Compressing Communication in Distributed Protocols

Yael Tauman Kalai, Ilan Komargodski

Compressing Communication in Distributed Protocols

Yael Tauman Kalai[1] and Ilan Komargodski[2⋆]

[1] Microsoft Research.
yael@microsoft.com
[2] Weizmann Institute of Science, Rehovot 76100, Israel.
ilan.komargodski@weizmann.ac.il

Abstract. We show how to compress communication in distributed protocols in which parties do not have private inputs. More specifically, we present a generic method for converting any protocol in which parties do not have private inputs, into another protocol where each message is *"short"* while preserving the same number of rounds, the same communication pattern, the same output distribution, and the same resilience to error. Assuming that the output lies in some universe of size M, in our resulting protocol each message consists of only $\mathsf{polylog}(M, n, d)$ many bits, where n is the number of parties and d is the number of rounds. Our transformation works in the full information model, in the presence of either static or adaptive Byzantine faults.

In particular, our result implies that for any such $\mathsf{poly}(n)$-round distributed protocol which generates outputs in a universe of size $\mathsf{poly}(n)$, long messages are not needed, and messages of length $\mathsf{polylog}(n)$ suffice. In other words, in this regime, any distributed task that can be solved in the $\mathcal{LOCAL}$ model, can also be solved in the $\mathcal{CONGEST}$ model with the *same* round complexity and security guarantees.

As a corollary, we conclude that for any $\mathsf{poly}(n)$-round collective coin-flipping protocol, leader election protocol, or selection protocols, messages of length $\mathsf{polylog}(n)$ suffice (in the presence of either static or adaptive Byzantine faults).

1 Introduction

In classical algorithmic design the goal is to design efficient algorithms, where the common complexity measures are time and space. In distributed algorithms, where a set of parties tries to perform a predefined task, there are more parameters of interest, such as round complexity, message complexity, fault-tolerance, and more.

⋆ Part of this work done while an intern at MSR New England. Supported in part by a grant from the I-CORE Program of the Planning and Budgeting Committee, the Israel Science Foundation, BSF and the Israeli Ministry of Science and Technology.

These measures have been studied in the literature under two main models: $\mathcal{LOCAL}$ and $\mathcal{CONGEST}$ [11]. The $\mathcal{LOCAL}$ model is aimed at studying "localized" executions of distributed protocols, and thus, messages of unlimited size are allowed. The $\mathcal{CONGEST}$ model is geared towards understanding the effect of congestion in the network, and thus, messages of poly-logarithmic size (in the number of parties) are allowed.[3]

Most of the work in distributed computing assumes one of the models above and focuses on optimizing resources such as round complexity, message complexity and fault-tolerance. We initiate the study of the following question:

Is there a generic way to transform protocols in the $\mathcal{LOCAL}$ model to protocols in the $\mathcal{CONGEST}$ model, without negatively affecting the round complexity, fault-tolerance and other resources?

We give a positive answer to this question for protocols in which parties do not have private inputs, without incurring *any* cost to the round complexity or the resilience to errors. More details follow.

Our model. In this work, our focus is on the synchronous, full information model. Namely, we consider a distributed model in which n parties are trying to perform a predefined task. Each party is equipped with a source of private randomness and a unique ID. We assume the existence of a global counter which synchronizes parties in between rounds, but the parties are asynchronous within each round. The goal is to fulfill the task even in the presence of Byzantine faults. In the full information model no restrictions are made on the computational power of the faulty parties or the information available to them. Namely, the faulty parties may be infinitely powerful, and we do not assume the existence of private channels connecting pairs of honest parties.

We model faulty parties by a computationally unbounded adversary who controls a subset of parties and whose aim is to bias the output of the protocol. We assume that the adversary has access to the entire transcript of the protocol, and once a party is corrupted, the adversary gains complete control over the party and can send any messages on its behalf, and the messages can depend on the entire transcript so far. In addition, we allow our adversary to be "rushing", i.e., it can schedule

[3] We note that often the term $\mathcal{CONGEST}$ is a short-hand writing for $\mathcal{CONGEST}(B)$, where B is a bandwidth constraint. In many cases, the convention is to set B to be bounded by $O(\log n)$, where n is the number of parties. Here, we take a more liberal interpretation, which allows for messages of size bounded by $\mathsf{polylog}(n)$ (see e.g., [14]).

the delivery of the messages within each round. We consider two classes of adversaries: *static* and *adaptive*. A static adversary is an adversary that chooses which parties to corrupt ahead of time, before the protocol begins. An adaptive adversary, on the other hand, is allowed to choose which parties to corrupt *adaptively* in the course of the protocol as a function of the messages seen so far.

