Efficient Distributed Coin-tossing Protocols

Hamidreza Amini Khorasgani, Hemanta K. Maji, Himanshi Mehta, Mingyuan Wang
Department of Computer Science
Purdue University
West Lafayette, Indiana, USA – 47907
Email: {haminikh, hmaji, mehta142, wang1929}@purdue.edu

2021 IEEE International Symposium on Information Theory (ISIT) | 978-1-5386-8209-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISIT45174.2021.9518000

***Abstract*—Ben-Or and Linial (1985) introduced the full information model for coin-tossing protocols involving n-processors with unbounded computational power using a common broadcast channel for all their communications. A bias-X coin-tossing protocol outputs 1 with probability X; otherwise, it outputs 0 with probability $(1 - X)$. A coin-tossing protocol's insecurity is the maximum change in the output distribution (in the statistical distance) that an adversary can cause. This work considers an adversary who monitors the protocol's communication and intervenes at most once by restarting the processor who just broadcast her message. For a given tolerance ε, our objective is to use the minimum number of processors, ensuring that this adversary can only change the output distribution by at most ε.**

Historically, the "threshold coin-tossing protocols" have been optimal or asymptotically optimal against various adversary models. However, for our model, Khorasgani, Maji, and Mukherjee (2019) prove the existence of coin-tossing protocols that achieve the same tolerance as the threshold protocols using a smaller number of processors. Unfortunately, their protocol is not computationally efficient.

Towards this objective, for any $X \in (0, 1)$ and $n \in \mathbb{N}$, this paper presents computationally efficient coin-tossing protocols approximating the new protocols of Khorasgani, Maji, and Mukherjee (2019). This protocol's running time is linear in the inverse of the accuracy parameter of this approximation, which can be set arbitrarily small.

A full version of this paper is accessible at: https://www.cs.purdue.edu/homes/hmaji/papers/KMMW20.pdf

I. Introduction

Ben-Or and Linial [1], [2] introduced the *full information* model to study collective coin-tossing protocols. In this model, all processors have an unbounded computational power and communicate over a broadcast channel. This paper studies *strong adaptive adversaries* introduced in [3] where the adversary sees a processor's message before it decides whether to corrupt that processor or not. Specifically, we consider the following model, first proposed in [4].

Representative Motivating Problem: Our Model. Consider an n-processor coin-tossing protocol in the full information model where, every round, a unique processor broadcasts her message. After each processor broadcasts her message, the output of the protocol $\in \{0, 1\}$ is a deterministic function of the broadcast messages. A bias-X coin-tossing protocol outputs 1 with probability X; otherwise, it outputs 0.

An eavesdropping adversary monitoring the messages in the broadcast channel may intervene at most once as follows. After round i, upon seeing the first i broadcast messages, the adversary decides whether to corrupt processor i or not.[1] If the adversary corrupts the processor, it restarts the processor, resulting in resampling of its private randomness. A coin-tossing protocol's *insecurity* is the maximum change in the output distribution that an adversary can cause.

Threshold Protocols. Historically, the elegant threshold coin-tossing protocols [5]–[7] have proven to be optimal in various adversarial settings. Consider $X \in (0, 1)$ such that there exists a threshold $t \in \{0, 1, \ldots, n + 1\}$ satisfying $\sum_{i=t}^{n} \binom{n}{i} \cdot 2^{-n} = X$. In a threshold-$t$ n-processor coin-tossing protocol, processor i broadcasts an independent and uniformly random bit $C_i \in \{0, 1\}$, where $1 \leqslant i \leqslant n$. The output of the protocol is 1 if (and only if) $\sum_{i=1}^{n} C_i \geqslant t$.

An adversary can increase the expected output by $\binom{n}{t-1} \cdot 2^{-(n+1)}$. To increase the expected output, the adversary shall restart an arbitrary processor who broadcasts 0. The increase in the expected outcome corresponds to the scenarios, where exactly $t - 1$ 1s were broadcast. In these scenarios, by re-sampling one 0 message, the output of the protocol changes to 1 with probability $1/2$. For the remaining scenarios, the re-sampling attack on one processor is futile. Hence, any adversary in our model can increase the expected output by $\binom{n}{t-1} \cdot 2^{-(n+1)}$.

Likewise, an adversary can decrease the expected output by $\binom{n}{t} \cdot 2^{-(n+1)}$. Consequently, the insecurity of the threshold protocol is $\varepsilon = \frac{1}{2^{n+1}} \cdot \max\left\{\binom{n}{t-1}, \binom{n}{t}\right\}$.

Khorasgani-Maji-Mukherjee Protocol. Recently, Khorasgani, Maji, and Mukherjee [8] define new coin-tossing protocols (KMM protocols) using a recursive geometric transformation (see Section III). These protocols have shown the potential to achieve lower insecurity than the threshold protocols using an equal number of processors.

