Optimally-secure Coin-tossing against a Byzantine Adversary

Hamidreza Amini Khorasgani  
Department of Computer Science, Purdue University, USA  
haminikh@purdue.edu

Hemanta K. Maji  
Department of Computer Science, Purdue University, USA  
hmaji@purdue.edu

Mingyuan Wang  
Department of Computer Science, Purdue University, USA  
wang1929@purdue.edu

Abstract

In their seminal work, Ben-Or and Linial (1985) introduced the full information model for collective coin-tossing protocols involving \( n \) processors with unbounded computational power using a common broadcast channel for all their communications. The design and analysis of coin-tossing protocols in the full information model have close connections to diverse fields like extremal graph theory, randomness extraction, cryptographic protocol design, game theory, distributed protocols, and learning theory. Several works have focused on studying the asymptotically best attacks and optimal coin-tossing protocols in various adversarial settings. While one knows the characterization of the exact or asymptotically optimal protocols in some adversarial settings, for most adversarial settings, the optimal protocol characterization remains open. For the cases where the asymptotically optimal constructions are known, the exact constants or poly-logarithmic multiplicative factors involved are not entirely well-understood.

In this work, we study \( n \)-processor coin-tossing protocols where every processor broadcasts an arbitrary-length message once. Note that, in this setting, which processor speaks and its message distribution may depend on the messages broadcast so far. An adaptive Byzantine adversary, based on the messages broadcast so far, can corrupt \( k = 1 \) processor. A bias-\( X \) coin-tossing protocol outputs 1 with probability \( X \); 0 with probability \( 1 - X \). For a coin-tossing protocol, its insecurity is the maximum change in the output distribution (in the statistical distance) that an adversarial strategy can cause. Our objective is to identify optimal bias-\( X \) coin-tossing protocols with minimum insecurity, for every \( X \in [0,1] \).

Lichtenstein, Linial, and Saks (1989) studied bias-\( X \) coin-tossing protocols in this adversarial model under the highly restrictive constraint that each party broadcasts an independent and uniformly random bit. The underlying message space is a well-behaved product space, and \( X \in [0,1] \) can only be integer multiples of \( 1/2^n \), which is a discrete problem. The case where every processor broadcasts only an independent random bit admits simplifications, for example, the collective coin-tossing protocol must be monotone. Surprisingly, for this class of coin-tossing protocols, the objective of reducing an adversary’s ability to increase the expected output is equivalent to reducing an adversary’s ability to decrease the expected output. Building on these observations, Lichtenstein, Linial, and Saks proved that the threshold coin-tossing protocols are optimal for all \( n \) and \( k \).

In a sequence of works, Goldwasser, Kalai, and Park (2015), Kalai, Komargodski, and Raz (2018), and (independent of our work) Haitner and Karidi-Heller (2020) prove that \( k = O(\sqrt{n} \cdot \text{polylog}(n)) \) corruptions suffice to fix the output of any bias-\( X \) coin-tossing protocol. These results consider parties who send arbitrary-length messages, and each processor has multiple turns to reveal its entire message. However, optimal protocols robust to a large number of corruptions do not have any apriori relation to the optimal protocol robust to \( k = 1 \) corruption. Furthermore, to make an informed choice of employing a coin-tossing protocol in practice, for a fixed target...
tolerance of insecurity, one needs a precise characterization of the minimum insecurity achieved by these coin-tossing protocols.

We rely on an inductive approach to constructing coin-tossing protocols to study a proxy potential function measuring the susceptibility of any bias-$X$ coin-tossing protocol to attacks in our adversarial model. Our technique is inherently constructive and yields protocols that minimize the potential function. It happens to be the case that threshold protocols minimize the potential function. We demonstrate that the insecurity of these threshold protocols is 2-approximate of the optimal protocol in our adversarial model. For any other $X \in [0,1]$ that threshold protocols cannot realize, we prove that an appropriate (convex) combination of the threshold protocols is a 4-approximation of the optimal protocol.

**Keywords and phrases** Multi-party Coin-tossing, Adaptive Adversaries, Byzantine adversary, Optimal Protocols

**Funding** The research effort is supported in part by an NSF CRII Award CNS–1566499, an NSF SMALL Award CNS–1618822, the IARPA HECTOR project, MITRE Innovation Program Academic Cybersecurity Research Award, a Purdue Research Foundation (PRF) Award, and The Center for Science of Information, an NSF Science and Technology Center, Cooperative Agreement CCF–0939370.

**Introduction**

In a seminal work, Ben-Or and Linial [BL85, BL89] introduced the full information model to study collective coin-tossing protocols. One relies on collective coin-tossing protocols to upgrade local private randomness of each of the $n$ processors into shared randomness that all processors agree. In this model, all the processors have unbounded computational power and communicate with each other over one broadcast channel. This model for the design and analysis of coin-tossing protocols turns out to be highly influential with close connections with diverse topics in mathematics and computer science, for example, extremal graph theory [Kru63, Kat68, Har66], extracting randomness from imperfect sources [SV84, CGH+85, Vaz85, Fri92], cryptography [CI93, DLMM11, DMM14, HOZ16, KMM19, KW20a, MW20], game theory [Bl64, Col71], circuit representation [Win71, OS08, OS11], distributed protocols [Asp97, Asp98, BJB98], and poisoning and evasion attacks on learning algorithms [DMM18, MDM19, MM19, EMM20].

A bias-$X$ $n$-processor coin-tossing protocol is an interactive protocol where every complete transcript is publicly associated with output 0 or 1, and the expected output for an honest execution of the protocol is $X \in [0,1]$. Given a bias-$X$ $n$-processor coin-tossing protocol $\pi$ and model for adversarial corruption and attack, let $\epsilon^+(\pi) \in [0,1]$ represent the maximum increase in the expected output that an adversarial strategy can cause. Similarly, let $\epsilon^-(\pi) \in [0,1]$ represent the maximum decrease in the expected output caused by an adversarial strategy. One defines the insecurity of a protocol $\pi$ as $\epsilon(\pi) := \max\{\epsilon^+(\pi), \epsilon^-(\pi)\}$. For a fixed $X \in [0,1]$, the optimal bias-$X$ $n$-processor protocol minimizes $\epsilon(\pi)$ among all bias-$X$ $n$-processor coin-tossing protocols.

For practical applications, given the tolerance for insecurity, one needs precise guarantees on the insecurity of coin-tossing protocols to estimate the necessary number of processors to keep the insecurity acceptably low. If the insecurity estimates for the potential coin-tossing protocols involve large latent constants or poly-logarithmic factors, then such a decision needs to be overly pessimistic in calculating the necessary number of processors. Consequently, it is essential to characterize coin-tossing protocols that are optimal or within a small constant
factor of the optimal protocol for every pair \((n, X)\). We emphasize that this outrightly rules out asymptotic bounds involving \(n\). This work contributes to this endeavor.