The focus of this work, is on protocols in which parties do not have private inputs. Many classical distributed tasks fall in this category, including collective coin-flipping, leader election, selection and more.

A concrete motivation: adaptively-secure coin-flipping. An important distributed task that was extensively studied in the full information model, is that of *collective coin-flipping*. In this problem, a set of n parties use private randomness and are required to generate a common random bit. The goal of the parties is to jointly output a somewhat uniform bit even in the case that some of the parties are faulty and controlled by a static (resp. adaptive) adversary whose goal is to bias the output of the protocol in some direction.

This problem was first formulated and studied by Ben-Or and Linial [1]. In the case of static adversaries, collective coin-flipping is well studied and almost matching upper and lower bounds are known [2, 13], whereas the case of adaptive adversaries has received much less attention. Ben-Or and Linial [1] showed that the majority protocol (in which each party sends a uniformly random bit and the output of the protocol is the majority of the bits sent) is resilient to $\Theta(\sqrt{n})$ adaptive corruptions. Furthermore, they conjectured that this protocol is optimal, that is, they conjectured that any coin-flipping protocol is resilient to at most $O(\sqrt{n})$ adaptive corruptions. Shortly afterwards, Lichtenstein, Linial and Saks [8] proved the conjecture for protocols in which each party is allowed to send only *one* bit. Very recently, Goldwasser, Kalai and Park [4] proved a different special-case of the aforementioned conjecture: any *symmetric* (many-bit) one-round collective coin-flipping protocol[4] is resilient to at most $\widetilde{O}(\sqrt{n})$ adaptive corruptions. Despite all this effort, proving a general lower bound, or constructing a collective coin-flipping protocol that is resilient to at least $\omega(\sqrt{n})$ adaptive corruptions, remains an intriguing open problem.

The result of [8] suggests that when seeking for a collective coin-flipping protocol that is resilient to at least $\omega(\sqrt{n})$ adaptive corruptions,

[4] A symmetric protocol Π is one that is oblivious to the order of its inputs: namely, for any permutation $\pi\colon [n] \to [n]$ of the parties, it holds that $\Pi(r_1, \ldots, r_n) = \Pi(r_{\pi(1)}, \ldots, r_{\pi(n)})$.

to focus on protocols that consist of many communication rounds, or protocols in which parties send long messages. Our main result (Theorem 1) is that long messages are not needed in adaptively secure coin-flipping protocols with $\mathsf{poly}(n)$ rounds, and messages of length $\mathsf{polylog}(n)$ suffice. This is true more generally for leader election protocols, and for selection protocols where the output comes from a universe of size at most quasi-polynomial in n.

1.1 Our Results

Our main result is that "long" messages are not needed for distributed tasks in which parties do not have private inputs. More specifically, we show how to convert any n-party d-round protocol, where parties do not have private inputs, and whose output comes from a universe of size M, into a d-round protocol, with the same communication pattern, the same output distribution, the same security guarantees, and where each message is of length $\mathsf{polylog}(M, n, d)$. Note that for many well studied distributed tasks, such as coin-flipping, leader election, and more, the output is from a universe of size at most $\mathsf{poly}(n)$, in which case our result says that if we consider $\mathsf{poly}(n)$-round protocols, then messages of length $\mathsf{polylog}(n)$ suffice.

Our results in more detail. Formally, we say that a protocol Π, in which parties do not have private inputs, is (t, δ, s)-*statically (resp., adaptively) secure* if for any adversary $\mathcal{A}$ that *statically (resp., adaptively)* corrupts at most $t = t(n)$ parties, and any subset S of the output universe such that $|S| = s$, it holds that

$$\Big|\Pr\left[\text{Output of } \mathcal{A}(\Pi) \in S\right] - \Pr\left[\text{Output of } \Pi \in S\right]\Big| \leq \delta,$$

where "Output of $\mathcal{A}(\Pi)$" means the output of the protocol when executed in the presence of the adversary $\mathcal{A}$, "Output of Π" means the output of the protocol when executed honestly, and the probabilities are taken over the internal randomness of the parties. In addition, we say that a protocol Π *simulates* a protocol Π' if the outcomes of the protocols are statistically close (when executed honestly) and their communication patterns are the same.

