proving lower bound on coin-flipping protocols induces limitations on the stability on such random processes in general (see Section I-B2 for a concrete example).

The focus of this work is *full-information* coin-flipping protocols, Ben-Or and Linial [4]. In this variant, the parties communicate solely over a single broadcast channel, and the Byzantine adversary [1] is assumed to be computationally *unbounded*. Two types of such adversaries are considered: a *static* adversary that chooses the parties it corrupts *before* the execution begins, and an *adaptive* adversary that can choose the parties it wishes to corrupt *during* the protocol execution (i.e., as a function of the messages seen so far). For static adversaries, full-information coin flip is well understood, and almost matching upper (protocols) and lower (attackers) bounds are known, see Section I-B. For adaptive adversaries, which are the focus of this work, much less is understood, and there are significant gaps between the upper and lower bounds. Ben-Or and Linial [4] proved that the n-party majority protocol is resilient to $O(\sqrt{n})$ corruptions (ignoring poly-logarithmic factors in n), and conjectured that this is a tight upper bound for any n-party protocol (of any round complexity). The works of Lichtenstein, Linial, and Saks [19], Goldwasser, Tauman Kalai, and Park [12] made progress towards proving the conjecture for *single-turn* (each party sends a single message) protocols, a case that was eventually proved by Tauman Kalai, Komargodski, and Raz [27]. Yet, the question for many-turn protocols was left completely open.

[1] Once it corrupts a party it completely controls it and can send arbitrary messages on its behalf.

2575-8454/20/$31.00 ©2020 IEEE
DOI 10.1109/FOCS46700.2020.00120

A. Our Results

We solve this intriguing question, showing that the output of any n-party protocol can be *fully biased* by an *adaptive* adversary corrupting $O(\sqrt{n})$ parties (ignoring poly-logarithmic factors).

Theorem I.1 (Biasing full-information coin-flipping protocols, informal)**.** *For any n-party full-information coin-flipping protocol, there exists $b \in \{0,1\}$ and an (unbounded) adversary that by adaptively corrupting $O(\sqrt{n})$ of the parties, enforces the outcome of the protocol to be b, except with probability $o(1)$.*

The above lower bound matches (up to poly-logarithmic factors) the upper bound achieved by the n-party majority protocol [4]. The bound extends to biased protocols, i.e., the protocol's expected outcome (in an all-honest execution) is not $1/2$. We also remark that the one side restriction (only possible to bias the protocol outcome to some $b \in \{0,1\}$) is inherent, as there exists, for instance, an n-party (single-turn) protocol that is resilient to $\Theta(n)$ corruptions trying to bias its outcome towards one.[2]

B. Related Work

1) Full-Information Coin Flip

We recall the main known results for n-party full-information coin-flipping protocols.

Adaptive adversaries: In the following we ignore poly-logarithmic factors in n.

Upper bounds (protocols).

Ben-Or and Linial [4] proved that the majority protocol is resilient to $O(\sqrt{n})$ corruptions.

Lower bounds (attacks).

Lichtenstein, Linial, and Saks [19] proved that no *single-bit* (a message is one bit), single-turn protocol is resilient to $\Omega(\sqrt{n})$ adaptive corruptions (hence, majority is optimal for such protocols). Dodis [9] proved that it is impossible to create a coin-flipping protocol resilient to $\Omega(\sqrt{n})$ adaptive corruptions by *sequentially repeating* another coin-flipping protocol, and then applying a deterministic function to the outcomes. Goldwasser, Tauman Kalai, and Park [12] proved that that no *symmetric* single-turn (many-bit) protocol is resilient to $\Omega(\sqrt{n})$ adaptive corruptions. Their result extends to *strongly adaptive* attacks (the attacker can decide to corrupt a party *after* seeing the message it is about to send) on single-turn protocols. Tauman Kalai, Komargodski, and Raz [27] fully answered the single-turn case by proving that no single-turn protocol is resilient to $\Omega(\sqrt{n})$ adaptive corruptions. Lastly, Etesami, Mahloujifar, and Mahmoody [10] presented an *efficient* and optimal strongly adaptive attack on protocols of certain properties (e.g., public coins).

Static adversaries: The case of static adversaries is well studied and understood.

Upper bounds (protocols).

Ben-Or and Linial [4] presented a protocol that tolerates $O\left(n^{0.63}\right)$ corrupted parties (an improvement on the $O(\sqrt{n})$ corrupted parties it takes to bias the majority protocol). Ajtai and Linial [1] presented a protocol that tolerates $O(n/\log^2 n)$ corruptions. Saks [26] presented a protocol that tolerates $O(n/\log n)$ corruptions. The protocol of [26] was later improved by Alon and Naor [2] to tolerate a constant fraction of corrupted parties. Shortly afterwards, Boppana and Narayanan [8] presented an optimal protocol resilient to $(1/2 - \delta)n$ corruptions for any $\delta > 0$.

Lower bounds (attacks).

Kahn, Kalai, and Linial [18] proved that no single-bit single-turn protocol can tolerate $\Omega(n/\log n)$ corruptions. Russell, Saks, and Zuckerman [25] proved that a protocol tolerating $\Omega(n)$ corruptions is either many-bit or has $\Omega(1/2 - o(1)) \cdot \log^*(n)$ rounds.