We study \(n\)-processor coin-tossing protocols where every processor broadcasts a message exactly once (i.e., single-turn), and there are \(n\) rounds, i.e., every round a unique processor broadcasts her message. The distribution of the messages sent by processors prescribed to speak in one round may depend on the messages sent in the previous rounds. For example, in one-round protocols, the distribution over the message space of the coin-tossing protocol is a product space. On the other hand, in single-turn \(n\)-round protocols, only one processor speaks in a round, and her message distribution possibly depends on all previously broadcast messages. Furthermore, which processor speaks in which round may depend on the messages sent in the previous rounds. We consider adaptive Byzantine adversaries who can corrupt \(k = 1\) processor, i.e., based on the evolution of the protocol, our adversary can corrupt one processor and fix her message arbitrarily. As is standard in cryptography, our adversary is always rushing, i.e., it can arbitrarily schedule all those processors who are supposed to speak in a round.

Variants this model have been studied, and we highlight, in the sequel, some of the most prominent works and their technical highlights.

**Lichtenstein, Linial, and Saks [LLS89].** Lichtenstein et al. [LLS89] consider the restriction where the \(i\)-th processor broadcasts an independent and uniformly random bit \(x_i\), where \(1 \leq i \leq n\), and the adversary can corrupt up to \(k\) processors, where \(1 \leq k \leq n\). The coin-tossing protocol is a function \(f: \{0, 1\}^n \rightarrow \{0, 1\}\). In this case, the underlying message space is \(\{0, 1\}^n\), which is a product space involving a small-size alphabet, and the probability distribution induced by the transcript is the uniform distribution over the message space. Note that, for \(n\)-processor coin-tossing protocols, the bias of such a protocol can only be an integral multiple of \(2^{-n}\). Therefore, this is a discrete optimization problem.

Given \(n\), \(k\), and \(X\), they begin with the objective of minimizing only the quantity \(\epsilon^+(\pi)\) over bias-\(X\) \(n\)-processor coin-tossing protocols \(\pi\). Their recursive characterization of the protocol that minimizes \(\epsilon^+\), incidentally, turns out to be identical to the optimal solution for the vertex isoperimetric inequality over the Boolean hypercube [Kru63, Kat68, Har66]. Therefore, a threshold protocol\(^1\) \(\pi\) is the optimal protocol and minimizes \(\epsilon^+\). The complementary protocol, which swaps the outputs 0 and 1 of \(\pi\), is also a threshold protocol and, consequently, minimizes \(\epsilon^-\). So, threshold protocols simultaneously minimize \(\epsilon^+\) and \(\epsilon^-\) and achieve optimal security.

**Significantly altering the output distribution.** For symmetric functions (i.e., permuting the inputs of the function \(f\) does not change its output), Goldwasser, Kalai, and Park [GKP15] prove that \(k = O(\sqrt{n} \cdot \text{polylog}(n))\) corruptions suffice to completely fix the output of any coin-tossing protocol even if the protocol relies on arbitrary-length messages. After that, Kalai, Komargodski, and Raz [KKR18] remove the restriction of symmetric functions. Recently, in independent work\(^2\), Haitner and Karidi-Heller [HK20] extend this result to multi-turn coin-tossing protocols. These papers use global analysis techniques for martingales that are inherently non-constructive; consequently, they prove the optimality of threshold protocols up to \(O(\text{polylog}(n))\) factors when the adversary corrupts at most \(k = O(\sqrt{n} \cdot \text{polylog}(n))\) processors.

**Challenge for arbitrary-length messages.** Our objective is to provide tight insecurity

---

\(^1\) More generally, protocols that output 1 for all strings smaller in the simplicial order than a threshold string are the optimal protocols.

\(^2\) A preliminary version of our work appears as [KMW20b].
estimates for the optimal coin-tossing protocols that use arbitrary-length messages. Let us understand why the technical approach of [LLS89] fails; and an entirely new approach is needed. In the full information model, without the loss of generality, one can assume that all interactive protocols are stateless, and processors use a fresh block of private randomness to generate the next message at any point during the evolution of the coin-tossing protocol [Jer85, JVV86, BGP00]. Furthermore, the security of the internal state of processors is not a concern, so, without loss of generality, every processor broadcasts its appropriate block of randomness whenever it speaks. A Byzantine adversary can corrupt a processor and arbitrarily set its randomness. So, for an appropriately large alphabet $\Sigma$, which depends on the randomness complexity of generating each message, our message space is $\Sigma^n$, a product space involving a large alphabet set. Over such product spaces, the isolated objective of minimizing $\epsilon^+$ does not entail the simultaneous minimization $\epsilon^-$. Given any $n \in \mathbb{N}$ and $X \in [0, 1]$, there exist protocols with $(\epsilon^+, \epsilon^-) = (1 - X^n, X^n)$, when the adversary can corrupt $k = 1$ processor (refer to Appendix A for the protocols). More generally, for product spaces over large alphabets, one does not expect such a vertex isoperimetric inequality [FHH99]. Finally, global analysis techniques of [GKP15, KKR18, HK20] analyze the case of a large number of corruptions $k$. The optimally secure protocol for $k = 1$ is not apriori related to the optimal protocols robust to a large number of corruptions. Furthermore, the inductive proof technique of Aspnes [Asp97, Asp98] is agnostic of the expected output of the coin-tossing protocol. Consequently, reconstructing the optimal protocol from the lower-bound on insecurity is not apparent.

We follow the geometric technique of Khorasgani, Maji, and Mukherjee [KMM19], which is inherently constructive, to obtain tight estimates of the optimally secure protocols.

Connection to Isoperimetric Inequalities

The connection to isoperimetric inequalities [Kru63, Kat68, Har66, Har99] (via the expansion of fixed density subset of product spaces) establishes the relevance to topics in theoretical computer science like expander graphs, complexity theory, and error-correcting codes.

Encoding Security of Coin-tossing Protocols. Every coin-tossing protocol is equivalent to a unique subset $S$ of an $n$-dimension product space $\Sigma^n$, where the size of the alphabet set $\sigma := |\Sigma|$ depends on the randomness complexity of the coin-tossing protocol. The elements of this product space represent the complete transcript of the coin-tossing protocol. The $i$-th coordinate of an element corresponds to the message sent by processor $i$, and the subset $S$ contains all elements of the product space on which the coin-tossing protocol outputs 1. One considers the uniform distribution over $\Sigma^n$ to sample the elements. This subsection considers a stronger Byzantine adversary who can edit one processor’s message after seeing the message of all processors.

The discussion in this subsection extends to arbitrary corruption threshold $k$. However, for the simplicity of the presentation, we consider the specific case of $k = 1$. Let $\partial S_k^+$ be the set of elements in $\overline{S}$ (the complement of $S$) that are at a Hamming distance $k = 1$ from the set $S$. Consequently, the strong Byzantine adversary can change an element from the set $\partial S_k^+ \subseteq \overline{S}$ into some element of $S$ by editing (at most) $k$ coordinates. Note that if the

---

3 Let $\pi$ be the original coin-tossing protocol. In the compiled $\pi'$, suppose parties reveal the block of randomness that they use to prepare their next-message in the protocol $\pi'$. The new protocol $\pi'$, first, emulates the next-message function of $\pi$ to generate the entire transcript, and, then, uses $\pi$ to determine the output.
stronger Byzantine adversary can see all the messages and then performs the edits, then it can increase the expected output by exactly $\epsilon^+ = \frac{|\partial S^+_k|}{\sigma^n}$.