Our main result is a generic communication compression theorem which, roughly speaking, states that (t, δ, s)-statically (resp., adaptively) secure protocols in the above model *do not* need "long" messages. Namely, we show that any secure protocol which sends arbitrary long messages

can be simulated by a protocol which is almost as secure and sends short messages.

Theorem 1 (Main theorem – informal). *Any (t, δ, s)-statically (resp., adaptively) secure d-round protocol that outputs m bits (or more generally, has an output universe of size 2^m), can be simulated by a d-round (t, δ', s)-statically (resp., adaptively) secure protocol, where $\delta' = \delta + \mathsf{negl}(n)$, and in which parties send random messages of length at most $m \cdot \mathsf{polylog}(n, d)$.*

Our results can also be seen as a transformation of protocols (in which parties do not have private inputs) in the $\mathcal{LOCAL}$ model to protocols in the $\mathcal{CONGEST}$ model, as discussed above. Our main theorem (Theorem 1) implies that any task, whose output consists of at most $\mathsf{polylog}(n)$ bits, and in which parties do not have private inputs, that can be solved in the $\mathcal{LOCAL}$ model with $d \leq \mathsf{poly}(n)$ rounds, can also be solved in the $\mathcal{CONGEST}$ model with d rounds.

Corollary 1. *Any n-party (t, δ, s)-statically (resp., adaptively) secure $\mathsf{poly}(n)$-round protocol that outputs $\mathsf{polylog}(n)$ bits in the $\mathcal{LOCAL}$ model, can be simulated by a (t, δ', s)-statically (resp., adaptively) secure protocol in the $\mathcal{CONGEST}$ model, where $\delta' = \delta + \mathsf{negl}(n)$.*

We emphasize that our results holds for any underlying communication pattern including the broadcast channel or the message-passing model with any underlying communication graph.

Finally, we note that the transformation in Theorem 1 preserves the computational efficiency of the honest parties, but the resulting protocol is *non-uniform*, even if the protocol we started with is uniform. We elaborate on this in Section 1.3.

1.2 Related Work

The resource of communication is central in several fields of computer science. The field of communication complexity is devoted to the study of which problems can be solved with as little communication as possible. We refer to the book of Kushilevitz and Nisan [6] for an introduction to the field. In cryptography, minimizing communication has been the focus of several works in several contexts, including private information retrieval [7], random access memory machines [9], and more.

Interestingly, in the setting of distributed computing most of the work focuses on optimizing other resources such as round complexity, fault-tolerance, and the quality of the outcome. Very few works focus on optimizing the maximal message length being sent during the protocols.

Moreover, most of the work in the literature focuses on *static* adversaries, and very few papers study distributed protocols with respect to *adaptive* adversaries. Our results hold in both settings.

Finally, we mention that separations between the $\mathcal{LOCAL}$ and $\mathcal{CONGEST}$ models are known for general tasks. For example, for network graphs of diameter $D = \Omega(\log n)$, computing the minimum spanning tree (MST) in the $\mathcal{LOCAL}$ model requires $\Theta(D)$ rounds, whereas in the $\mathcal{CONGEST}$ model every distributed MST algorithm has round complexity $\Omega(D + \sqrt{n}/\log^2 n)$ [12].

1.3 Overview of Our Techniques

In this section we provide a high-level overview of our main ideas and techniques. First, we observe that one can assume, without loss of generality, that any protocol in which parties do not have private inputs, can be transformed into a public-coin protocol, in which honest parties' messages consist only of random bits. This fact is a folklore, and for the sake of completeness we include a proof sketch of it in Section 4.

Our main result is a generic transformation that converts any public-coin protocol, in which parties send arbitrarily long messages, into a protocol in which parties send messages of length $m \cdot \mathsf{polylog}(n \cdot d)$, where m is the number of bits the protocol outputs, n is the number of parties participating in the protocol, and d is the number of communication rounds. The resulting protocol simulates the original protocol, has the same round complexity and satisfies the same security guarantees. Next, we elaborate on how this transformation works.

Suppose for simplicity that in our underlying protocol each message sent is of length $L = L(n)$ (and thus the messages come from a universe of size 2^L), and think of L as being very large. We convert any such protocol into a new protocol where each message consists of only ℓ bits, where think of ℓ as being significantly smaller than L. This is done by a priori choosing 2^ℓ messages within the 2^L-size universe, and restricting the parties to send messages from this restricted universe. Thus, now each message is of length ℓ, which is supposedly significantly smaller than L. We note that a similar approach was taken in [10] in the context of transforming public randomness into private randomness in communication complexity, in [3] to reduce the number of random bits needed for property testers, and most recently in [4] to prove a lower bound for coin-flipping protocols in the setting of strong adaptive adversaries.