2) Data-Poisoning Attacks

Consider a learning algorithm that tries to learn an hypothesis from a training set comprised of samples taken from different sources. The random process corresponding to this learning task can be naturally viewed as (some form of) a coin-flipping protocol. As first noticed by Mahloujifar and Mahmoody [20], an attacker on the latter coin flip induces a so-called *data-poisoning attack*: increasing the probability of a desired property (i.e., poisoning the training data) by tampering with a small number of sources. For this application, however, the attacker would better have the ability to force a

[2] Consider the n-party single-turn protocol in which each party broadcasts a $(1/n, 1 - 1/n)$-biased bit (i.e., equals zero with probability $1/n$) and the protocol output is set to the AND of these bits. It is clear that the protocol expected outcome is $(1 - 1/n)^n \approx 1/e$ (can be made $1/2$ by slightly changing the distribution), and that even $n/2$ adaptive corruptions cannot change the protocol outcome to a value larger than $(1 - 1/n)^{n/2} \approx \sqrt{1/e}$.

predetermined output (rather than forcing some output, as our attack achieves). Hence, the attack on coin-flipping protocol we apply should be *bi-directional* (have to ability to (almost-) fully determine the coin, rather than biasing it to some arbitrary value). While this goal is unachievable in some models (see Footnote 2), it is achievable in some important ones.

Mahloujifar and Mahmoody [20] translated a two-directional *static* attack into a static data-poisoning attack on learning algorithms. Their attack was further improved in [22, 23]. Mahloujifar and Mahmoody [21] translated the two-directional *adaptive* attack of [27] on single-bit, single-turn coin-flipping protocols, into an adaptive data-poisoning attack. Finally, Etesami et al. [11] facilitated their strongly adaptive attack on single-turn coin-flipping protocols (see Section I-B) to obtain a strongly-adaptive data-poisoning attack.

Previously all known adversaries dealt only with *single-turn* coin-flipping protocols, translating into data-poisoning attacks which tamper some small set of samples. With the tools we present (see Theorem I.1) it is now possible to discuss data-poisoning attacks that corrupts a small amount of *sources* (rather than tampering samples without consideration of the sources they are sampled from).

Open Questions & Future Work

In this work we show that the expected outcome of *any* n-party full-information coin-flipping protocol can be biased to either $o(1)$ or to $1-o(1)$, using $O(\sqrt{n})$ corruptions. The above $o(1)$, however, stands for $1/\operatorname{loglog}(n)$, and it remains an intriguing question whether it can be pushed to $2^{-\operatorname{polylog}(n)}$ as can be achieved, for instance, when attacking the n-party majority protocol. Such attacks are known for *uniform* single-bit single-turn protocols (a secondary result of [27]) and for *strongly adaptive* attackers against single-turn protocols [10].

A second question, motivated by the data-poisoning attacks described in Section I-B2, is what security models enable $o(n)$-corruption *bi-directional* attacks (the attacker can bias the protocol output to both directions). In a work in progress, [15], we show that in the strongly-adaptive corruption model, a $O(\sqrt{n})$-corruption *bi-directional* attack exists against any protocol.

Paper Organization

A rather elaborated description of our attack on coin-flipping protocols is given in Section II. Refer to the full version for the formal proofs [14].

II. Our Technique

In this section we give a rather elaborated description of our adaptive attack, and its analysis, on full-information coin-flipping protocols. Let Π be an n-party, ℓ-round full-information coin-flipping protocol. We prove that one can either bias the expected outcome of Π to less than $\varepsilon := 1/\operatorname{loglog}(n)$, or to lager than $1-\varepsilon$.

Similarly to previous adaptive attacks on full-information coin-flipping protocols, our attack exploits the "jumps" in the protocol expected outcome; assume without loss of generality (refer to the preliminaries of the full version for justification) that in each round only a *single* party sends a message and let $\mathrm{Msg} = (\mathrm{Msg}_1, \ldots, \mathrm{Msg}_\ell)$ denote the protocol transcript (i.e., parties' messages) in a random all-honest execution of Π. For $\mathrm{msg} \in \mathrm{Supp}(\mathrm{Msg})$, let $\Pi(\mathrm{msg})$ denote the final outcome of the execution described by msg, and for $\mathrm{msg}_{\le i} \in \mathrm{Supp}(\mathrm{Msg}_{\le i} := (\mathrm{Msg}_1, \ldots, \mathrm{Msg}_i))$ let $\Pi(\mathrm{msg}_{\le i}) := \mathbb{E}\big[\Pi(\mathrm{Msg}) \mid \mathrm{Msg}_{\le i} = \mathrm{msg}_{\le i})\big]$ be the expected outcome given a partial transcript. We refer to $\Pi(\mathrm{Msg}_{\le i}) - \Pi(\mathrm{Msg}_{<i})$ (i.e., the change in the expected outcome induced by the i^{th} message) as the i^{th} jump in the protocol execution. Our attack exploits these gaps in a very different manner than what previous attacks did. First, the decision whether to corrupt a given message is based on the (conditional) *variance* of the jumps (L_2 norm), a more subtle measure than the *maximal* possible change (L_∞ norm) considered by previous attacks. Second, even when it decides that the next message is useful for biasing the protocol's outcome, it only *gently* alters the message: it corrupts the party about to send the message with a certain probability, and when corrupting, only moderately changes the message distribution. Being gentle allows the attack to bypass the main obstacles in attacking many-turn protocols (see details in Section II-B). Furthermore, in some setting of interest the attack can be made efficient. Efficient attacks where not even known for the single-turn, arbitrary message length case (the recursive attack of [27] is inherently inefficient). Finally, the gentleness of the attack make analyzing it a rather simple (and pleasurable!) task; the transcript of the gently attacked execution is not "too different" from the all-honest (unattacked) execution of the protocol. Consequently, the analysis requires one only to get a good understanding of the all-honest execution, and not of the typically very complicated

execution the attack induces.[3] Details below.