Analogously, one defines the set $\partial S^-_k \subseteq S$ that contains all elements at a Hamming distance $k = 1$ from the set $\overline{S}$. So, a stronger Byzantine adversary can reduce the expected output by $\epsilon^- = \frac{|\partial S^-_k|}{\sigma^n}$.

**Extremal Graph Theory Perspective.** The (width-$k$) vertex perimeter of the set $S$, represented by $\partial_{N,k}S$, is the set of all elements in $\overline{S}$ that are at a Hamming distance of at most $k$ from some element in $S$. Observe that the perimeter $\partial_{V,k}S$ is identical to the set $\partial S^+_k$. Similarly, the vertex perimeter of the set $\overline{S}$ (which is $\partial_{V,k}\overline{S}$) is identical to the set $\partial S^-_k$.

The objective of extremal graph theory is to characterize the optimal set $S$ of a density-$\Sigma$ that minimizes its vertex perimeter. This optimal set $S$, in turn, characterizes the bias-$X$ coin-tossing protocol with the minimum $\epsilon^+$. In Appendix A, we saw that minimizing $\epsilon^+$ does not automatically entail the simultaneous minimization of $\epsilon^-$ for general $\Sigma$. In fact, that example highlighted that the protocol minimizing $\epsilon^+$ resulted in a protocol where the stronger Byzantine adversary can force the outcome 0 with certainty. Therefore, there is a disconnect between the cryptographic objective of simultaneously minimizing $\epsilon = \max\{\epsilon^+, \epsilon^-\}$ with the standard objective in extremal graph theory for large alphabet set $\Sigma$.

**Cryptography-inspired Extremal Graph Theory.** Instead of minimizing the vertex perimeter of a density-$\Sigma$ set $S$, one should consider the alternative objective of minimizing the symmetric perimeter of $S$ defined under various norms.

$$\partial_{V,k,\ell}(S) := \left(\left|\partial_{V,k}S\right|^\ell + \left|\partial_{V,k}\overline{S}\right|^\ell\right)^{1/\ell}.$$ 

The $\ell = \infty$ case corresponds to our cryptographic objective; however, this norm is difficult to analyze. Consequently, we study the norm $\ell = 1$ as a proxy, which is a 2-approximation of the norm $\ell = \infty$. Our results provide evidence that such symmetric perimeters may be more well-behaved in general.

Recall that, in our setting, the element in $\Sigma^n$ is exposed one coordinate at a time and our Byzantine adversaries cannot go back to edit previously exposed coordinates. So, our Byzantine adversaries have lesser power than the stronger Byzantine adversaries considered in this section. Consequently, the minimum achievable insecurity for bias-$X$ $n$-processor coin-tossing protocols in our setting lower-bounds the proxy norm above. For instance, when $\ell = 1$, our results imply that the density of the symmetric perimeter is $1/\sqrt{n}$ for any dense set $S$, irrespective of the size of the alphabet set.

**Remark.** We identify a density-$X$ set with its corresponding bias-$X$ coin-tossing protocol. Using the independent bounded differences inequality for the Hamming distance function (using Azuma’s inequality [Azu67]) on a constant-density subset $S$ implies that $k = O(\sqrt{n})$ edits suffice to achieve any constant $\epsilon^+$ and $\epsilon^-$, for any $\sigma$. However, for small $k$ (for example, $k = 1$), obtaining meaningful guarantees on both $\epsilon^+$ and $\epsilon^-$ is not possible for large $\sigma$. On the other hand, interestingly, we shall show that $\max\{\epsilon^+, \epsilon^-\} \geq 1/\sqrt{n}$ for any $\sigma$. This result lends support to the hypothesis that the symmetric perimeter is more well-behaved.

### 1.1 Our Contributions

Any $n$-processor coin-tossing protocol $\pi$ is equivalent to a depth-$n$ tree, where each node $v$ corresponds to a partial transcript. For every leaf of this tree, one associates the output of

---

\[4\] For $\Sigma = \{0, 1\}$, this entailment holds; otherwise, it is not known to hold in general.
the coin-tossing protocol $\in \{0, 1\}$. For a partial transcript $v$, the *color* of $v$, represented by $x_v$, represents the expected output of the coin-tossing protocol conditioned on the partial transcript being $v$. For example, the leaves have color $\in \{0, 1\}$, and the color of the root of a bias-$X$ coin-tossing protocol is $X$. The probability $p_v$ represents the probability that the partial transcript $v$ is generated during the protocol evolution of $\pi$.

A Byzantine adversary, in this interpretation of a coin-tossing protocol, that corrupts at most $k = 1$ processor is equivalent to a prefix-free set of edges. That is, for any two edges $(u, v)$ and $(u', v')$ such that $u$ is the parent of $v$ and $u'$ is the parent of $v'$, the root to leaf path through $u$ does not pass through $u'$. Any such collection of edges corresponds to a unique Byzantine adversarial strategy. For example, if an edge $(u, v)$ lies in this set and $u$ is the parent of $v$, then this edges indicates that the Byzantine adversary decides to interfere when the protocol generates the partial transcript $u$, and this adversary sends the next message that generates the partial transcript $v$. Note that the partial transcript $u$ uniquely identifies the processor that the adversary needs to corrupt.

Let $\tau$ be one such attack strategy. Suppose $\tau$ is a collection of $\ell$ edges, namely, $\{(u_i, v_i)\}_{i=1}^\ell$. Assume $u_i$ is the parent of $v_i$, for $i = 1, \ldots, \ell$. Then, we define the score of the attack strategy $\tau$ on protocol $\pi$ as

$$\text{Score}(\pi, \tau) := \sum_{i=1}^{\ell} p_{u_i} \cdot |x_{u_i} - x_{v_i}|.$$  

The term $\text{Score}(\pi, \tau)$ represents the vulnerability of protocol $\pi$ under attack strategy $\tau$. Furthermore, we define

$$\text{Score}(\pi) := \sup_{\tau} \text{Score}(\pi, \tau).$$

Intuitively, $\text{Score}(\pi)$ represents the insecurity of the protocol under the most devastating attack, a.k.a., our potential function.

We emphasize that our score is not identical to the deviation in output distribution that a Byzantine adversary causes. It is a 2-approximation of that quantity. Define the insecurity as the maximum change that a Byzantine adversary can cause to the output distribution. Then, it is evident that the insecurity of $\pi$ is at least $\text{Score}(\pi)/2$.