Hamidreza Amini Khorasgani, Hemanta K. Maji, Himanshi Mehta, and Mingyuan Wang are supported in part by an NSF CRII Award CNS–1566499, NSF SMALL Awards CNS–1618822 and CNS–2055605, the IARPA HECTOR project, MITRE Innovation Program Academic Cybersecurity Research Awards (2019–2020, 2020–2021), a Purdue Research Foundation (PRF) Award, and The Center for Science of Information, an NSF Science and Technology Center, Cooperative Agreement CCF–0939370.

[1] We state our adversarial model in this manner to be consistent with the cryptography literature where a *strong* adversary [3] receives the broadcast message first and might block this message's delivery to other parties. In fact, at round i, when processor i has just prepared its message to broadcast, the adversary, upon seeing the first $i - 1$ broadcast messages and also the new message prepared by processor i, decides whether to corrupt processor i or not.

Authorized licensed use limited to: Purdue University. Downloaded on June 08,2022 at 16:09:20 UTC from IEEE Xplore. Restrictions apply.

TABLE I

INSECURITY OF BIAS-X COIN-TOSSING PROTOCOLS, WHERE $X \in (0, 1/2]$ AND $n = 5$. COLUMN $\varepsilon_{\text{THRESH}}$ PRESENTS THE INSECURITY OF THE THRESHOLD PROTOCOL [5]–[7], AND COLUMN ε_{KMM} PRESENTS THE INSECURITY OF THE KHORASGANI-MAJI-MUKHERJEE PROTOCOL [8].

X	$\varepsilon_{\text{Thresh}}$	ε_{KMM}	Improvement
1/32	$5/64 \approx 0.078125$	$0.0217\ldots$	$0.05642\ldots$
6/32	$10/64 \approx 0.15625$	$0.0923\ldots$	$0.06395\ldots$
16/32	$10/64 \approx 0.15625$	$0.1415\ldots$	$0.01475\ldots$

One can precisely compute the insecurity of the KMM protocols using an exponential-time algorithm. Table I illustrates this reduction in insecurity for the representative example of $n = 5$ and all $X \in (0, 1/2]$ realizable by threshold protocols.[2] Before this work, it was unknown whether one can efficiently implement the KMM protocols. This paper presents a computationally efficient approximation of the bias-X n-processor KMM protocol for any $X \in (0, 1)$ and $n \in \mathbb{N}$.

II. OUR CONTRIBUTIONS

For a bias-X distributed coin-tossing protocol Π, let $\varepsilon(\Pi)$ represent its insecurity in our model. Let $\Pi_{\text{KMM}}(n, X)$ represent the bias-X n-processor KMM protocol (see Section III). Our contributions are both theoretical and experimental.

Theoretical Results. For any $n \in \mathbb{N}$ and $X \in (0, 1)$, we present a computationally efficient bias-X n-processor coin-tossing protocol. Our protocol, denoted by $\Pi_{\text{Our}}(n, X, \delta)$, is additionally parametrized by an accuracy parameter δ. Intuitively, the smaller δ is, the more accurately our protocol $\Pi_{\text{Our}}(n, X, \delta)$ approximates the KMM-protocol $\Pi_{\text{KMM}}(n, X)$. The running time of our protocol is linearly dependent on the inverse of the accuracy parameter δ.

Theorem 1: For any $n \in \mathbb{N}$, $X \in (0, 1)$, and accuracy parameter $\delta \in (0, 1)$, there exists a computationally efficient bias-X n-processor coin-tossing protocol $\Pi_{\text{Our}}(n, X, \delta)$ s.t.

$$\varepsilon\left(\Pi_{\text{Our}}(n, X, \delta)\right) < \varepsilon\left(\Pi_{\text{KMM}}(n, X)\right) + n\delta.$$

Furthermore, processor i generates the next message in time linear in $(n - i)/\delta$, for $1 \leqslant i \leqslant n$.

Note that, Khorasgani et al. [8] show that the insecurity of the KMM protocol satisfies

$$\varepsilon\left(\Pi_{\text{KMM}}(n, X)\right) \geqslant \sqrt{\frac{1}{2(n+1)}} \cdot X(1 - X).$$

Consequently, for constant $X \in (0, 1)$, when the accuracy parameter $\delta \ll \frac{1}{n^{3/2}}$, we have

$$\varepsilon\left(\Pi_{\text{Our}}(n, X, \delta)\right) \leqslant (1 + \mathrm{o}(1)) \cdot \varepsilon\left(\Pi_{\text{KMM}}(n, X)\right).$$

Remark 1: We emphasize that one can optimize the protocol to offload (nearly) the entire computational cost to a one-time offline precomputation, independent of the bias X of the coin-tossing protocol. This precomputation step's running time is n/δ, and the next-message generation of our n'-processors coin-tossing protocol, where $n' \leqslant n$, takes constant time. Therefore, this precomputation step enables a processor to participate in an arbitrary number of coin-tossing protocol instances involving $n' < n$ processors; thus, amortizing the precomputation step's computational cost over multiple coin-tossing instances.