We start by describing an attack on *robust* protocols—for some $b \in \{0,1\}$, the protocol has *no* $1/\sqrt{n}$ jumps towards b.[4] For correctness, we focus on robust protocols with respect to $b = 0$. That is, we assume that (for simplicity, with probability one)

$$\Pi(\mathrm{Msg}_{\le i}) \ge \Pi(\mathrm{Msg}_{<i}) - 1/\sqrt{n} \tag{1}$$

We start with the case of single-turn robust protocols, and extend it to the many-turn (robust) setting in Section II-B. The extension to arbitrary (non-robust) protocols is described in Section II-C.

A. Attacking Robust Single-Turn Coin Flip

When corrupting a *single* message of the attacked protocol, we replace the original (honest) message distribution with the following parameterized variant.

Definition II.1 (Biased)**.** *For a distribution P, a constant $\alpha \ge 0$ and a function $f : \mathrm{Supp}(P) \mapsto [-1/\alpha, \infty)$ such that $\mathbb{E}\big[f(P)\big] = 0$, let $\mathrm{Biased}^f_\alpha\big(P\big)$ be the distribution defined by*

$$\mathbb{P}\big[\mathrm{Biased}^f_\alpha\big(P\big) = x\big] := \mathbb{P}\big[P = x\big] \cdot (1 + \alpha \cdot f(x))$$

That is, the distribution P is "nudged" towards larger values of f, i.e., increasing the probability of positive elements (causing $\mathbb{E}\big[f\big(\mathrm{Biased}^f_t\big(P\big)\big)\big] \ge \mathbb{E}\big[f(P)\big]$), and the larger α is the larger the bias. The function f to be considered by our attack is the protocol's expected output given the current message. While it is tempting to choose t as large as possible, and thus maximize a single round effect on the final outcome, our attack takes a more subtle approach.

The attack: Our attacker on an n-party, single-turn (n-round), robust protocol Π is defined below. In the following let $\mathrm{jump}(\mathrm{msg}_{\le i}) := \Pi(\mathrm{msg}_{\le i}) - \Pi(\mathrm{msg}_{<i})$, i.e., the difference in expected outcome induced by the last (i^{th}) message in the partial transcript.

Algorithm II.2 (Single-turn attacker)**.**

For $i = 1$ to n, do the following before *the i^{th} message is sent:*

1) *Let $\mathrm{msg}_{<i}$ be the previously sent messages, let $Q_i := \mathrm{Msg}_i\,|_{\mathrm{Msg}_{<i}=\mathrm{msg}_{<i}}$, let $\mathrm{jump}_i := \mathrm{jump}(\mathrm{msg}_{<i}, \cdot)$ and let $v_i := \mathrm{Var}[\mathrm{jump}_i(Q_i)]$.*
2) *If $v_i \ge 1/n$, corrupt the i^{th} party with probability $1/\varepsilon^3 \cdot \sqrt{v_i}$. If corrupted, instruct it to send the next message according to $\mathrm{Biased}^{\mathrm{jump}_i}_{1/\sqrt{v_i}}\big(Q_i\big)$.*
   Else, corrupt the i^{th} party with probability $1/\varepsilon^3 \cdot 1/\sqrt{n}$. If corrupted, instruct it to send the next message according to $\mathrm{Biased}^{\mathrm{jump}_i}_{\sqrt{n}}\big(Q_i\big)$. [5]

That is, a message (party) is corrupted with probability proportional to the (conditional) standard deviation it induces on the expected outcome of Π, or $1/\sqrt{n}$ if it is smaller. If corrupted, the message distribution is modified so that the change it induces on the expected outcome of Π is biased towards one, where the bias is proportional to the inverse of the standard deviation (up to $\sqrt{n}$).[6] In the following we argue that the attacker indeed biases the expected outcome of Π to $1-\varepsilon$, and that the expected number of corruptions is $O(\sqrt{n}/\varepsilon^3)$. Thus, a Markov bound yields the existence of the required attacker.

We prove the success of our attack by showing that the attacked protocol has too little "liveliness" to resist the attacker bias, and thus the final outcome is (with high probability) the value the attacker biases towards. Our notion of liveliness is the *conditional variance* of some underlying distribution induced by the attack. Having little liveliness according to our notion almost directly implies the success of the attack, with no need for additional tail inequalities as used by some of the previous works.

Let $\widehat{\mathrm{Msg}} = (\widehat{\mathrm{Msg}}_1, \ldots, \widehat{\mathrm{Msg}}_n)$ be the message distribution induced by the above attack, and consider the *sub*-martingale $S = (S_0, \ldots, S_n)$ with respect to $\widehat{\mathrm{Msg}}$ defined as $S_i := \Pi(\widehat{\mathrm{Msg}}_{\le i})$, i.e., the expected honest outcome (if all messages were sampled honestly from now on) at every step of the biased execution. By definition, $S_0 = \mathbb{E}\big[\Pi(\mathrm{Msg})\big] = 1/2$ and $S_n \in \{0,1\}$.