For an arbitrary $n \in \mathbb{N}^*$ and $t \in \{0, 1, \ldots, n+1\}$, let $\pi^{n,t}$ denote the $n$-processor $t$-threshold threshold protocol. In this threshold protocol, every processor broadcasts an independent and uniformly random bit. The output of this threshold protocol is 1 if and only if the total number of ones in the complete transcript is $\geq t$. An $n$-processor $t$-threshold protocol has color $2^{-n} \cdot \left(\sum_{i=t}^{n} \binom{n}{i}\right)$.

We prove the following theorem about the threshold protocol.

**Theorem 1.** For any bias $X$ $n$-processor protocol $\pi$, where $X = 2^{-n} \cdot \left(\sum_{i=t}^{n} \binom{n}{i}\right)$, where $0 \leq t \leq n + 1$, then

$$\text{Score}(\pi^{n,t}) \leq \text{Score}(\pi).$$

That is, the threshold protocol is the protocol that minimizes the score. Equivalently, the insecurity of the threshold protocol is a 2-approximation of the optimal insecurity in our corruption model (refer to Corollary 1).

Furthermore, we also prove the following result. Suppose $X$ is not a root-color that admits a threshold protocol, and $X_0$ is inverse-polynomially far from both 0 and 1. Suppose $X$ is intermediate to the bias of the threshold protocols $\pi^{n-1,t}$ and $\pi^{n-1,t-1}$. Let $\pi$ be a protocol where the first processor decides to run the threshold protocol $\pi^{n-1,t}$ or $\pi^{n-1,t-1}$ with suitable probability so that the resulting protocol is a bias-$X$ protocol. Then, the insecurity of this protocol $\pi$ is a 4-approximation of the protocols with minimum insecurity against Byzantine adversaries (refer to Corollary 2).
1.2 Prior Works

In this section, we summarize results in the full information model. It is beyond the scope of this paper to cover coin-tossing results in the computational setting like [Blu82, Cle86, ABC+85, MNS09, BOO10, AO16, BHLT17, BHMO18].

**Static corruption.** The case of static corruption is well-understood. In this setting, given a coin-tossing protocol, the adversary has to corrupt the processors before the beginning of the protocol. There is a close relation of this literature to results in randomness extraction [SV84, CGH+85, Vaz85, Fri92], game theory [BI64, Col71], and circuit representation [Win71, OS08, OS11]. Over the years, constructions of coin-tossing protocols were introduced that were robust to \( k = O(n^{0.63}) \) corruptions [BL85, BL89], \( k = O(n/\log^2 n) \) corruptions [AL93, CZ16], \( k = O(n/\log n) \) corruptions [Sak89], and \( k = (1/2 - \delta)n \) [AN90, BN93, Fei99] (for any positive constant \( \delta \)).

On the other hand, the seminal work of Kahn, Kalai, and Linial [KKL88] proves that \( k = \Omega(n/\log n) \) corruptions suffice to completely fix the output of a coin-tossing protocol where every message of a processor is a single bit. In fact, robustness to \( k = \Omega(n) \) corruption necessitates multi-bit messages or super-constant number of rounds [RSZ99].

**Adaptive corruption.** For adaptive Byzantine adversaries, Ben-Or and Linial [BL85, BL89] showed that majority protocol is resilient to \( O(\sqrt{n}) \) corruptions, and they conjectured this protocol is asymptotically optimal. The case of adaptive corruption where the adversary sees everyone’s messages before intervening is closely related to the vertex isoperimetric problem over the Boolean hypercube [Kru63, Kat68, Har66]. Threshold protocols are optimal for this adversarial model, for arbitrary corruption threshold \( k \). Dodis [Dod00] proved that robustness to \( k = O(\sqrt{n} \cdot \text{polylog}(n)) \) corrupts suffice to fix the outcome of any single-turn coin-tossing protocol. Recently, independent of our work, Haitner and Karidi-Heller [HK20] extend this bound even for multi-turn protocols.

The constructions closest to our problem are the works of Lichtenstein, Linial, and Saks [LLS89], which characterized the optimal coin-tossing protocol for all \( n \in \mathbb{N} \) and corruption threshold \( k \leq n \) for coin-tossing protocols that are single-turn, \( n \)-round, adaptive Byzantine adversaries, and each processor sends one-bit uniformly random bit. Subsequently, Goldwasser, Kalai, and Park [GKP15] and Kalai, Konargodski, and Raz [KK15, KKR18] prove that \( k = O(\sqrt{n} \cdot \text{polylog}(n)) \) corrupts suffice to fix the outcome of any single-turn coin-tossing protocol. Recently, independent of our work, Haitner and Karidi-Heller [HK20] extend this bound even for multi-turn protocols.

Aspnes [Asp97, Asp98] considered the case where an adaptive adversary, if it does not like the message set by a particular processor, kills that processor. Other processors detect this event and move forward with the protocol assuming a placeholder message for that processor. This model of attack is very closely related to the strong adversary model introduced by Goldwasser, Kalai, and Park [GKP15].

**Constructive potential-based approaches.** Recently, in the field of fair coin-tossing, Khorasgani, Maji, and Mukherjee [KMM19] introduced the approach of geometric transformation for designing optimal protocols. They showed that this approach yields protocols with less susceptibility than the majority protocols [Blu82, Cle86]. Subsequently, this approach has also been used to obtain new black-box separation results for fair coin-tossing protocols [KMW20a, MW20], which settled a longstanding open problem regarding the optimality of the protocol of Blum [Blu82] and Cleve [Cle86] that uses one-way functions in a black-box manner.
1.3 Technical Overview

The techniques closest to our approach are those introduced by Aspnes [Asp97, Asp98] and Khorasgani et al. [KMM19, KMW20a, MW20].

Aspnes’ technique [Asp97, Asp98] tracks the locus of all possible \((\epsilon^+, \epsilon^-)\) corresponding to any \(n\)-processor \(k\)-corruption threshold protocol. However, the information regarding the root-color is lost and, consequently, the technique does not yield the optimal protocol construction. Next, one lower-bounds this space using easy-to-interpret (hyperbolic) curves and obtains bounds on the insecurity of any \(n\)-processor protocol with \(k\) corruption threshold (against adversaries who erase the messages of processors).

The technique of Khorsganiet al. [KMM19, KMW20a, MW20] use a potential function as a proxy to study the actual problem at hand. They maintain the locus of all \(n\)-processor bias-\(X\) protocols that minimize the potential function. Next, they inductively build the next curve of \((n + 1)\)-processors bias-\(X\) protocols that minimize the potential function. Their approach outrightly yields optimal constructions that minimize the potential function, and easily handle the case of processors sending arbitrary-length messages.

**High-level summary of our approach.** We use the potential function as introduced in Section 1.1, which is a 2-approximation of the optimal insecurity against Byzantine adversaries, for any \(n\)-processor bias-\(X\) protocol. Let \(C_n(X)\) represent the minimum realizable potential for bias-\(X\) \(n\)-processor coin-tossing protocols.

