Must the Communication Graph of MPC Protocols be an Expander?

Elette Boyle* Ran Cohen† Deepesh Data‡ Pavel Hubáček§

May 31, 2018

Abstract

Secure multiparty computation (MPC) on incomplete communication networks has been studied within two primary models: (1) Where a partial network is fixed a priori, and thus corruptions can occur dependent on its structure, and (2) Where edges in the communication graph are determined dynamically as part of the protocol. Whereas a rich literature has succeeded in mapping out the feasibility and limitations of graph structures supporting secure computation in the fixed-graph model (including strong classical lower bounds), these bounds do not apply in the latter dynamic-graph setting, which has recently seen exciting new results, but remains relatively unexplored.

In this work, we initiate a similar foundational study of MPC within the dynamic-graph model. As a first step, we investigate the property of graph *expansion*. All existing protocols (implicitly or explicitly) yield communication graphs which are expanders, but it is not clear whether this is inherent.

Our results consist of two types (for constant fraction of corruptions):

- Upper bounds: We demonstrate secure protocols whose induced communication graphs are *not* expander graphs, within a wide range of settings (computational, information theoretic, with low locality, even with low locality *and* adaptive security), each assuming some form of input-independent setup.
- Lower bounds: In the setting without setup and adaptive corruptions, we demonstrate that for certain functionalities, *no* protocol can maintain a non-expanding communication graph against all adversarial strategies. Our lower bound relies only on protocol correctness (not privacy), and requires a surprisingly delicate argument.

More generally, we provide a formal framework for analyzing the evolving communication graph of MPC protocols, giving a starting point for studying the relation between secure computation and further, more general graph properties.

*IDC Herzliya. E-mail: elette.boyle@idc.ac.il.

†MIT and Northeastern University. E-mail: rancohen@mit.edu.

‡UCLA. E-mail: deepeshdata@ucla.edu.

§Computer Science Institute, Charles University, Prague. E-mail: hubacek@iuuk.mff.cuni.cz.

Contents

1 Introduction

The field of secure multiparty computation (MPC), and more broadly fault-tolerant distributed computation, constitutes a deep and rich literature, yielding a vast assortment of protocols providing strong robustness and even seemingly paradoxical privacy guarantees. A central setting is that of n parties who wish to jointly compute some function of their inputs while maintaining correctness (and possibly input privacy) in the face of adversarial behavior from a constant fraction of corruptions.

Since the original seminal results on secure multiparty computation [40, 6, 21, 60], the vast majority of MPC solutions to date assume that every party can (and will) communicate with every other party. That is, the underlying point-to-point communication network forms a complete graph. Indeed, many MPC protocols begin directly with every party secret sharing his input across all other parties (or simply sending his input, in the case of tasks without privacy such as Byzantine agreement [59, 54, 28]).

There are two classes of exceptions to this rule, which consider MPC on incomplete communication graphs.

Fixed-Graph Model. The first corresponds to an area of work investigating achievable security guarantees in the setting of a *fixed* partial communication network. In this model, communication is allowed only along edges of a fixed graph, known a priori, and hence where corruptions can take place as a function of its structure. This setting is commonly analyzed within the distributed computing community. In addition to positive results, this is the setting of many fundamental lower bounds: For example, to achieve secure Byzantine agreement against t corruptions, the graph must be $(t+1)$-connected [27, 33].[1] For graphs with lower connectivity, the best one can hope for is a form of "almost-everywhere agreement," where some honest parties are not guaranteed to output correctly, as well as restricted notions of privacy [31, 37, 20, 41, 42]. Note that because of this, one cannot hope to achieve protocols with standard security in this model with $o(n^2)$ communication, even for simple functionalities such as Byzantine agreement.

Dynamic-Graph Model. The second, more recent approach addresses a model where all parties have the *ability* to initiate communication with one another, but make use of only a subset of these edges as determined dynamically during the protocol. We refer to this as the "dynamic-graph model." When allowing for negligible error (in the number of parties), the above lower bounds do not apply, opening the door for dramatically different approaches and improvements in complexity. Indeed, distributed protocols have been shown for Byzantine agreement in this model with as low as $\tilde{O}(n)$ bits of communication [50, 14], and secure MPC protocols have been constructed whose communication graphs have degree $o(n)$—and as low as polylog(n) [26, 12, 19, 13].[2] However, unlike the deep history of the model above, the current status is a sprinkling of positive results. Little is known about what types of communication graphs must be generated from a secure MPC protocol execution.

Gaining a better understanding of this regime is motivated not only to address fundamental questions, but also to provide guiding principles for future protocol design. In this work, we take a foundational look at the dynamic-graph model, asking:

What properties of induced communication graphs
are necessary to support secure computation?

[1] If no setup assumptions are assumed, the connectivity bound increases to $2t+1$.

[2] This metric is sometimes referred to as the communication *locality* of the protocol [12].