For $i \in [n]$, let Q_i be the value of Q_i in the attacked execution, determined by $\widehat{\mathrm{Msg}}_{i-1}$, and let $Y_i := \mathrm{jump}(\widehat{\mathrm{Msg}}_{<i}, q)$ for $q \leftarrow Q_i$. Since Q_i is the i^{th} honest message distribution (determined by $\widehat{\mathrm{Msg}}_{<i}$), it holds

[3] Gentle attacks, in the general sense that the attacker does not try to maximize the effect of the attack in each round, but rather keeps the attacked execution similar to the all-honest one, were found useful in many other settings. A partial list includes attacking different types of coin-flipping protocols [24, 16, 5, 3], and proving parallel repetition of computationally sound proofs [17, 13, 6].

[4] An almost accurate example for a (bi-directional) robust protocol is the single-bit, single-turn, n-party *majority* protocol: in each round, a single party broadcasts an unbiased coin, and the protocol's final output is set to the majority of the coins. It is well known that the absolute value of most jumps is (typically) order of $1/\sqrt{n}$.

[5] Since we assume Equation (1), in both cases $\mathrm{Biased}\big(Q_i\big)$ is indeed a valid probability distribution.

[6] Assuming Π is the single-turn, n-party, single-bit majority protocol, then (typically) each v_i is of (absolute) order $1/n$. Thus, in expectation, the above attack corrupts $1/\varepsilon^3 \cdot \sqrt{n}$ parties. If corrupted, the party's bit message is set to 1 with probability $\approx 1/2 \cdot (1 + \sqrt{n} \cdot 1/\sqrt{n}) = 1$.

that $\mathbb{E}\big[Y_i \mid \widehat{\mathrm{Msg}}_{<i}\big] = 0$. Since the Y_i's are also independent of each other conditioned on $\widehat{\mathrm{Msg}}$, the sequence $Y = (Y_1, \ldots, Y_n)$ is a *martingale difference sequence with respect to* $(\widehat{\mathrm{Msg}}_i, Y_i)$, where the corresponding martingale is (somewhat of) an honest execution, coupled with the biased execution. The core of our analysis lies in the following lemma.

Lemma II.3. $\mathbb{E}\big[\sum_{i=1}^{n} \mathrm{Var}[Y_i \mid \widehat{\mathrm{Msg}}_{<i}]\big] \leq \varepsilon^3$.

The proof of Lemma II.3 is sketched below, but first we use it for analyzing the quality of the attack. We first argue about the expected number of corruptions. By construction, the probability the attacker corrupts the i^{th} party is

$$\begin{aligned} &1/\varepsilon^3 \cdot \max\Big\{\sqrt{\mathrm{Var}[Y_i \mid \widehat{\mathrm{Msg}}_{<i}]}, 1/\sqrt{n}\Big\} \\ &\leq 1/\varepsilon^3 \cdot \max\Big\{\sqrt{n} \cdot \mathrm{Var}[Y_i \mid \widehat{\mathrm{Msg}}_{<i}], 1/\sqrt{n}\Big\} \\ &\leq 1/\varepsilon^3 \cdot (\sqrt{n} \cdot \mathrm{Var}[Y_i \mid \widehat{\mathrm{Msg}}_{<i}] + 1/\sqrt{n}). \end{aligned} \tag{2}$$

Hence by Lemma II.3, the expected number of corruptions is bounded by

$$\begin{aligned} 1/\varepsilon^3 \cdot \mathbb{E}\Big[\sum_{i=1}^{n}\Big(\sqrt{n} \cdot \mathrm{Var}[Y_i \mid \widehat{\mathrm{Msg}}_{<i}] + 1/\sqrt{n}\Big)\Big] \\ \leq \sqrt{n}(1 + 1/\varepsilon^3). \end{aligned}$$

We next argue about the bias induced by the attack. Since Y is a martingale difference sequence with respect to $(\widehat{\mathrm{Msg}}_i, Y_i)$, i.e., $\mathbb{E}\big[Y_i \mid \widehat{\mathrm{Msg}}_{<i}, Y_{<i}\big] = 0$, it is easy to verify that

$$\mathbb{E}\Big[\Big(\sum_{i=1}^{n} Y_i\Big)^2\Big] = \sum_{i=1}^{n} \mathbb{E}\big[Y_i^2\big] = \mathbb{E}\Big[\sum_{i=1}^{n} \mathrm{Var}[Y_i \mid \widehat{\mathrm{Msg}}_{<i}]\Big] \tag{3}$$

Hence by Lemma II.3 and Chebyshev's inequality, we deduce that

$$\mathbb{P}\Big[\Big|\sum_{i=1}^{n} Y_i\Big| \geq \varepsilon\Big] \leq \varepsilon \tag{4}$$

Furthermore, since S_i are the "biased towards one" variants of Q_i (and thus of Y_i), there exists a (rather) straightforward coupling between S and Y for which

$$S_i - S_{i-1} \geq Y_i \tag{5}$$

Since, by definition, $S_0 = 1/2$, it follows that $\mathbb{P}\big[S_n \leq 0\big] = \mathbb{P}\big[\sum_{i=1}^{n} Y_i \leq -1/2\big] \leq \varepsilon$, and since $S_n \in \{0, 1\}$, we deduce that $\mathbb{P}\big[S_n = 1\big] \geq 1 - \varepsilon$. Namely, the output of the attacked protocol is 1 with probability at least $1 - \varepsilon$. (The same argument works also if it is only guaranteed that $S_0 \geq \varepsilon$, as happens in Section II-C)