[2]Bias-X coin-tossing protocols are isomorphic to bias-$(1-X)$ coin-tossing protocols. Therefore, it suffices to study $X \in (0, 1/2]$.

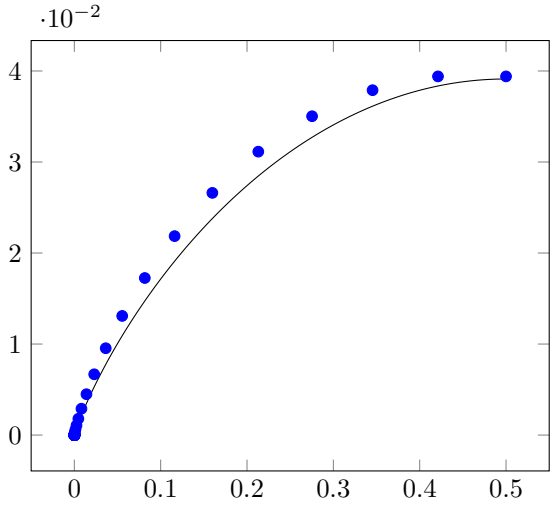

Fig. 1. For $n = 101$, the blue circles denote the insecurity of bias-X coin-tossing protocols that are implementable using a threshold protocol. Versus the plot of $\varepsilon\left(\Pi_{\text{Our}}(n, X, \delta)\right)$, for all $X \in [0, 1/2]$ and $\delta = 10^{-6}$. Lower insecurity is better.

Remark 2: The recommended order of setting the parameters of our protocols is as follows. First, decide an upper bound on the number of processors n. Next, fix a lower bound for the variance term $X(1 - X)$. Finally, choose a sufficiently small accuracy parameter δ. Setting the accuracy parameter δ to be $\mathrm{o}\big(X(1 - X)/n^{3/2}\big)$, ensures that the insecurity of our protocol is within $1 + \mathrm{o}(1)$ multiplicative factor of the optimal achievable insecurity. For instance, when $\delta = X(1 - X)/n^2$, the running time of our protocol is $n^3/X(1 - X)$.

Experimental Results. We implement our protocol and show that our protocol's insecurity is observably smaller than the insecurity of threshold protocols. As a representative example, Fig. 1 plots our protocol's insecurity, for $n = 101$ processors and $X \in [0, 1/2]$ with accuracy parameter $\delta = 10^{-6}$. Fig. 1 also plots the insecurity of all bias-X coin-tossing protocols that can be implemented using a threshold protocol. Note that the insecurity of our protocol is less than the insecurity of threshold protocol. This reduction in insecurity is prominent when X is close to $1/4$ (and, by symmetry, when X is close to $3/4$).

Finally, our experiments uncover an exciting phenomenon. Fig. 2 indicates that the insecurity of our protocols for $X = 1/2$ tends towards the insecurity of the majority protocol, as n tends to infinity. This experiment reinforces the conjecture that the majority protocol is asymptotically optimal.

Authorized licensed use limited to: Purdue University. Downloaded on June 08,2022 at 16:09:20 UTC from IEEE Xplore. Restrictions apply.

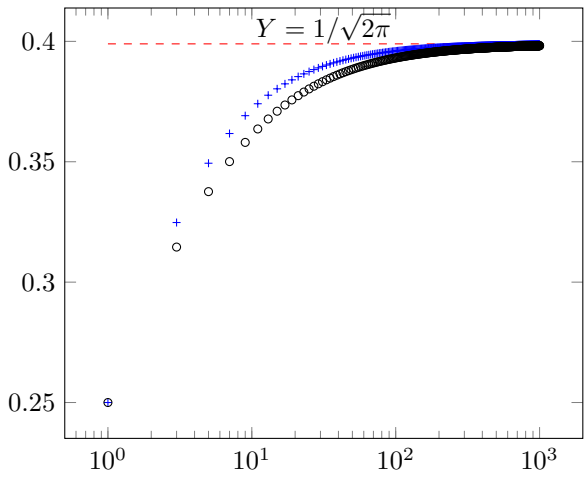

Fig. 2. For $n \in \{1, 3, \ldots, 1001\}$, the blue $+$ show the plot of $\sqrt{n} \cdot \varepsilon_{\text{Maj}}(n)$, the insecurity of the majority coin-tossing protocol. The red dashed line shows the limit of the insecurity of majority protocol using Stirling's approximation, when $n \to \infty$. The black $\circ$ show the plot of $\sqrt{n} \cdot \varepsilon\left(\Pi_{\text{Our}}(n, X, \delta)\right)$, where $X = 1/2$ and $\delta = 10^{-6}$. The graph uses log scale on the X-axis.