A priori, it may seem that such an approach is doomed to fail, since by restricting the honest parties to send messages from a small universe

within the large 2^L-size universe, we give the adversary a significant amount of information about future messages (especially in the multi-round case). Intuitively, the reason security is not compromised is that there are *many* possible restrictions, and it suffices to prove that a few (or only one) of these restrictions is secure. In other words, very loosely speaking, since we believe that most of the bits sent by honest parties are not "sensitive", we believe that it is safe to post some information about each message ahead of time.

For the sake of simplicity, in this overview we focus on static adversaries, and to simplify matters even further, we assume the adversary always corrupts the first t parties. This simplified setting already captures the high-level intuition behind our security proof in Section 3.

Let us first consider one-round protocols. Note that for one-round protocols restricting the message space of honest parties does not affect security at all since we consider rushing adversaries, who may choose which messages to send based on the content of the messages sent by all honest parties in that round. Thus, reducing the length of messages is trivial in this case, assuming the set of parties that the adversary corrupts is predetermined. We mention that even in this extremely simplified setting, we need ℓ to be linear in m for correctness ("simulation"), i.e., in order to ensure that the output is distributed correctly.

Next, consider a multi-round protocol Π. We denote by H the restricted message space, i.e., H is a subset of the message universe of size 2^ℓ, and denote by Π_H the protocol Π, where the messages are restricted to the set H. Suppose that for any set H there exists an adversary $\mathcal{A}^H$ that biases the outcome of Π_H, say towards 0.[5] We show that in this case there exists an adversary $\mathcal{A}$ in the underlying protocol that biases the outcome towards 0. Loosely speaking, at each step the adversary $\mathcal{A}$ will simulate one of the adversaries $\mathcal{A}^H$. More specifically, at any point in the underlying protocol, the adversary will randomly choose a set H such that the transcript so far is consistent with a run of protocol Π_H with the adversary $\mathcal{A}^H$, and will simulate the adversary $\mathcal{A}^H$. The main difficulty is to show that with high probability there exists such H (i.e., the remaining set of consistent H's is non-empty). This follows from a counting argument and basic probability analysis.

In our actual construction, we have a distinct set H of size 2^ℓ corresponding to *each* message of the protocol. Thus, if the underlying pro-

[5] Of course, it may be that for different sets H, the adversary $\mathcal{A}^H$ biases the outcome to a different value. For simplicity we assume here that all the adversaries bias the outcome towards a fixed message, which we denote by 0.

tocol Π has d rounds, and all the parties send a message in each round, then the resulting (short-message) protocol is associated with $d \cdot n$ sets $H_1, \ldots, H_{d \cdot n}$ each of size 2^ℓ, where the message of the j^{th} party in the i^{th} round is restricted to be in the set $H_{i,j}$. We denote all these sets by a matrix $H \in \left(\{0,1\}^L\right)^{d \cdot n \times 2^\ell}$, where the row (i,j) of H corresponds to the set of messages that the j^{th} party can send during the i^{th} round.

Note that there are $2^{L \cdot 2^\ell \cdot d \cdot n}$ such matrices. Each time an honest party sends a uniformly random message in Π it reduces the set of consistent matrices by approximately a 2^L-factor (with high probability). Any time the adversary $\mathcal{A}$ sends a message, it also reduces the set of consistent matrices H, since his message is consistent only with some of the adversaries $\mathcal{A}^H$, but again a probabilistic argument can be used to claim that it does not reduce the set of matrices by too much, and hence, with high probability there always exist matrices H that are consistent with the transcript so far.

We briefly mention that the analysis in the case of adaptive corruptions follows the same outline presented above. One complication is that the mere decision of whether to corrupt or not reduces the set of consistent matrices H. Nevertheless, we argue that many consistent matrices remain.

We emphasize that the above is an over-simplification of our ideas, and the actual proof is more complex. We refer to Section 3 for more details.