Proving Lemma II.3: We start with two simple observations. The first is that for any distribution P, constant $alpha \geq 0$ and function $f : \mathrm{Supp}(P) \mapsto [-1/\alpha, \infty)$ such that $\mathbb{E}\big[f(P)\big] = 0$, it holds that

$$\begin{aligned} &\mathbb{E}\big[f\big(\mathrm{Biased}_{\alpha}^{f}(P)\big)\big] \\ &= \sum_{x \in \mathrm{Supp}(P)} f(x) \cdot \mathbb{P}\big[\mathrm{Biased}_{\alpha}^{f}(P) = x\big] \\ &= \sum_{x \in \mathrm{Supp}(P)} f(x) \cdot \mathbb{P}\big[P = x\big] \cdot (1 + \alpha \cdot f(P)) \\ &= \mathbb{E}\big[f(P) \cdot (1 + \alpha \cdot f(P))\big] = \mathbb{E}\big[f\big] + \alpha \cdot \mathbb{E}\big[f^2(P)\big] \\ &= 0 + \alpha \cdot \mathrm{Var}\big[f(P)\big]. \end{aligned} \tag{6}$$

A second immediate observation is that for any $p \in [0, 1]$:

$$\Big(p \cdot \mathrm{Biased}_{\alpha}^{f}(P) + (1 - p) \cdot P\Big) \equiv \mathrm{Biased}_{p \cdot \alpha}^{f}(P) \tag{7}$$

Let V_i be the value of the variable v_i in the execution of the attack (determined by $\widehat{\mathrm{Msg}}_{<i}$), and let $V_i' := \max\{v_i, 1/n\}$. For a partial transcript $\mathrm{msg}_{<i} \in \mathrm{Supp}(\widehat{\mathrm{Msg}}_{<i})$, applying the above observations with respect to $P := \mathrm{Msg}_i\big|_{\mathrm{Msg}_{<i} = \mathrm{msg}_{<i}}$, $p := 1/\varepsilon^3 \cdot \sqrt{V_i'\big|_{\widehat{\mathrm{Msg}}_{<i} = \mathrm{msg}_{<i}}}$, $\alpha := 1/\sqrt{V_i'\big|_{\widehat{\mathrm{Msg}}_{<i} = \mathrm{msg}_{<i}}}$ and $\mathrm{jump}_i := \mathrm{jump}(\mathrm{msg}_{<i}, \cdot)$, yields that

$$\begin{aligned} &\mathbb{E}\big[S_i - S_{i-1} \mid \widehat{\mathrm{Msg}}_{<i} = \mathrm{msg}_{<i}\big] \\ &= \mathbb{E}\big[\mathrm{jump}_i\big(\mathrm{Biased}_{1/\varepsilon^3}^{\mathrm{jump}_i}\big(\mathrm{Msg}_i\big|_{\mathrm{Msg}_{<i} = \mathrm{msg}_{<i}}\big)\big)\big] \\ &= 1/\varepsilon^3 \cdot \mathrm{Var}\big[\mathrm{jump}(\mathrm{Msg}_{\leq i}) \mid \mathrm{Msg}_{<i} = \mathrm{msg}_{<i}\big] \\ &= 1/\varepsilon^3 \cdot \mathrm{Var}\big[Y_i \mid \widehat{\mathrm{Msg}}_{<i} = \mathrm{msg}_{<i}\big]. \end{aligned} \tag{8}$$

It follows that

$$\mathbb{E}\big[S_n - S_0\big] = 1/\varepsilon^3 \cdot \mathbb{E}\Big[\sum_{i=1}^{n} \mathrm{Var}[Y_i \mid \widehat{\mathrm{Msg}}_{<i}]\Big] \tag{9}$$

and since both S_0 and S_n take values in $[0, 1]$, we conclude that $\mathbb{E}\big[\sum_{i=1}^{n} \mathrm{Var}[Y_i \mid \widehat{\mathrm{Msg}}_{<i}]\big] \leq \varepsilon^3$.

B. Attacking Robust Many-Turn Coin Flip

Moving to many-turn protocols, one can no longer decide whether to attack a message *independently* of the other messages. There are simply too many massages, and whatever such strategy one takes, it either corrupts too many parties, or biases the protocol's outcome by too little. So rather, the strategy to consider is to decide whether to corrupt or not, *per party*, and not per message, with the exception of "highly influential" messages. Given this mandatory change, the main challenge is that

once deciding to corrupt a party, we should corrupt its messages in a way that does not significantly reduce the the influence of its future messages. Otherwise, a corrupt party might never be useful for biasing the protocol outcome. For instance, consider the n-party n^2-round majority protocol, i.e., each party sends n bits, that is equipped with the following "punishing" mechanism: once a party's coins are "too suspicious", say contain a 1-run of length $\log^2 n$, its coins are ignored from this point on. If the protocol is single turn (i.e., each party sends all its bis in a single message), then the effect of such punishing mechanism can be taken into consideration at the time of corrupting the party (single) message. But if the parties send their bits in turns (compared to all at once, like in the single-turn case), the task of attacking protocols with more (implicit) unpredictable mechanisms that determine message influence, is much more challenging. Here the gentle approach described in the previous section is extremely useful. Actually, being useful in such scenarios is what we had in mind when designing it in the first place.