III. Khorasgani-Maji-Mukherjee Protocol

Let $\Pi_{\text{KMM}}(n, X)$, for $n \in \mathbb{N}$ and $X \in (0, 1)$, represent the bias-X n-processor coin-tossing protocol introduced in [8]. Let $A_n(X)$, for $n \in \mathbb{N}$, be a function with domain $[0, 1]$ that upper-bounds the insecurity of $\Pi_{\text{KMM}}(n, X)$.

Base Case. For $n = 1$, define $A_n(X) = X(1 - X)$. Let $\Pi_{\text{KMM}}(n, X)$ be the 1-processor protocol that broadcasts 1 with probability X; otherwise, the processor broadcasts 0.

Inductive Definition for $n > 1$. Assume that we already know the function $A_{n-1}(X)$ and the protocols $\Pi_{\text{KMM}}(n-1, X)$ for all $X \in [0, 1]$. Inductively, define the value $A_n(x)$ and the protocol $\Pi_{\text{KMM}}(n, x)$, for any particular $x \in [0, 1]$, as follows.

1) Let $Z = x_0 \in [0, x]$ be the (unique) solution of the equation $Z + A_{n-1}(Z) = x$.
2) Let $Z = x_1 \in [x, 1]$ be the (unique) solution of the equation $Z - A_{n-1}(Z) = x$.
3) Define $A_n(x) := \mathsf{H.M.}\left(A_{n-1}(x_0), A_{n-1}(x_1)\right)$, where $\mathsf{H.M.}(a, b) := 2ab/(a+b)$ is the harmonic mean of (a, b).
4) The protocol $\Pi_{\text{KMM}}(n, x)$ is recursively defined below.
   a) Define
   $$p_0 := \frac{A_{n-1}(x_1)}{A_{n-1}(x_0) + A_{n-1}(x_1)}, \text{ and}$$
   $$p_1 := 1 - p_0.$$
   b) Processor 1 either broadcasts 0 with probability p_0, or broadcasts 1 with probability p_1.
   c) If the first message is 0, then the remaining processors $\{2, 3, \ldots, n\}$ implement the protocol $\Pi_{\text{KMM}}(n-1, x_0)$. Otherwise, if the first message is 1, then the processors $\{2, 3, \ldots, n\}$ implement the protocol $\Pi_{\text{KMM}}(n-1, x_1)$.

The intuition underlying this recursive definition is illustrated by Fig. 3. In general, this geometric transformation, referred to as $T(\cdot)$, applies to any convex upwards function $C(X)$ with domain $[0, 1]$ and roots at 0 and 1. The transformed curve $T(C)$ is defined as follows. Consider Fig. 3, where one plots $C(X)$ instead of $A_{n-1}(X)$. The locus of the point marked by • defines the point $(x, T(C)(x))$, and, consequently, the transformed function $T(C)$.

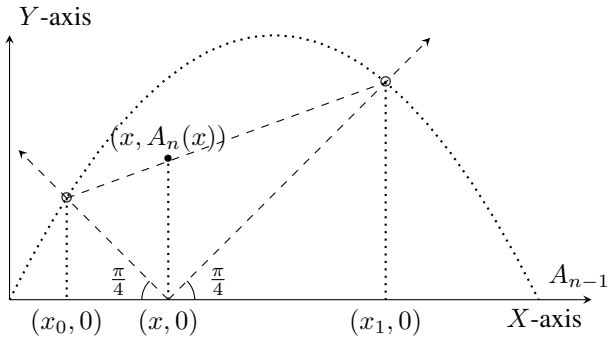

Fig. 3. A pictorial summary of the definition of curve A_n. Given curve A_{n-1}, this figure shows how A_n is defined at x. Probabilities p_0 and p_1 (in the definition of protocol $\Pi_{\text{KMM}}(n, x)$) are obtained by solving $p_0 + p_1 = 1$ and $p_0 x_0 + p_1 x_1 = x$.

We emphasize that $A_n(X)$ *upper-bounds* the insecurity of the protocol $\Pi_{\text{KMM}}(n, X)$. Note that the curves $A_n(X)$ are *not* quadratic except for base case $n = 1$. Furthermore, for general n, it seems likely that no elementary function can express the function $A_n(X)$. Given the protocol $\Pi_{\text{KMM}}(n, X)$, it is possible to compute its insecurity using exhaustive search, which is used to prepare Table I's entries.

IV. Relevant Prior Works

In this section, we highlight a few representative coin-tossing protocols for various security notions.

First, consider a *static* adversary who corrupts a processor before the protocol begins. When this particular processor is supposed to broadcast during the execution of the protocol, the adversary may restart it. In this case, the following protocol guarantees that the adversary can increase/decrease the expected outcome only by $\Theta(1/n)$.[3] Each processor broadcasts an independent bit that is 0 with probability c/n and 1 with probability $(1-c/n)$. The output of the protocol is the AND of all the broadcast bits. We choose c such that $(1-c/n)^n = 1/2$. This protocol, irrespective of the processor that the adversary corrupts, has insecurity $c/(2n)$.