2 Preliminaries

In this section we present the notation and basic definitions that are used in this work. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. For a distribution X we denote by $x \leftarrow X$ the process of sampling a value x from the distribution X. Similarly, for a set X we denote by $x \leftarrow X$ the process of sampling a value x from the uniform distribution over X. Unless explicitly stated, we assume that the underlying probability distribution in our equations is the uniform distribution over the appropriate set. We let $\mathbf{U}_L$ denote the uniform distribution over $\{0,1\}^L$. We use $\log x$ to denote a logarithm in base 2.

A function $\mathsf{negl}\colon \mathbb{N} \to \mathbb{R}$ is said to be *negligible* if for every constant $c > 0$ there exists an integer N_c such that $\mathsf{negl}(n) < n^{-c}$ for all $n > N_c$.

The *statistical distance* between two random variables X and Y over a finite domain Ω is defined as

$$\mathsf{SD}(X,Y) \triangleq \frac{1}{2}\sum_{\omega\in\Omega}\left|\Pr[X=\omega]-\Pr[Y=\omega]\right|. \tag{1}$$

The Model

The communication model and distributed tasks. We consider the *synchronous model* where a set of n parties $\mathsf{P}_1,\ldots,\mathsf{P}_n$ run protocols. Each protocol consists of *rounds* in which parties send messages. We assume the existence of a global counter which synchronizes parties in between rounds (but they are asynchronous within a round).

The focus of this work is on tasks where parties do not have any private inputs. Examples of such tasks are coin-flipping protocols, leader election protocols, Byzantine agreement protocols, etc.

Throughout this paper, we restrict ourselves to public-coin protocols.

Definition 1 (Public-coin protocols). *A protocol is* public-coin *if all honest parties' messages consist only of uniform random bits.*

Jumping ahead, we consider adversaries in the full information model. In Section 4 we argue that the restriction to public-coin protocols is without loss of generality since in the full information model any protocol (in which parties do not have private inputs) can be converted into a public-coin one, without increasing the round complexity and without degrading security (though this transformation may significantly increase the communication complexity).

The adversarial model. We consider the *full information model* where it is assumed the adversary is all powerful, and may see the entire transcript of the protocol. The most common adversarial model considered in the literature is the Byzantine model, where a bound $t = t(n) \leq n$ is specified, and the adversary is allowed to corrupt up to t parties. The adversary can see the entire transcript, has full control over all the corrupted parties, and can send any messages on their behalf. Moreover, the adversary has control over the order of the messages sent within each round of the protocol.[6] We focus on the Byzantine model throughout this work.

Within this model, two types of adversaries were considered in the literature: *static* adversaries, who need to specify the parties they corrupt

[6] Such an adversary is often referred to as "rushing".

before the protocol begins, and *adaptive* adversaries, who can corrupt the parties *adaptively* based on the transcript so far. Our results hold for both types of adversaries. Throughout this work, we focus on the adaptive setting, since the proof is more complicated in this setting. In Subsection 3.1 we mention how to modify (and simplify) the proof for the static setting.

Correctness and security. For any protocol Π and any adversary $\mathcal{A}$, we denote by

$$\mathsf{out}(\mathcal{A}_\Pi \mid \mathsf{r}_1, \ldots, \mathsf{r}_n)$$

the output of the protocol Π when executed with the adversary $\mathcal{A}$, and where each honest party P_i uses randomness r_i.

Let Π be a protocol whose output is a string in $\{0,1\}^m$ for some $m \in \mathbb{N}$. Loosely speaking, we say that an adversary is "successful" if he manages to bias the output of the protocol to his advantage. More specifically, we say that an adversary is "successful" if he chooses a predetermined subset $M \subseteq \{0,1\}^m$ of some size s, and succeeds in biasing the outcome towards the set M. To this end, for any set size s, we define

$$\begin{aligned}\mathsf{succ}_{\mathsf{s}}(\mathcal{A}_\Pi) &\stackrel{\text{def}}{=} \max_{M \subseteq \{0,1\}^m \text{ s.t. } |M|=s} \mathsf{succ}_M(\mathcal{A}_\Pi) \\ &\stackrel{\text{def}}{=} \max_{M \subseteq \{0,1\}^m \text{ s.t. } |M|=s} \Bigg(\Pr_{\mathsf{r}_1,\ldots,\mathsf{r}_n}[\mathsf{out}(\mathcal{A}_\Pi \mid \mathsf{r}_1, \ldots, \mathsf{r}_n) \in M] - \\ &\qquad\qquad \Pr_{\mathsf{r}_1,\ldots,\mathsf{r}_n}[\mathsf{out}_\Pi(\mathsf{r}_1, \ldots, \mathsf{r}_n) \in M] \Bigg),\end{aligned}$$

where $\mathsf{out}_\Pi(\mathsf{r}_1, \ldots, \mathsf{r}_n)$ denotes the outcome of the protocol Π if all the parties are honest, and use randomness $\mathsf{r}_1, \ldots, \mathsf{r}_n$.