Next, we prove that if an \(n\)-processor threshold protocol has potential \(\delta\) and bias-\(X\), then the point \((\delta, X)\) lies on the optimal curve \(C_n(X)\). Therefore, the potential of these threshold protocols are 2-approximation of the optimal bias-\(X\) protocol against Byzantine adversaries.

After that, inductively, we prove that the linear interpolation of the set of points \((\delta, X)\) realized by \(n\)-processor threshold protocols with potential \(\delta\) and root-color \(X\), where \(0 \leq t \leq n + 1\), is a lower-bound to the actual curve \(C_n(X)\). Finally, we argue that a linear interpolation of appropriate threshold functions yields a protocol with potential that is 4-approximation of the optimal protocol against Byzantine adversaries.

**The curves and the inductive transformation.** Consider the case of \(n = 1\) and arbitrary bias-\(X\). If \(X = 0\) or \(X = 1\), then we have \(C_1(X) = 0\). If \(X \in (0, 1/2]\), then we include that edge that sets the output to 1. This observation creates a potential of \(C_1(X) = 1 - X\). Similarly, we have \(C_1(X) = X\), for all \(X \in [1/2, 1)\). Our characterization of the curve \(C_1(X)\) is complete (refer to Figure 2).

Next, consider the case of \(n = 2\) and bias-\(X\). This case is sufficient to understand how to inductively build the locus of the curve \(C_{n+1}(X)\) inductive from \(C_n(X)\). Consider any arbitrary 2-processor bias-\(X\) coin-tossing protocol. Suppose the first processor sends message 1, 2, \ldots, \(\ell\). Let \(x_i\), for \(1 \leq i \leq \ell\), be the expected output conditioned on the first message being \(i\). At the root of this protocol, we have two options. Corrupt processor one and send the message that achieves the highest potential. Or, defer the intervention to a later point in time.

Corrupting the root of this protocol causes the potential to become

\[
\max_{i=1}^{\ell} |X - x_i|.
\]

Deferring the intervention to a later point in time results in the potential becoming at least

\[
\sum_{i=1}^{\ell} p_i \cdot C_1(x_i),
\]

where \(p_i\) is the probability that processor 1 outputs \(i\). The actual potential of \(\pi\) is the
maximum of these two quantities. Our objective is to characterize the choice of \( x_1, \ldots, x_\ell \) such that the potential is minimized (refer to Figure 1).

2 Preliminaries

We use \( \mathbb{N}^* \) for the set of positive integers. For any two curves \( C_1, C_2 \) defined on \([0,1]\), we write \( C_1 \preceq C_2 \) (\( C_1 \) is below \( C_2 \)) to denote that \( C_1(x) \leq C_2(x) \) for each \( x \in [0,1] \). A curve \( C \) defined on \([0,1]\), is called concave if for all \( 0 \leq x < y \leq 1 \), and any \( \alpha \in [0,1] \), we have \( C(\alpha x + (1-\alpha)y) \geq \alpha C(x) + (1-\alpha)C(y) \). Statistical distance between two distributions \( A \) and \( B \) defined over discrete sample space \( \Omega \) is defined as \( \text{SD}(A,B) := \frac{1}{2} \sum_{x \in \Omega} |A(x) - B(x)| \).

A function \( f: \mathbb{N} \rightarrow \mathbb{R} \) is called negligible if for any polynomial \( p(n) \), \( f(n) = o(1/p(n)) \).

2.1 Coin-tossing Protocols

In this work, we consider coin-tossing protocols among \( n \) processors in the full information model. That is, all processors communicate through one single broadcast channel. In particular, we consider an \( n \)-round protocol. At round \( i \), the \( i^{th} \) processor will broadcast a (random) message based on the first \( i-1 \) broadcast messages. After every processor broadcasts her messages, the final output \( \in \{0,1\} \) is a deterministic function of all the broadcast messages. We do not limit to protocols with unbiased output (i.e., the probability of the output being 1 is 1/2).

▶ Definition 1 ((n, \( X_0 \))-Coin-tossing protocols). For any \( n \in \mathbb{N}^* \) and \( X_0 \in [0,1] \), an \((n, X_0)\)-coin-tossing protocol is an \( n \)-round coin-tossing protocol among \( n \) processors, where the expectation of the output is \( X_0 \).

We often refer to the expected output \( X_0 \) as the color of the protocol. The insecurity of a coin-tossing protocol is the maximum change (in terms of statistical distance) that the adversary can cause to the distribution of the output of the protocol.

In this work, threshold protocols will be very useful examples, which are defined as follows.

▶ Definition 2 ((n, \( t \))-Threshold protocol). In an \((n, t)\)-threshold protocol, denoted by \( \pi_{n,t} \), each processor broadcasts an (independently) uniform bit. The output is 1 if the total number of 1-message \( \geq t \). In particular, when \( n \) is odd and \( t = \frac{n+1}{2} \), this is the majority protocol.

2.2 Adversarial Setting

In this work, we consider Byzantine adaptive adversaries. Such an adversary will eavesdrop on the execution of the protocol. After every round, it will decide whether to corrupt the processor, who is going to speak next. Once a processor is corrupted, the adversary takes full control and fixes the message that she is going to send. We will focus on such adversaries that corrupt (at most) one processor.

3 A Geometric Perspective

In this section, we shall study the insecurity of coin-tossing protocols through a geometric perspective.

5 Here, \( t \in \{0, 1, \ldots, n+1\} \).
Protocol tree. For every coin-tossing processor protocol, we will think of it as a tree. Every edge represents a message, and the root denotes the beginning of the protocol. Therefore, every node $u$ on this tree represents a partial transcript of the protocol. And we can associate it with a color $x_u$ and a probability $p_u$, where $x_u$ is the expected output conditioned on partial transcript $u$, and $p_u$ is the probability that partial transcript $u$ happens. For an $(n, X_0)$-coin-tossing protocol, by our definition, its protocol tree shall have depth $n$, and the color at the root shall be $X_0$.

Attack. A Byzantine adaptive adversary that corrupts at most one processor can be viewed equivalently as a collection of edges $\{ (u_i, v_i) \}$, where $u_i$ is the parent of $v_i$. This implies that when partial transcript $u_i$ happens, the attacker intervenes and fixes the next message to be $v_i$. Since this attacker corrupts at most one processor during the entire collection of the protocol, this collection of edges must be prefix-free. That is, no parent node of an edge is on the path from the root to other edges.