Next, consider an *adaptive* adversary, one who decides whether to corrupt a processor based on all broadcast messages up to that point in time. Our work considers this model. Historically, majority protocol (threshold protocols,

[3]This insecurity is optimal up to a constant factor. Note that the expected outcome before the protocol begins is 1/2 and after the protocol ends is $\in \{0, 1\}$. Therefore, the expected outcome "jumps" by 1/2 (in the absolute value) during the protocol execution. Intuitively, by an averaging argument, there exists a processor whose message results in a jump of $\Omega(1/n)$ in the expected output. Therefore, a static adversary, by corrupting this processor, changes the output distribution by $\Omega(1/n)$.

Authorized licensed use limited to: Purdue University. Downloaded on June 08,2022 at 16:09:20 UTC from IEEE Xplore. Restrictions apply.

in general) are the only known protocol [5]–[7] that are $\Theta(1/\sqrt{n})$ insecure. They are also shown to be asymptotically optimal [4], [8] up to a constant factor for various other adversarial settings. However, whether the majority protocol is *the optimal* or asymptotically optimal protocol in our model remains unknown. Recently, Khorasgani et al. [8] show the existence of protocols whose insecurity has the potential to be lower than majority. The precise insecurity of their protocols is not well-understood. Moreover, their protocol is also not efficiently implementable.

Another exciting security model for coin-tossing protocols is where the adversary is *rushing*. For a rushing adversary, she gets to see every processor's message before deciding to intervene. This notion of security is motivated mainly by considering designing election schemes where changing one voter's vote has minimal effect on the result's overall outcome. The first setting that has been well-studied is when the adversary statically corrupts a processor and can arbitrarily set her broadcast message after observing all other processors' messages. This problem is the well-known problem of the *influence of variables on boolean functions*. When each processor broadcasts a uniformly random bit, the *tribes* function is asymptotically optimal protocol [9]. However, characterizing the exact optimal protocol in this setting remains open. Furthermore, the case where parties' messages can have arbitrary distribution is not well-understood [10], [11].

For static rushing adversaries, many works consider how many processors an adversary needs to corrupt to fix the output. The work of [9] shows that, for any function, an adversary needs to corrupt at most $\Theta\left(\frac{n}{\log n}\right)$ processors to completely bias the output. Ajtai and Linial [12] proved the existence of Boolean functions that are resilient to $\Theta\left(\frac{n}{\log^2 n}\right)$, which almost matches the upper bound. Their randomized construction was recently made explicit by Chattopadhyay and Zuckerman [13]. Finally, if one relaxes the setting such that processors can interact and send multiple messages, the upper bound of [9] no longer holds. The elegant "baton passing" protocol [14] and "lightest bin" protocol [15] are known to be resilient to up to αn corruptions, where $\alpha \in (0, 1/2)$.

In the model of adaptive rushing adversaries, Lichtenstein, Linial, and Saks [16] first showed that if every party sends a uniform bit, majority protocol (more generally, threshold protocols) achieves the optimal security. Kalai, Komargodski, and Raz [17] showed that when parties may send arbitrarily long messages, an adaptive rushing adversary can corrupt (at most) $\sqrt{n}\,\mathrm{polylog}\, n$ parties to fix the output completely. Very recently, Haitner and Karidi-Heller [18] extended this result to the setting where every party takes multiple turns to speak.

Lastly, Goldwasser, Kalai, and Park [3] proposed a strong adaptive adversary. That is, the adversary first sees every processor's message and, then, adaptively chooses which processor to corrupt to set its broadcast message arbitrarily. This security notion is closely related to the vertex isoperimetric inequalities [19] for the boolean hypercube.

V. Technical Overview

Our protocols are parametrized by an accuracy parameter $\delta \in (0, 1)$. Our objective is to maintain approximations $\widetilde{A}_{i,\delta}(X)$ to the curves $A_i(X)$, for each $1 \leqslant i \leqslant n$, satisfying the following invariant. Given the curve $\widetilde{A}_{i-1,\delta}(X)$, one can computationally efficiently (inductively) compute $\widetilde{A}_{i,\delta}(X)$ without significantly degrading the approximation quality. Among several possible similarity measures, the L-infinity norm ($\|\cdot\|_\infty$) seems an appropriate choice. For two curves C and D over the domain $[0, 1]$, define $\|C - D\|_\infty \leqslant \varepsilon$, if $|C(x) - D(x)| \leqslant \varepsilon$, for all $x \in [0, 1]$.