Intuitively, the reason we parameterize over the set size s is that we may hope for different values of $\mathsf{succ}_M(\mathcal{A}_\Pi)$ for sets M of different sizes, since for a large set M it is often the case that $\Pr_{\mathsf{r}_1,\ldots,\mathsf{r}_n}[\mathsf{out}_\Pi(\mathsf{r}_1, \ldots, \mathsf{r}_n) \in M]$ is large, and hence $\mathsf{succ}_M(\mathcal{A}_\Pi)$ is inevitably small, whereas for small sets M the value $\mathsf{succ}_M(\mathcal{A}_\Pi)$ may be large.

For example, for coin-flipping protocols (where $m = 1$ and the outcome is a uniformly random bit in the case that all parties are honest), often an adversary is considered successful if it biases the outcome to his preferred bit with probability close to 1, and hence an adversary is considered successful if $\mathsf{succ}_M(\mathcal{A}_\Pi) \geq \frac{1}{2} - o(1)$ for either $M = \{0\}$ or $M = \{1\}$, whereas for general selection protocols (where m is a parameter) one often considers subsets $M \subseteq \{0,1\}^m$ of size $\gamma \cdot 2^m$ for some constant $\gamma > 0$,

and an adversary is considered successful if there exists a constant $\delta > 0$ such that $\mathsf{succ}_M(\mathcal{A}_{\Pi}) \geq \delta$.

Definition 2 (Security). *Fix any constant $\delta > 0$, any $t = t(n) \leq n$, and any n-party protocol Π whose output is an element in $\{0,1\}^m$. Fix any $s = s(m)$. We say that Π is* (t, δ, s)-adaptively secure *if for any adversary $\mathcal{A}$ that adaptively corrupts up to $t = t(n)$ parties, it holds that*

$$\mathsf{succ}_{\mathsf{s}}(\mathcal{A}_{\Pi}) \leq \delta.$$

We note that this definition generalizes the standard security definition for coin-flipping protocols and selection protocols. We emphasize that our results are quite robust to the specific security definition that we consider, and we could have used alternative definitions as well. Intuitively, the reason is that we show how to transform any d-round protocol Π into another d-round protocol with short messages, that simulates Π (see Definition 3 below), where this transformation is *independent* of the security definition. Then, in order to prove that the resulting protocol is as secure as the original protocol Π, we show that if there exists an adversary for the short protocol that manages to break security according to some definition, then there exists an adversary for Π that "simulates" the adversary of the short protocol and breaches security in the same way. (See Section 1.3 for more details, and Section 3 for the formal argument).

Finally, we mention that an analogous definition to Definition 2 can be given for static adversaries. Our results hold for the static definition as well.

Definition 3 (Simulation). *Let Π be an n-party protocol with outputs in $\{0,1\}^m$. We say that an n-party protocol Π' simulates Π if*

$$\mathsf{SD}\left(\mathsf{out}_{\Pi}, \mathsf{out}_{\Pi'}\right) = \mathsf{negl}(n),$$

where out_{Π} is a random variable that corresponds to the output of protocol Π assuming all parties are honest, and $\mathsf{out}_{\Pi'}$ is a random variable that corresponds to the output of protocol Π' assuming all parties are honest.

3 Compressing Communication in Distributed Protocols

In this section we show how to transform any n-party d-round t-adaptively secure public-coin protocol, that outputs messages of length m and sends messages of length L, into an n-party d-round t-adaptively secure public-coin protocol in which every party sends messages of length $\ell = m \cdot \mathsf{polylog}(n, d)$.

Throughout this section, we fix μ^* to be the negligible function defined by

$$\mu^* = \mu^*(n,d) = \left(\sqrt{\varepsilon} + 1 - (1-\varepsilon)^{dn}\right) \cdot 2dn, \tag{2}$$

and where $\varepsilon = 2^{-\log^2(dn)}$.