A second challenge is that that attacker has to decide whether to corrupt a party *before* it is certain the party can be used to significantly bias the protocol outcome; the influence a party has on the final outcome might vary between different executions, and there is no way to tell in advance whether a certain party will be influential since other parties (which we do not corrupt) introduce randomness to the process. We overcome this obstacle by choosing the parties to corrupt not merely based on the influence of their next message, as we did in the single-turn case. Rather, in addition to the single-case mechanism, each party is corrupted with (fixed) probability $1/\sqrt{n}$, and it reenters the "lottery" every time the messages it sent had sufficient (potential) influence on the outcome. To keep the attack gentle, the messages of the chosen to be corrupted party are modified only to gain a moderate bias.

Normal protocols: With the realization that parties should be given multiple chances to become corrupted (the "lottery" mentioned above), the presentation of the attack is simplified by splitting the parties into pseudo-parties for which we decide *independently* whether to corrupt or not. Those pseudo-parties have relatively small *overall* influence on the protocol's outcome, except the unavoidable case of having a single very influential message.

Definition II.4 (Normal protocols, informal)**.** *Let* $\mathsf{party}(\mathrm{msg}_{<i})$ *be the identity of the party about to send the* i^{th} *message as described by the partial transcript* $\mathrm{msg}_{<i}$*. A party* P *has a* large jump *in* $\mathrm{msg} \in \mathrm{Supp}(\mathrm{Msg})$*, if* $\mathsf{party}(\mathrm{msg}_{<i}) = \mathsf{P}$ *for some* $i \in [\ell]$ *such that*

$$\mathrm{Var}[\mathrm{jump}(\mathrm{Msg}_{\le i}) \mid \mathrm{Msg}_{<i} = \mathrm{msg}_{<i}] \ge 1/n$$

An n*-party protocol is* normal *if the following hold:*[7]

1) *A large-jump party sends only a* single *message.*
2) *For a small-jumps party, a party that has no large jumps,* P *it holds that*

$$\sum_{\substack{i\in[\ell]:\\ \mathsf{party}(\mathrm{Msg}_{<i})=\mathsf{P}}} \mathrm{Var}[\mathrm{jump}(\mathrm{Msg}_{\le i}) \mid \mathrm{Msg}_{<i}] \le 1/n$$

(I.e., the overall sum of conditional variances the party P *"has" is bounded.)*

We enforce normality by partitioning, if needed, the messages of the parties into sequences, viewing each such sequence as a separate pseudo-party. The advantage of considering protocols in their normal form is that our attack, described below, either corrupts *all* messages sent by a (pseudo-)party, or corrupts *none* of them.

The attack: Our attacker on an n-party, ℓ-round, robust, normal protocol Π is defined as follows:

Algorithm II.5 (Many-turn attacker)**.**

For $i = 1$ *to* ℓ*, do the following* before *the* i^{th} *message is sent:*

1) *Let* $\mathrm{msg}_{<i}$ *be the previously sent messages, let* $Q_i := \mathrm{Msg}_{\le i}|_{\mathrm{Msg}_{<i}=\mathrm{msg}_{<i}}$*, let* $\mathrm{jump}_i := \mathrm{jump}(\mathrm{msg}_{<i}, \cdot)$*, and let* $v_i := \mathrm{Var}[\mathrm{jump}_i(Q_i)]$.
2) *If* $v_i \ge 1/n$*, corrupt the party sending the* i^{th} *message with probability* $1/\varepsilon^3 \cdot \sqrt{v_i}$*. If corrupted, instructs it to send its next message according to* $\mathrm{Biased}^{\mathrm{jump}_i}_{1/\sqrt{v_i}}(Q_i)$.
   Else, if the i^{th} *message is the first message to be sent by the party, corrupt this party with probability* $1/\varepsilon^3 \cdot 1/\sqrt{n}$*. If corrupted (now or in previous rounds), instruct it to send its next message according to* $\mathrm{Biased}^{\mathrm{jump}_i}_{\sqrt{n}}(Q_i)$.

That is, a large-jump party is treated like in the single-turn case, whereas a small-jumps party is corrupted with probability proportional to $1/\sqrt{n}$ (again like in the single-turn case)—when corrupted, *all* messages

[7] Actually, we can only guarantee a relaxed variant of the normality condition.

of the small-jumps party are modified.[8] The analysis of the above attack is similar to the single-turn case. Let $\widehat{\mathrm{Msg}} = (\widehat{\mathrm{Msg}}_1, \ldots, \widehat{\mathrm{Msg}}_\ell)$, $S = (S_0, \ldots, S_\ell)$ and $Y = (Y_1, \ldots, Y_\ell)$ be as in the single-turn case. Similarly to the single turn case, the core of the proof lies in the following lemma.

Lemma II.6. $\mathbb{E}\big[\sum_{i=1}^{\ell} \mathrm{Var}[Y_i \mid \widehat{\mathrm{Msg}}_{<i}]\big] = O(\varepsilon^3)$.