We approximate the curve $A_i(X)$ with a piece-wise linear curve $\widetilde{A}_{i,\delta}(X)$. For simplicity of presentation, assume that $1/\delta \in \mathbb{N}$. Our global invariant is that the piece-wise linear curve $\widetilde{A}_{i,\delta}(X)$, for $1 \leqslant i \leqslant n$, consists of line segments such that their X-projections are of length δ. Intuitively, the curve $\widetilde{A}_{i,\delta}(X)$ is implicitly defined by the samples stored as the following array

$$\widetilde{S}_{i,\delta} := \left[\widetilde{A}_{i,\delta}(0),\ \widetilde{A}_{i,\delta}(\delta),\ \widetilde{A}_{i,\delta}(2\delta), \ldots,\ \widetilde{A}_{i,\delta}(1)\right].$$

For the rest of the paper, we shall use $\widetilde{A}_{i,\delta}$ to be the piece-wise linear curve implicitly defined by the array $\widetilde{S}_{i,\delta}$.

For the base case $n = 1$, we know that $A_1(X) = X(1-X)$. So, for $u \in \{0, 1, \ldots, 1/\delta\}$, we generate the array

$$\widetilde{S}_{1,\delta}[u] := A_1(u \cdot \delta).$$

Next, for the inductive step $i > 1$, our objective is to compute the array $\widetilde{S}_{i,\delta}$ from the array $\widetilde{S}_{i-1,\delta}$. That is, compute the values of $\widetilde{S}_{i,\delta}[u] := T\left(\widetilde{A}_{i-1,\delta}\right)(u\delta)$, where $u \in \{0, 1, \ldots, 1/\delta\}$ (refer to Fig. 3 for the definition of the geometric transformation $T(\cdot)$). A crucial step in this inductive step is to compute $x_0 \in [0, u\delta]$ such that $x_0 + \widetilde{A}_{i-1,\delta}(x_0) = u\delta$. Towards this objective, one needs to identify the unique index $u_0 \in \{0, 1, \ldots, u\}$ such that

$$u_0\delta + \widetilde{S}_{i-1,\delta}[u_0] \leqslant u\delta$$
$$(u_0 + 1)\delta + \widetilde{S}_{i-1,\delta}[u_0 + 1] > u\delta$$

Once the index u_0 is identified, by linear interpolation, we obtain the value of x_0 and $\widetilde{A}_{i-1,\delta}(x_0)$.

The inductive step, also requires the computation of $x_1 \in [u\delta, 1]$ such that $x_1 - \widetilde{A}_{i-1,\delta}(x_1) = u\delta$. Since all the curves are symmetric about $X = 1/2$, we have $\widetilde{A}_{i-1,\delta}(x_1) = \widetilde{A}_{i-1,\delta}(1 - x_1)$. Therefore, solving the above equation is equivalent to solving $(1 - x_1) + \widetilde{A}_{i-1,\delta}(1 - x_1) = (1/\delta - u)\delta$, which is identical to the problem of obtaining x_0.

Given $y_0 = \widetilde{A}_{i-1,\delta}(x_0)$, and $y_1 = \widetilde{A}_{i-1,\delta}(x_1)$, we define

$$\widetilde{S}_{i,\delta}[u] := T(\widetilde{A}_{i-1,\delta})(u\delta) = \mathsf{H.M.}(y_0, y_1) = 2y_0y_1/(y_0 + y_1).$$

The curve $\widetilde{A}_{i,\delta}$ is implicitly defined by the linearly interpolating between the points $\left(v\delta, \widetilde{S}_{i,\delta}[v]\right)$ and $\left((v + 1)\delta, \widetilde{S}_{i,\delta}[v + 1]\right)$, where $v \in \{0, 1, \ldots, 1/\delta - 1\}$.

Authorized licensed use limited to: Purdue University. Downloaded on June 08,2022 at 16:09:20 UTC from IEEE Xplore. Restrictions apply.

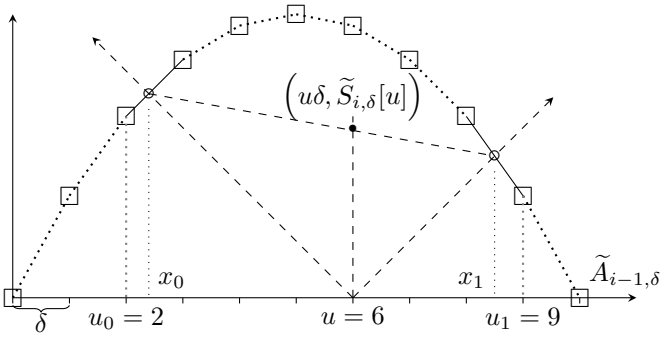

Fig. 4. Suppose $\delta = 1/10$. Assume that we already have the piece-wise linear curve $\widetilde{A}_{i-1,\delta}$ available. The figure shows the computation of $\widetilde{S}_{i,\delta}[u]$ for the representative value of $u = 6$.