Given a protocol tree $\pi$, let an attack strategy $\tau$ be the collection of edges $\{ (u_i, v_i) \}$, where $u_i$ is the parent of $v_i$. We define the following score function.

\[ \text{Score}(\pi, \tau) := \sum_{(u_i, v_i) \in \tau} p_u \cdot |x_u - x_v|. \]

That is, $\text{Score}(\pi, \tau)$ is the average of the absolute change in color the attacker $\tau$ causes. Intuitively, it represents the vulnerability of protocol $\pi$ in the presence of the attack $\tau$. Furthermore, for any protocol $\pi$, let us define

\[ \text{Score}(\pi) := \sup_{\tau} \text{Score}(\pi, \tau). \]

Intuitively, $\text{Score}(\pi)$ represents the score of the most devastating attacks on protocol $\pi$. Finally, we define

\[ C_n(X_0) := \inf_{\pi} \text{Score}(\pi), \]

where the infimum is taken over all $(n, X_0)$-coin-tossing protocols $\pi$. Intuitively, $C_n(X_0)$ represents the score of the optimal protocol against the most devastating attack among all protocols with $n$ processors and color $X_0$.

Remark 1. We remark that for a protocol $\pi$, the deviation (to the distribution of the output) an attack $\tau$ causes is not exactly $\text{Score}(\pi, \tau)$. However, one can always bi-partition the set $\tau$ as $\tau_0$ and $\tau_1$. $\tau_0$ will consist of all edges $(u_i, v_i)$ that decrease the expected output, i.e., $x_u \geq x_v$, while $\tau_1$ will consist of all edges $(u_i, v_i)$ that increase the expected output, i.e., $x_u < x_v$. Consequently, the summation of the deviations caused by attack $\tau_0$ and $\tau_1$ shall be $\text{Score}(\pi, \tau)$. Therefore, there must exist an attack that deviates the protocol by $\text{Score}(\pi, \tau)/2$. In light of this, for any $(n, X_0)$-coin-tossing protocol, there must exist an attack that deviates the protocol by $C_n(X_0)/2$. Hence, any $(n, X_0)$-coin-tossing protocol is (at least) $C_n(X_0)/2$ insecure.

3.1 Geometric Transformation of $C_n$

In this section, we shall see how we can (inductively) construct $C_n$ from a geometric perspective.

Let us start with the simplest case $n = 1$. If $X_0 = 0$ or 1, the output is independent of the message and is always fixed. Hence, the score is always 0. If $X_0 \in (0, 1/2]$, the attack with the highest score is to fix the message such that the output is fixed to be 1. Hence, the score is $1 - X_0$. Similarly, when $X_0 \in (1/2, 1)$, the score is $X_0$. Consequently, $C_1$ is the
The optimal protocol shall, however, pick the above quantity. Therefore, by our definition, the highest score, which is

\[ p \cdot \max (|x - x_1|, |x - x_2|, |x - x_3|, p_1 \cdot C_n(x_1) + p_2 \cdot C_n(x_2) + p_3 \cdot C_n(x_3)) \]

Let \( \pi \) be an \((n+1, x)\)-coin-tossing protocol. Suppose there are three possible messages that the first processor might send, namely \( m_1, m_2, \) and \( m_3 \). Conditioned on the first message being \( m_1, m_2, \) and \( m_3 \), the expected output is \( x_1, x_2, \) and \( x_3 \), respectively. The probability of the first message being \( m_1, m_2, \) and \( m_3 \) are \( p_1, p_2, \) and \( p_3 \), respectively. Note that after the first processor sends message \( m_i \), the remaining protocol \( \pi_i \) becomes a \((n, x_i)\)-coin-tossing protocol.

An adaptive adversary that corrupts at most one processor has four choices for the first processor. Either it can carry out the attack now by fixing the first processor’s message to be \( m_i \), for \( i \in \{1, 2, 3\} \), or it can defer the attack to subprotocols \( \pi_1, \pi_2, \) and \( \pi_3 \). If it fixes the first processor’s message to be \( m_i \), this will increase the score by \(|x_i - x|\). On the other hand, if it defers the attack to each subprotocol, by the definition of curve \( C_n \), it can ensure a score of (at least) \( C_n(x_i) \) in subprotocol \( \pi_i \). Overall, it ensures a score of (at least)

\[ p_1 \cdot C_n(x_1) + p_2 \cdot C_n(x_2) + p_3 \cdot C_n(x_3). \]

Note that it must hold that \( x = p_1 x_1 + p_2 x_2 + p_3 x_3 \). Therefore, \( p_1 \cdot C_n(x_1) + p_2 \cdot C_n(x_2) + p_3 \cdot C_n(x_3) \) must lie between \( y_1 \) and \( y_2 \) in Figure 1.

The most devastating attack will do the attack based on which strategy results in the highest score, which is

\[ \max (|x - x_1|, |x - x_2|, |x - x_3|, p_1 \cdot C_n(x_1) + p_2 \cdot C_n(x_2) + p_3 \cdot C_n(x_3)) \]

The optimal protocol shall, however, pick \( x_1, \ldots, x_\ell \) and \( p_1, \ldots, p_\ell \) accordingly to minimize the above quantity. Therefore, by our definition,

\[ C_{n+1}(x) := \inf_{x_1, \ldots, x_\ell \in [0,1]} \max_{p_1, \ldots, p_\ell \in [0,1], p_1 + \cdots + p_\ell = 1, \sum_{i=1}^\ell p_i x_i = x} \left( |x - x_1|, \ldots, |x - x_\ell|, \sum_{i=1}^\ell p_i \cdot C_n(x_i) \right). \]
For convenience, let us define geometric transformation $T$, which takes any curve $C$ on $[0, 1]$ as input, and outputs a curve $T(C)$ defined as

$$T(C)(x) := \inf_{x_1, \ldots, x_{\ell} \in [0, 1]} \max \left( |x - x_1|, \ldots, |x - x_{\ell}|, \sum_{i=1}^{\ell} p_i \cdot C(x_i) \right).$$

Hence, by our definition, $C_{n+1}$ is exactly $T(C_n)$.

4 Tight Bounds on $C_n$ and the Implications

In this section, we shall first prove a tight lower bound on the curve $C_n$.

We define our lower bound curve $L_n$ through threshold protocols. Recall that an $(n, t)$-threshold protocol $\pi^{n,t}$ is a protocol where each processor broadcast an (independent) uniform bit. The final output is 1 if the number of 1-message is $\geq t$. Trivially, the color of $(n, t)$-threshold protocol $\pi^{n,t}$ is

$$\text{Color}(\pi^{n,t}) = 2^{-n} \cdot \left( \sum_{i=t}^{n} \binom{n}{i} \right).$$

We argue that the score of $\pi^{n,t}$ is

$$\text{Score}(\pi^{n,t}) = 2^{-n} \cdot \left( \frac{n-1}{t-1} \right).$$

To see this, note that, without of loss of generality, we can assume that anytime the adversary fixes a message, it fixes that message to be 1. Moreover, which message that the adversary fixes does not matter; effectively, the output will be 1 if and only if the rest $n-1$ messages contain $\geq t-1$ 1-message. Therefore, by fixing one message to be 1, it changes the expected output of the protocol to be $2^{-(n-1)} \cdot \left( \sum_{i=t-1}^{n-1} \binom{n-1}{i} \right)$. Easily, one can verify that

$$2^{-(n-1)} \cdot \left( \sum_{i=t-1}^{n-1} \binom{n-1}{i} \right) - 2^{-n} \cdot \left( \sum_{i=t}^{n} \binom{n}{i} \right) = 2^{-n} \cdot \binom{n-1}{t-1}.$$

For a $n$-processor threshold protocol, threshold $t \in \{n+1, n, \ldots, 0\}$. We define the lower bound curve $L_n$ as follows.