The challenge in proving Lemma II.6 is that unlike the single-turn proof, it might be that the following does not hold:[9]

$$\mathbb{E}\big[S_i - S_{i-1} \mid \widehat{\mathrm{Msg}}_{<i} = \mathrm{msg}_{<i}\big] \geq 1/\varepsilon^3 \cdot \mathrm{Var}[Y_i \mid \widehat{\mathrm{Msg}}_{<i} = \mathrm{msg}_{<i}]$$

Indeed, let V_i be the value of the variables v_i in the execution of the attack described by $\widehat{\mathrm{Msg}}$. Assume that conditioned on $\widehat{\mathrm{Msg}}_{<i} = \mathrm{msg}_{<i}$, it holds that $V_i < 1/n$ and that a small-jumps party P is about to send the i^{th} message. Unlike the single-turn case, the conditional probability that P is corrupted is no longer guaranteed to be $1/\varepsilon^3 \cdot 1/\sqrt{n}$: the previous messages sent by P in $\mathrm{msg}_{<i}$ might *leak* whether P is corrupted or not. If the latter happens, then (by the same argument we used for proving the lemma in the single-turn case) it might be that $\mathbb{E}\big[S_i - S_{i-1} \mid \widehat{\mathrm{Msg}}_{<i} = \mathrm{msg}_{<i}\big] < 1/\varepsilon^3 \cdot \mathrm{Var}[Y_i \mid \widehat{\mathrm{Msg}}_{<i} = \mathrm{msg}_{<i}]$. Fortunately, since we only slightly modify each message of a corrupted small-jumps party (proportionally to the conditional variance the message induces on the protocol's outcome), and since (due to the partitioning) the overall variance of the messages a small-jumps party sends is at most $1/n$, a KL-divergence argument yields that on average (in some sense) for a message sent by a small-jumps party it holds that $\mathbb{E}\big[S_i - S_{i-1} \mid \widehat{\mathrm{Msg}}_{i-1} = \mathrm{msg}_{<i}\big] = \Omega(1/\varepsilon^3 \cdot \mathrm{Var}\big[Y_i \mid \widehat{\mathrm{Msg}}_{i-1} = \mathrm{msg}_{<i}\big])$, which suffices for the proof of the lemma to go through.

C. Attacking Non-Robust Coin Flip

The high level idea of attacking a non-robust protocol (has large jumps to both directions) is trying to transform it into a (almost) robust protocol with respect to $b = 0$ (with no large jumps downward), with the assurance that if we fail transforming it into a robust protocol, it would already be completely biased (towards 0). If we succeeded transforming it into a (almost) robust protocol we apply our attack on robust protocols.

More formally, assume that with probability at least $1/\log n$, Π has a large negative jump, i.e., $-1/\sqrt{n}$, and consider the following "one-shot" attacker on Π:[10]

Algorithm II.7 (Negative jumps attacker)**.**

For $i = 1$ to n, do the following before *the i^{th} message is sent:*

1) *Let $\mathrm{msg}_{<i}$ be the previously sent messages.*
2) *If there exists $m_i^- \in \mathrm{Supp}(\mathrm{Msg}_i|_{\mathrm{Msg}_{<i}=\mathrm{msg}_{<i}})$ such that $\Pi(\mathrm{msg}_{<i}, m_i^-) < \Pi(\mathrm{msg}_{<i}) - 1/\sqrt{n}$, and no party was corrupted yet, instruct the party sending the i^{th} message to send m_i^-.*

It is clear that the above adversary biases the outcome of Π toward zero by at least $1/\sqrt{n}\log(n)$. Let Π_1 be the *protocol* induced by the above (deterministic) attack: all parties emulate the attacker in their head, and when it decides to (deterministically) corrupt a party, the corrupted party follows its (deterministic) instructions. If the protocol Π_1 has a large negative jump with probability larger than $1/\log n$, apply the above attack on Π_1 resulting in the protocol Π_2, and so on. Let $t \leq \sqrt{n} \cdot \log(n)$ denote the number of these attack applications. If the expected outcome of Π_t is at most ε, then we are done: the t-adaptive adversary that runs these t attacks iteratively, makes Π output 0 with probability $1 - \varepsilon$. Otherwise, Π_t has the following property:

$$\mathbb{P}\big[\exists j \in [n] \colon \Pi_t(\widetilde{\mathrm{Msg}}_{\leq j}) < \Pi_t(\widetilde{\mathrm{Msg}}_{<j}) - 1/\sqrt{n}\big] \leq 1/\log(n) \quad (10)$$

letting $\widetilde{\mathrm{Msg}}$ be the messages of a random execution of Π_t. If the above happens, then we apply the attack on robust protocols (Algorithm II.5) on Π_t, instructing the adversary to halt if it encounters a large negative jump. With careful analysis (actually, we need to slightly refine the attack for that), one can show that the above attack on Π_t encounters large negative jumps with probability $O(\varepsilon)$ (recall that we set ε to $1/\log\log(n)$). Hence, it successfully biases the expected output of Π_t to $1 - O(\varepsilon)$ (since with overwhelming probability the attack carries as if there are no large negative jumps). Composing the

[8] Assuming Π is an n-party, n^2-round, single-bit majority protocol in which each party sends n bits, then (typically) the change induced by any given message is (absolute) order of $1/n$. Hence, each v_i is of order $1/n^2$, and each party will be independently corrupted with probability $1/\varepsilon^3 \cdot 1/\sqrt{n}$ (i.e., first *if* of Step (2) is never triggered). Thus, in expectation, the above attack corrupts $1/\varepsilon^3 \cdot \sqrt{n}$ parties. If corrupt, each of the n bit-messages the party sends is 1 with probability $\approx 1/2 \cdot (1 + \sqrt{n} \cdot 1/n) = 1/2 + 1/2\sqrt{n}$.

[9] Recall that the above equality allowed us to argue that $\mathbb{E}\big[\sum_{i=1}^{\ell} \mathrm{Var}[Y_i \mid \widehat{\mathrm{Msg}}_{<i}]\big] \leq \varepsilon^3 \cdot \mathbb{E}\big[S_\ell - S_0\big] \leq \varepsilon^3$.