We remark that the (piece-wise linear) curve $\widetilde{A}_{i,\delta}$ is *not* identical to the curve $T\left(\widetilde{A}_{i-1,\delta}\right)$. However, we shall prove that the curve $\widetilde{A}_{i,\delta}$ is indeed *close* to the curve $T\left(\widetilde{A}_{i-1,\delta}\right)$.

A. Technical Proof Outlines

This section highlights the primary technical lemmas needed to prove that the approximation does not degrade significantly during the recursive computations in our protocol.

Lemma 1 (Transformation of close curves is close): Let C and D be two convex upwards functions over the domain $[0, 1]$ with roots at 0 and 1. Then, the following inequality holds.

$$\|T(C) - T(D)\|_\infty \leqslant \|C - D\|_\infty .$$

Intuitively, if the curves C and D are already close, then their respective T-transform curves are also close.

Next, we need to define a new operator. Let C be a function over the domain $[0, 1]$. A δ-*sample* of C is the array

$$[C(0), C(\delta), C(2\delta), \ldots, C(1)] .$$

Let $\mathsf{Sample\&Linearize}\,(C, \delta)$ be the piece-wise linear curve defined by linearly interpolating these sampled points.

Lemma 2 (Linearizing does not harm significantly): Let C be a convex upwards function over the domain $[0, 1]$. Suppose that, for any point $x \in [0, 1]$, the line tangent to C at $(x, C(x))$ has slope $\in [-1, 1]$. Then, the following bound holds.

$$\|C - \mathsf{Sample\&Linearize}\,(C, \delta)\|_\infty \leqslant \delta/2.$$

The approximation quality of our algorithm follows from these results.

1) Note that by Lemma 2, we have $\left\|A_1 - \widetilde{A}_{1,\delta}\right\|_\infty \leqslant \delta/2$, because $\widetilde{A}_{1,\delta} = \mathsf{Sample\&Linearize}\,(A_1, \delta)$.
2) Suppose we already have the guarantee that $\left\|A_{i-1} - \widetilde{A}_{i-1,\delta}\right\|_\infty \leqslant \rho$. Then, by Lemma 1, we have $\left\|A_i - T\left(\widetilde{A}_{i-1,\delta}\right)\right\|_\infty \leqslant \rho$, because $A_i = T(A_{i-1})$. For the next step, note that we have the following identity

$$\widetilde{A}_{i,\delta} = \mathsf{Sample\&Linearize}\left(T\left(\widetilde{A}_{i-1,\delta}\right), \delta\right).$$

Therefore, by Lemma 2 and the triangle inequality, we conclude that $\left\|A_i - \widetilde{A}_{i,\delta}\right\|_\infty \leqslant \rho + \delta/2$.

From the above observations, inductively, it follows that, for $i \geqslant 1$,

$$\left\|A_i - \widetilde{A}_{i,\delta}\right\|_\infty \leqslant i\delta/2.$$

We remark that the exact inductive proof maintains additional invariants that the curves $\widetilde{A}_{i,\delta}$ are convex upwards, symmetric around $X = 1/2$, and their slope $\in [-1, +1]$. We forego these subtleties in favor of highlighting only the primary ideas underlying the proof. We refer the readers to the full version of this paper for more technical details.

B. Running-time Optimizations

For $i > 1$, the computation of u_0 corresponding to every $u \in \{0, 1, \ldots, 1/\delta\}$ can be optimized. Our objective is to compute the array $\mathsf{ptr}_{i-1}[u] = u_0$ establishing this mapping, for all u. Note that this mapping is non-decreasing. Therefore, for each $i \geqslant 2$, one can compute this entire mapping in $\Theta(1/\delta)$ time.

C. Our Protocol

The message of processor 1 in the protocol $\Pi_{\text{Our}}(n, X, \delta)$ is defined as follows.

1) Let $Z = x_0 \in [0, x]$ be the (unique) solution of the equation $Z + \widetilde{A}_{n-1,\delta}(Z) = x$.
2) Let $Z = x_1 \in [x, 1]$ be the (unique) solution of the equation $Z - \widetilde{A}_{n-1,\delta}(Z) = x$.
3) Define
$$p_0 := \frac{\widetilde{A}_{n-1,\delta}(x_1)}{\widetilde{A}_{n-1,\delta}(x_0) + \widetilde{A}_{n-1,\delta}(x_1)}, \text{ and}$$
$$p_1 := 1 - p_0.$$
4) Processor 1 either broadcasts 0 with probability p_0, or broadcasts 1 with probability p_1.

If the first message is 0, then the remaining processors $\{2, 3, \ldots, n\}$ implement the protocol $\Pi_{\text{Our}}(n-1, x_0, \delta)$. Otherwise, if the first message is 1, then the processors $\{2, 3, \ldots, n\}$ implement the protocol $\Pi_{\text{Our}}(n-1, x_1, \delta)$.