Theorem 2. *Fix any $m = m(n)$, $d = d(n)$, $L = L(n)$, and any n-party d-round public-coin protocol Π that outputs messages in $\{0,1\}^m$ and in which all parties send messages of length $L = L(n)$. Then, for any constant $\delta > 0$, any $t = t(n) < n$, and any $s = s(m)$, if Π is (t,δ,s)-adaptively secure then there exists an n-party d-round (t,δ',s)-adaptively secure public-coin protocol, that simulates Π, where all parties send messages of length $\ell = m \cdot \log^4(n \cdot d)$, and where $\delta' \leq \delta + \mu^*$ (and $\mu^* = \mu^*(n,d)$ is the negligible function defined in Equation (2)).*

Proof. Fix any $m = m(n)$, $d = d(n)$, $L = L(n)$, and any n-party d-round public-coin protocol Π that outputs messages in $\{0,1\}^m$ and in which all parties send messages of length $L = L(n)$. Fix any constant $\delta > 0$, any $t = t(n) < n$, and any $s = s(m)$ such that Π is (t,δ,s)-adaptively secure. We start by describing the construction of the (short message) protocol. Let

$$N = 2^\ell = 2^{m \cdot \log^4(n \cdot d)}. \tag{3}$$

Let

$$\mathcal{H} = \{H : [d \cdot n] \times \{0,1\}^\ell \to \{0,1\}^L\}$$

be the set all possible $[d \cdot n] \times \{0,1\}^\ell \equiv [d \cdot n] \times [N]$ matrices, whose elements are from $\{0,1\}^L$. Note that $|\mathcal{H}| = 2^{d \cdot n \cdot N \cdot L}$. We often interpret $H : [d \cdot n] \times \{0,1\}^\ell \to \{0,1\}^L$ as a function

$$H : [d] \times [n] \times \{0,1\}^\ell \to \{0,1\}^L,$$

or as a matrix where each row is described by a pair from $[d] \times [n]$. We abuse notation and denote by

$$H(i,j,\mathsf{r}) \triangleq H((i-1)n + j, \mathsf{r}).$$

As a convention, we denote by R a message from $\{0,1\}^L$ and by r and a message from $\{0,1\}^\ell$.

From now on, we assume for the sake of simplicity of notation, that in protocol Π, in each round, all the parties send a message. Recall that we also assume for the sake of simplicity (and without loss of generality)

that Π is a public-coin protocol (see Definition 1). For any $H \in \mathcal{H}$ we define a protocol Π_H that simulates the execution of the protocol Π, as follows.

The Protocol Π_H. In the protocol Π_H, for every $i \in [d]$ and $j \in [n]$, in the i^{th} round, party P_j sends a random string $\mathsf{r}_{i,j} \leftarrow \{0,1\}^\ell$. We denote the resulting transcript in round i by

$$\mathsf{Trans}_{H,i} = (\mathsf{r}_{i,1}, \ldots, \mathsf{r}_{i,n}) \in \left(\{0,1\}^\ell\right)^n,$$

and denote the entire transcript by

$$\mathsf{Trans}_H = (\mathsf{Trans}_{H,1} \ldots, \mathsf{Trans}_{H,d}).$$

We abuse notation, and define for every round $i \in [d]$,

$$H(\mathsf{Trans}_{H,i}) = (H(i,1,\mathsf{r}_{i,1}), \ldots, H(i,n,\mathsf{r}_{i,n})).$$

Similarly, we define

$$H(\mathsf{Trans}_H) = (H(\mathsf{Trans}_{H,1}) \ldots, H(\mathsf{Trans}_{H,d})).$$

The outcome of protocol Π_H with transcript Trans_H is defined to be the outcome of protocol Π with transcript $H(\mathsf{Trans}_H)$.

It is easy to see that the round complexity of Π_H (for every $H \in \mathcal{H}$) is the same as that of Π. Moreover, we note that with some complication in notation we could have also preserved the exact communication pattern (instead of assuming that in each round all parties send a message).

In order to prove Theorem 1 it suffices to prove the following two lemmas.

Lemma 1. *There exists a subset $\mathcal{H}_0 \subseteq \mathcal{H}$ of size $\frac{|\mathcal{H}|}{2}$, such that for every matrix $H \in \mathcal{H}_0$ it holds that Π_H is (t, δ', s)-adaptively secure for $\delta' = \delta + \mu^*$, where μ^* is the negligible function defined in Equation (2).*

Lemma 2. *There exists a negligible function $\mu = \mu(n,d)$ such that,*

$$\Pr_{H \leftarrow \mathcal{H}}[\mathsf{SD}(\mathsf{out}_{\Pi_H}, \mathsf{out}_\Pi) \leq \mu] \geq \frac{2}{3}.$$

Indeed, given Lemmas 1 and 2, we obtain that there exists an $H \in \mathcal{H}$ such that Π_H is (t, δ', s)-adaptively secure and it simulates Π.