[10] Interestingly, assuming the next-message function of Π is efficient, e.g., Π is public-coin, the following attacker is the only reason for the inefficiency of our attack.

attack that transforms Π into Π_t with the attack on Π_t, yields the required attack on Π.

Acknowledgment

We are grateful to Raz Landau, Nikolaos Makriyannis, Eran Omri and Eliad Tsfadia for very helpful discussions. The first author is thankful to Michael Ben-Or for encouraging him to tackle this beautiful question.

Research supported by ERC starting grant 638121, and Israel Science Foundation grant 666/19, Check Point Institute for Information Security.

References

[1] M. Ajtai and N. Linial. The influence of large coalitions. *Combinatorica*, 13(2):129–145, 1993.

[2] N. Alon and M. Naor. Coin-flipping games immune against linear-sized coalitions. *SIAM Journal on Computing*, 22(2):403–417, 1993.

[3] A. Beimel, I. Haitner, N. Makriyannis, and E. Omri. Tighter bounds on multi-party coin flipping via augmented weak martingales and differentially private sampling. In *Proceedings of the 59th Annual Symposium on Foundations of Computer Science (FOCS)*., 2018.

[4] M. Ben-Or and N. Linial. Collective coin flipping, robust voting schemes and minima of banzhaf values. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 408–416, 1985.

[5] I. Berman, I. Haitner, and A. Tentes. Coin flipping of any constant bias implies one-way functions. *Journal of the ACM*, 65(3):14, 2018.

[6] I. Berman, I. Haitner, and E. Tsfadia. A tight parallel-repetition theorem for random-terminating interactive arguments. In *Annual International Cryptology Conference (CRYPTO)*, 2020.

[7] M. Blum. How to exchange (secret) keys. *ACM Transactions on Computer Systems*, 1983.

[8] R. B. Boppana and B. O. Narayanan. Perfect-information leader election with optimal resilience. *SIAM Journal on Computing*, 29(4):1304–1320, 2000.

[9] Y. Dodis. Impossibility of black-box reduction from non-adaptively to adaptively secure coin-flipping. Technical Report TR00-39, Electronic Colloquium on Computational Complexity, 2000.

[10] O. Etesami, S. Mahloujifar, and M. Mahmoody. Computational concentration of measure: Optimal bounds, reductions, and more. In *Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 345–363, 2020.

[11] O. Etesami, S. Mahloujifar, and M. Mahmoody. Computational concentration of measure: Optimal bounds, reductions, and more. In *Symposium on Discrete Algorithms (SODA)*, pages 345–363, 2020.

[12] S. Goldwasser, Y. Tauman Kalai, and S. Park. Adaptively secure coin-flipping, revisited. In *Automata, Languages and Programming, 24th International Colloquium (ICALP)*, pages 663–674, 2015.

[13] I. Haitner. A parallel repetition theorem for any interactive argument. *SIAM Journal on Computing*, 42(6):2487–2501, 2013.

[14] I. Haitner and Y. Karidi-Heller. A tight lower bound on full-information adaptively secure coin flip. 2020. URL https://eccc.weizmann.ac.il/report/2020/071/.

[15] I. Haitner and Y. Karidi-Heller. A tight lower bound on strongly adaptively secure full-information coin flip and applications to data poisining. Work in progress, 2020.

[16] I. Haitner and E. Omri. Coin Flipping with Constant Bias Implies One-Way Functions. *SIAM Journal on Computing*, pages 389–409, 2014. Preliminary version in *FOCS'11*.

[17] J. Håstad, R. Pass, D. Wikström, and K. Pietrzak. An efficient parallel repetition theorem. In *Theory of Cryptography (TCC)*, pages 1–18, 2010.

[18] J. Kahn, G. Kalai, and N. Linial. The influence of variables on boolean functions. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 68–80, 1988.

[19] D. Lichtenstein, N. Linial, and M. Saks. Some extremal problems arising from discrete control processes. *Combinatorica*, 9(3):269–287, 1989.

[20] S. Mahloujifar and M. Mahmoody. Blockwise p-tampering attacks on cryptographic primitives, extractors, and learners. In *Theory of Cryptography (TCC)*, pages 245–279, 2017.

[21] S. Mahloujifar and M. Mahmoody. Can adversarially robust learning leveragecomputational hardness? In *Algorithmic Learning Theory*, pages 581–609, 2019.

[22] S. Mahloujifar, M. Mahmoody, and A. Mohammed. Multi-party poisoning through generalized p-tampering. Technical Report 1809.03474, arXiv, 2018.

[23] S. Mahloujifar, D. I. Diochnos, and M. Mahmoody. The curse of concentration in robust learning: Evasion and poisoning attacks from concentration of measure. In *AAAI Conference on Artificial*

Intelligence, pages 4536–4543, 2019.

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

[25] A. Russell, M. Saks, and D. Zuckerman. Lower bounds for leader election and collective coin-flipping in the perfect information model. *SIAM Journal on Computing*, 31(6):1645–1662, 2002.

[26] M. Saks. A robust noncryptographic protocol for collective coin flipping. *SIAM Journal on Discrete Mathematics*, 2(2):240–244, 1989.

[27] Y. Tauman Kalai, I. Komargodski, and R. Raz. A lower bound for adaptively-secure collective coin-flipping protocols. In *International Symposium on Distributed Computing (DISC)*, 2018.