Definition 4. For every $n \in \mathbb{N}^*$, let $L_n$ be the curve that linearly connects points

$$P_{n,t} := (\text{Color}(\pi^{n,t}), \text{Score}(\pi^{n,t})) = \left( 2^{-n} \cdot \left( \sum_{i=t}^{n} \binom{n}{i} \right), 2^{-n} \cdot \left( \frac{n-1}{t-1} \right) \right)$$

for $t = n+1, n, \ldots, 0$. That is, $L_n$ linearly interpolates all the points defined by the color and score of $(n, t)$-threshold protocols.

As an example, $L_1$ is shown in Figure 2.

In particular, we have the following theorem regarding the curve $L_n$ and the curve $C_n$.

Theorem 2. For all $n \in \mathbb{N}^*$, $L_n \preceq C_n$.

---

6 For any node $u$, let its two children node be $v_0$ and $v_1$. Since every message is a uniform bit for threshold protocol, it must hold that $|x_u - x_{v_0}| = |x_u - x_{v_1}|$. Therefore, whether the attack picks edge $(u, v_0)$ or $(u, v_1)$ does not change the score.

7 When $t = n+1$, the color is 0, and when $t = 0$, the color is 1.
Figure 2 The (black) dashed curve is $L_1$ and the (blue) solid curve is $C_1$. Note that $P_{1,t}$ corresponds to the point defined by $(1,t)$-threshold protocol.

Remark 2. Note that, by the definition of $C_n$, we have

$$C_n(\text{Color}(\pi_{n,t})) := \inf_{\pi} \text{Score}(\pi) \leq \text{Score}(\pi_{n,t}).$$

On the other hand, by Theorem 2,

$$C_n(\text{Color}(\pi_{n,t})) \geq L_n(\text{Color}(\pi_{n,t})) = \text{Score}(\pi_{n,t}).$$

Therefore, $C_n(\text{Color}(\pi_{n,t})) = \text{Score}(\pi_{n,t})$. That is, points $P_{n,t}$ is on the curve $C_n$ as well. This also implies that threshold protocol is the protocol that minimizes the score function.

We defer the proof of Theorem 2 to Section 4.1. Let us first discuss the implications of this theorem. We have the following corollaries.

Corollary 1 (Threshold protocols). For any $n \in \mathbb{N}^*$ and $X_0 \in [0,1]$ such that $X_0 = 2^{-n} \cdot (\sum_{t=1}^{n} \binom{n}{t})$ for some $t \in \{0,1,\ldots,n+1\}$. The insecurity of $(n,t)$-threshold protocol is at most two times the insecurity of the least insecure $(n,X_0)$-coin-tossing protocols.

This corollary is immediate from Theorem 2. This is because the insecurity of threshold protocol $\pi_{n,t}$ is exactly $\text{Score}(\pi_{n,t})$; for any other $(n,\text{Color}(\pi_{n,t}))$-coin-tossing protocol, in light of Remark 1, we know its insecurity is at least

$$C_n(\text{Color}(\pi_{n,t})) / 2 \geq L_n(\text{Color}(\pi_{n,t})) / 2 = \text{Score}(\pi_{n,t}) / 2.$$  

Therefore, the insecurity of the threshold protocol is at most two times the insecurity of the optimal protocol.

Corollary 2 (Non-threshold protocols). For an arbitrary color $X_0 \in (0,1)$ that does not correspond to any threshold protocol, we can consider a linear combination of threshold protocols. Specifically, suppose $\text{Color}(\pi_{n,t}) < X_0 < \text{Color}(\pi_{n,t-1})$, consider an $(n+1,X_0)$-coin-tossing protocol as follows. The first processor sends a bit. If this bit is 0, the rest $n$ processors execute the $(n,t)$-threshold protocol; if this bit is 1, the rest $n$ processors execute the $(n,t-1)$-threshold protocol. The probability of this bit being 0 is defined to be

$$\frac{\text{Color}(\pi_{n,t-1}) - X_0}{\text{Color}(\pi_{n,t-1}) - \text{Color}(\pi_{n,t})}.$$ 

For $X_0$ that is not negligibly close to 0 or 1, the insecurity of this protocol is at most $4 + o(1)$ times the insecurity of the least insecure $(n+1,X_0)$-protocol.
Without loss of generality, assume $X_0 < 1/2$. Therefore, $t > n/2$. One can easily see that 
the insecurity of this protocol is bounded by 
$$\max \left( \text{Color} \left( \frac{n^{t-1}}{t} \right) - \text{Color} \left( \frac{n^t}{t} \right), \frac{\text{Score} \left( \frac{n^{t-1}}{t} \right) + \text{Score} \left( \frac{n^t}{t} \right)}{2} \right),$$
which is bounded by $2^{-n} \cdot \left( \begin{array} {c} n \\ t \end{array} \right)$. On the other hand, Theorem 2 says that every $(n+1, X_0)$-coin-tossing protocol is at least $L_{n+1}(X_0)/2$-insecure, which is at least $2^{-(n+2)} \left( \begin{array} {c} n+1 \\ t \end{array} \right)$. When $X_0$ is non-negligibly bounded away from 0 and 1, by Chernoff’s bound, we must have $|t - n/2| \leq \sqrt{n} \log n$. Consequently, $\left( \begin{array} {c} n \\ t \end{array} \right)$ and $\left( \begin{array} {c} n+1 \\ t \end{array} \right)$ are $(1 + o(1))$ approximation to each other. Hence, the insecurity of this protocol is (at most) $(4+o(1))$-approximate of the optimal $(n+1, X_0)$-protocol.

### 4.1 Proof of Theorem 2

To prove this theorem, it suffices to prove the following claims.

- **Claim 1.** If $A \leq B$, then $T(A) \leq T(B)$.
- **Claim 2.** $L_{n+1} = T(L_n)$.

**Proof of Theorem 2 using Claim 1 and Claim 2.** We prove this theorem inductively. The base case $n = 1$ is trivial (See Figure 2).

Suppose the statement is correct for $n$, i.e., $L_n \preceq C_n$. Then we have

$$L_n \preceq C_n \xrightarrow{\text{Claim 1}} T(L_n) \preceq T(C_n) \xrightarrow{\text{Claim 2}} L_{n+1} \preceq C_{n+1}$$

This completes the proof.

Next we prove Claim 1 and Claim 2.

**Proof of Claim 1.** Since $A \leq B$, for all $x, x_1, \ldots, x_\ell$, and $p_1, \ldots, p_\ell$, we have

$$\max \left( |x - x_1|, \ldots, |x - x_\ell|, \sum_{i=1}^{\ell} p_i \cdot A(x_i) \right) \leq \max \left( |x - x_1|, \ldots, |x - x_\ell|, \sum_{i=1}^{\ell} p_i \cdot B(x_i) \right).$$

Therefore, by definition, for all $x$, $T(A)(x) \leq T(B)(x)$, or equivalently $T(A) \preceq T(B)$.