Precomputation. Processors can precompute the arrays $\widetilde{S}_{1,\delta}, \ldots, \widetilde{S}_{n,\delta}$ and the pointer arrays $\mathsf{ptr}_1, \ldots, \mathsf{ptr}_{n-1}$ in an offline precomputation step. After that, the next-message generation takes only constant time.

References

[1] M. Ben-Or and N. Linial, "Collective coin flipping, robust voting schemes and minima of banzhaf values," in *26th FOCS*. IEEE Computer Society Press, Oct. 1985, pp. 408–416. 1

[2] M. Ben-Or and N. Linial, "Collective coin flipping," *Advances in Computing Research*, 1989. 1

[3] S. Goldwasser, Y. T. Kalai, and S. Park, "Adaptively secure coin-flipping, revisited," in *ICALP 2015, Part II*, ser. LNCS, M. M. Halldórsson, K. Iwama, N. Kobayashi, and B. Speckmann, Eds., vol. 9135. Springer, Heidelberg, Jul. 2015, pp. 663–674. 1, 4

[4] R. Cleve and R. Impagliazzo, "Martingales, collective coin flipping and discrete control processes (extended abstract)," 1993. 1, 4

Authorized licensed use limited to: Purdue University. Downloaded on June 08,2022 at 16:09:20 UTC from IEEE Xplore. Restrictions apply.

[5] M. Blum, "How to exchange (secret) keys (extended abstract)," in *15th ACM STOC*. ACM Press, Apr. 1983, pp. 440–447. 1, 2, 4

[6] B. Awerbuch, M. Blum, B. Chor, S. Goldwasser, and S. Micali, "How to implement bracha's o (log n) byzantine agreement algorithm," *Unpublished manuscript*, 1985. 1, 2, 4

[7] R. Cleve, "Limits on the security of coin flips when half the processors are faulty (extended abstract)," in *18th ACM STOC*. ACM Press, May 1986, pp. 364–369. 1, 2, 4

[8] H. A. Khorasgani, H. K. Maji, and T. Mukherjee, "Estimating gaps in martingales and applications to coin-tossing: Constructions and hardness," in *TCC 2019, Part II*, ser. LNCS, D. Hofheinz and A. Rosen, Eds., vol. 11892. Springer, Heidelberg, Dec. 2019, pp. 333–355. 1, 2, 3, 4

[9] J. Kahn, G. Kalai, and N. Linial, "The influence of variables on Boolean functions (extended abstract)," in *29th FOCS*. IEEE Computer Society Press, Oct. 1988, pp. 68–80. 4

[10] J. Bourgain, J. Kahn, G. Kalai, Y. Katznelson, and N. Linial, "The influence of variables in product spaces," *Israel Journal of Mathematics*, vol. 77, no. 1-2, pp. 55–64, 1992. 4

[11] Y. Filmus, L. Hambardzumyan, H. Hatami, P. Hatami, and D. Zuckerman, "Biasing Boolean functions and collective coin-flipping protocols over arbitrary product distributions," in *ICALP 2019*, ser. LIPIcs, C. Baier, I. Chatzigiannakis, P. Flocchini, and S. Leonardi, Eds., vol. 132. Schloss Dagstuhl, Jul. 2019, pp. 58:1–58:13. 4

[12] M. Ajtai and N. Linial, "The influence of large coalitions," *Combinatorica*, vol. 13, no. 2, pp. 129–145, 1993. 4

[13] E. Chattopadhyay and D. Zuckerman, "Explicit two-source extractors and resilient functions," in *48th ACM STOC*, D. Wichs and Y. Mansour, Eds. ACM Press, Jun. 2016, pp. 670–683. 4

[14] M. E. Saks, "A robust noncryptographic protocol for collective coin flipping," *SIAM J. Discrete Math.*, vol. 2, no. 2, pp. 240–244, 1989. 4

[15] U. Feige, "Noncryptographic selection protocols," in *40th FOCS*. IEEE Computer Society Press, Oct. 1999, pp. 142–153. 4

[16] D. Lichtenstein, N. Linial, and M. Saks, "Some extremal problems arising from discrete control processes," *Combinatorica*, vol. 9, no. 3, pp. 269–287, 1989. 4

[17] Y. Tauman Kalai, I. Komargodski, and R. Raz, "A lower bound for adaptively-secure collective coin-flipping protocols," in *32nd International Symposium on Distributed Computing (DISC 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018. 4

[18] I. Haitner and Y. Karidi-Heller, "A tight lower bound on adaptively secure full-information coin flip," *FOCS*, 2020. 4

[19] L. H. Harper, "Optimal numberings and isoperimetric problems on graphs," *Journal of Combinatorial Theory*, vol. 1, no. 3, pp. 385–393, 1966. 4

Authorized licensed use limited to: Purdue University. Downloaded on June 08,2022 at 16:09:20 UTC from IEEE Xplore. Restrictions apply.