The proofs of Lemmas 1 and 2 can be found in the full version [5].

3.1 Static Adversaries

We note that Theorem 2 holds also for static adversary. For completeness, we restate the theorem for static adversaries.

Theorem 3. *Fix any $m = m(n)$, $d = d(n)$, $L = L(n)$, and any n-party d-round public-coin protocol Π that outputs messages in $\{0,1\}^m$ and in which all parties send messages of length $L = L(n)$. Then, for any constant $\delta > 0$, any $t = t(n) < n$, and any $s = s(m)$, if Π is (t, δ, s)-statically secure then there exists an n-party d-round (t, δ', s)-statically secure public-coin protocol that simulates Π, where all parties send messages of length $\ell = m \cdot \log^4(n \cdot d)$, and where $\delta' \leq \delta + \mu^*$ (where $\mu^* = \mu^*(n, d)$ is the negligible function defined in Equation (2)).*

The proof of Theorem 3 is almost identical to the proof of Theorem 2. An outline is given in the full version [5].

4 Public-Coin Protocols

In this section we show how to convert any distributed protocol in which parties do not have private inputs into a public-coin protocol.

Theorem 4. *Every protocol Π in which parties do not have private inputs can be transformed into a protocol Π' which simulates Π and such that the messages sent in Π' are uniformly random. Moreover, the protocol Π' preserves the security of Π and its round complexity.*

The proof sketch of this theorem can be found in the full version [5].

Acknowledgments We thank Nancy Lynch, Merav Parter and David Peleg for helpful remarks and pointers. The second author thanks his advisor Moni Naor for his continuous support.

References

1. Ben-Or, M., Linial, N.: Collective coin flipping, robust voting schemes and minima of banzhaf values. In: 26th Annual Symposium on Foundations of Computer Science, FOCS. pp. 408–416 (1985)
2. Feige, U.: Noncryptographic selection protocols. In: 40th Annual Symposium on Foundations of Computer Science, FOCS. pp. 142–153 (1999)
3. Goldreich, O., Sheffet, O.: On the randomness complexity of property testing. Computational Complexity 19(1), 99–133 (2010)

4. Goldwasser, S., Kalai, Y.T., Park, S.: Adaptively secure coin-flipping, revisited. In: 42nd International Colloquium on Automata, Languages and Programming,, ICALP. pp. 663–674 (2015)
5. Kalai, Y.T., Komargodski, I.: Compressing communication in distributed protocols. Electronic Colloquium on Computational Complexity (ECCC) 22, 92 (2015)
6. Kushilevitz, E., Nisan, N.: Communication complexity. Cambridge University Press (1997)
7. Kushilevitz, E., Ostrovsky, R.: Replication is NOT needed: SINGLE database, computationally-private information retrieval. In: 38th Annual Symposium on Foundations of Computer Science, FOCS. pp. 364–373 (1997)
8. Lichtenstein, D., Linial, N., Saks, M.E.: Some extremal problems arising form discrete control processes. Combinatorica 9(3), 269–287 (1989)
9. Naor, M., Nissim, K.: Communication preserving protocols for secure function evaluation. In: 33rd Annual ACM Symposium on Theory of Computing, STOC. pp. 590–599 (2001)
10. Newman, I.: Private vs. common random bits in communication complexity. Inf. Process. Lett. 39(2), 67–71 (1991)
11. Peleg, D.: Distributed Computing: A Locality-sensitive Approach. Society for Industrial and Applied Mathematics (2000)
12. Peleg, D., Rubinovich, V.: A near-tight lower bound on the time complexity of distributed minimum-weight spanning tree construction. SIAM J. Comput. 30(5), 1427–1442 (2000)
13. Russell, A., Saks, M.E., Zuckerman, D.: Lower bounds for leader election and collective coin-flipping in the perfect information model. SIAM J. Comput. 31(6), 1645–1662 (2002)
14. Sarma, A.D., Molla, A.R., Pandurangan, G., Upfal, E.: Fast distributed pagerank computation. Theor. Comput. Sci. 561, 113–121 (2015)