Before we prove Claim 2, the following claim will be useful.

- **Claim 3.** Let $U$ be an arbitrary concave curve. Suppose $0 \leq x_0 < x < x_2 \leq 1$ satisfies that $x - x_0 = x_1 - x = \frac{U(x_0) + U(x_1)}{2}$, then $T(U)(x) = \frac{U(x_0) + U(x_1)}{2}$. That is, $x_0$ and $x_1$ witness the transformation $T$ of $U$ at $x$.

**Proof of Claim 3.** To see this, let us use Figure 3 for intuition. In Figure 3, $U(x)$ is a concave curve and the choice of $x_0$ and $x_1$ satisfies that $x - x_0 = x_1 - x = \frac{U(x_0) + U(x_1)}{2}$. Recall that
Figure 3 The geometric transformation of curve $U(x)$. Intuitively, if $x'_1$, $x'_2$, and $x'_3$ are in $(x_0, x_1)$, the shaded region is always above line segment $AB$ by the concavity of $U$.

$$T(U)(x) := \inf_{l = 1}^{\ell} \max \left( |x - x'_1|, \ldots, |x - x'_\ell|, \sum_{i=1}^{\ell} p_i \cdot D(x'_i) \right).$$

By definition, clearly, $T(U)(x) \leq \frac{U(x_0) + U(x_1)}{2}$. To prove the other direction, we need to show that, for any choices of $x'_1, x'_2, \ldots, x'_\ell$ and $p_1, p_2, \ldots, p_\ell$, we have

$$\frac{U(x_0) + U(x_1)}{2} \leq \max \left( |x - x'_1|, \ldots, |x - x'_\ell|, \sum_{i=1}^{\ell} p_i \cdot U(x'_i) \right)$$

Firstly, if there exists an $x'_i$ such that $|x - x'_i| \geq |x_1 - x|$, then the statement trivially holds. Next, if for all $i$, $|x - x'_i| \leq |x_1 - x|$, then by the concavity of curve $U$,

$$\frac{1}{2} (U(x_1) + U(x_2)) \leq \sum_{i=1}^{\ell} p_i \cdot U(x'_i).$$

This completes the proof.

Now, we prove Claim 2.

Proof of Claim 2. Recall that $L_n$ is the curve that linearly connects points $P_{n,n+1}, P_{n,n}, \ldots, P_{1,1}, P_{0,0}$, where

$$P_{n,t} := \left( 2^{-n} \cdot \left( \sum_{i=t}^{n} \binom{n}{i} \right), \ 2^{-n} \cdot \left( \binom{n-1}{t-1} \right) \right).$$

Let us first observe some properties of $L_n$.

Claim 4. $L_n$ is a concave curve and the slope of any line segment of $L_n$ is in $[-1, 1]$.

Proof of Claim 4. Easily, we can verify that the slope of line segment $P_{n,t}P_{n,t-1}$ is

$$\frac{2^{-n} \cdot \binom{n-1}{t-1} - 2^{-n} \cdot \binom{n-1}{t-2}}{2^{-n} \cdot \left( \sum_{i=t}^{n} \binom{n}{i} \right) - 2^{-n} \cdot \left( \sum_{i=t-1}^{n} \binom{n}{i} \right)} = \frac{2t - 2 - n}{n}.$$  

Since the slope of $P_{n,t}P_{n,t-1}$ decreases as $t$ decreases, this proves that $L_n$ is concave. Moreover, for any $t \in \{n+1, \ldots, 1\}$, the slope of $P_{n,t}P_{n,t-1}$ is in $[-1, 1]$. 

\clearpage
One can verify that this is indeed correct.

Hence, by Claim 3, it suffices to set

Now, let us prove

Next, we verify

We first verify it at

The relation between (black solid) \( L_n \) and (blue dashed) \( L_{n+1} \). The geometric transformation of \( L_n \) is exactly \( L_{n+1} \).

\( n \) is exactly \( \frac{2t-2-n}{n} \).

\( n \) and \( t \) is the middle point of \( P_{n,t} \) and \( P_{n,t-1} \).

Proof of Claim 5. One just need to verify that

and

Now, let us prove \( L_{n+1} = T(L_n) \) with all the claims that we have proven. It suffices to verify \( L_{n+1}(x) = T(L_n)(x) \) for all \( x \in (0, 1) \). In light of Claim 4 and Claim 5, we know the relation between \( L_n \) and \( L_{n+1} \) looks like Figure 4.

We first verify it at \( x = \text{Color}(\pi^{n+1,t}) \). In this case, we can set \( x_0 = \text{Color}(\pi^{n,t}) \) and \( x_1 = \text{Color}(\pi^{n,t-1}) \). One can verify that

and

Hence, by Claim 3,

Next, we verify \( L_{n+1} = T(L_n) \) for some \( x \) such that \( \text{Color}(\pi^{n+1,t+1}) < x < \text{Color}(\pi^{n+1,t}) \).

By Claim 3, it suffices to set \( x_0 = x - L_{n+1}(x) \) and \( x_1 = x + L_{n+1}(x) \) and verify that

Note that

One can verify that this is indeed correct.
References


18 Optimally-secure Coin-tossing against a Byzantine Adversary


Dod00 Yevgeniy Dodis. Impossibility of black-box reduction from non-adaptively to adaptively secure coin-flipping. Electronic Colloquium on Computational Complexity (ECCC), 7(39), 2000.


Kat68 G Katona. A theorem for finite sets, theory of graphs (p. erdős and g. katona, eds.), 1968.


KKR18 Yael Tauman Kalai, Ilan Komargodski, and Ran Raz. A lower bound for adaptively-secure collective coin-flipping protocols. In Ulrich Schmid and Josef Widder, editors, 32nd International Symposium on Distributed Computing, DISC 2018, New Or-


A Some Examples

**Figure 5** An example $n$-processor coin-tossing protocol that is easy to deviate toward 0, but hard to deviate toward 1. In this protocol, $X_k = X_0 + k \cdot \frac{1 - X_0}{n}$. Adversary can corrupt the first processor and achieve $\epsilon^+ = X_1 - X_0 = \frac{1 - X_0}{n}$ by setting its message to be 1 or achieve $\epsilon^- = X_0 - 0 = X_0$ by setting its message to be 0.

**Figure 6** An example $n$-processor coin-tossing protocol that is easy to deviate toward 1, but hard to deviate toward 0. In this protocol, $X_k = k \cdot \frac{X_0}{n}$. Adversary can corrupt the first processor and achieve $\epsilon^+ = 1 - X_0$ by setting its message to be 1 or achieve $\epsilon^- = X_1 - X_0 = \frac{X_0}{n}$ by setting its message to be 0.