On the necessity of graph expansion. Classical results tell us that the fully connected graph suffices for secure computation. Protocols achieving low locality indicate that a variety of significantly sparser graphs, with many low-weight cuts, can also be used [26, 12, 19, 13]. We thus consider a natural extension of connectivity to the setting of low degree. Although the positive results in this setting take different approaches and result in different communication graph structures, we observe that in each case, the resulting sparse graph has high *expansion*.

Roughly, a graph is an expander if every subset of its nodes that is not "too large" has a "large" boundary. Expander graphs have good mixing properties and in a sense "mimic" a fully connected graph. There are various ways of formalizing expansion; in this work we consider a version of *edge* expansion, pertaining to the number of outgoing edges from any subset of nodes. We consider a variant of the expansion definition which is naturally monotonic: that is, expansion cannot decrease when extra edges are added (note that such monotonicity also holds for the capacity of the graph to support secure computation).

Indeed, expander graphs appear explicitly in some works [50, 19], and implicitly in others (e.g., using random graphs [48], pseudorandom graphs [12], and averaging samplers [14], to convert from almost-everywhere to everywhere agreement). High connectivity and good mixing intuitively go hand-in-hand with robustness against corruptions, where adversarial entities may attempt to impede or misdirect information flow.

This raises the natural question: Is this merely an artifact of a convenient construction, or is high expansion *inherent*? That is, we investigate the question: Must the communication graph of a generic MPC protocol, tolerating a linear number of corruptions, be an expander graph?

1.1 Our Results

More explicitly, we consider the setting of secure multiparty computation with n parties in the face of a linear number of active corruptions. As common in the honest-majority setting, we consider protocols that guarantee output delivery. Communication is modeled via the dynamic-graph setting, where all parties have the ability to initiate communication with one another, and use a subset of edges as dictated by the protocol. We focus on the synchronous setting.

Our contributions are of the following three kinds:

Formal definitional framework. As a first contribution, we provide a formal framework for analyzing and studying the evolving communication graph of MPC protocols. The framework abstracts and refines previous approaches concerning specific properties of protocols implicitly related to the graph structure, such as the degree [12]. This gives a starting point for studying the relation between secure computation and further, more general, graph properties.

Upper bounds. We present secure protocols whose induced communication graphs are decidedly *not* expander graphs, within a range of settings. This includes with computational security, with information-theoretic security, with low locality, even with low locality *and* adaptive security (in a hidden-channels model [19]) — but all with the common assumption of some form of input-independent *setup* information. The resulting communication graph has a low-weight cut, splitting the n parties into two equal (linear) size sets with only poly-logarithmic edges connecting them.

Theorem 1.1 (MPC with non-expanding communication graph, informal)**.** *For any efficient functionality f and any constant $\epsilon > 0$, there exists a protocol in the PKI model, assuming digital*

signatures, securely realizing f against $(1/4-\epsilon)\cdot n$ static corruptions, such that with overwhelming probability the induced communication graph is non-expanding.

Theorem 1.1 is stated in the computational setting with static corruptions; however, this approach extends to various other settings, albeit at the expense of a lower corruption threshold. (See Section 4 for more details.)

Theorem 1.2 (extensions of Theorem 1.1, informal)**.** *For any efficient functionality f, there exists a protocol securely realizing f, in the settings listed below, against a linear number of corruptions, such that with overwhelming probability the induced communication graph is non-expanding:*

- *In the setting of Theorem 1.1 with poly-logarithmic locality.*
- *Unconditionally, in the information-theoretic PKI model (with or without low locality).*
- *Unconditionally, in the information-theoretic PKI model, facing adaptive adversaries.*
- *Under standard cryptographic assumptions, in the PKI model, facing adaptive adversaries, with poly-logarithmic locality.*

As an interesting special case, since our protocols are over point-to-point channels and do not require a broadcast channel, these results yield the first Byzantine agreement protocols whose underlying communication graphs are not expanders.

The results in Theorems 1.1 and 1.2 all follow from a central transformation converting existing secure protocols into ones with low expansion. At a high level, the first $n/2$ parties will run a secure computation to elect two representative committees of poly-logarithmic size: one amongst themselves and the other from the other $n/2$ parties. These committees will form a "communication bridge" across the two halves (see Figure 2). The setup is used to certify the identities of the members of both committees to the receiving parties, either via public-key infrastructure for digital signatures (in the computational setting) or correlated randomness for information-theoretic signatures [63, 62] (in the information-theoretic setting).

Interestingly, this committee-based approach can be extended to the adaptive setting (with setup), in the hidden-channels model considered by [19], where the adversary is not aware which communication channels are utilized between honest parties.[3] Here, care must be taken to not reveal more information than necessary about the identities of committee members to protect them from being corrupted.

As a side contribution, we prove the first instantiation of a protocol with poly-logarithmic locality and information-theoretic security (with setup), by adjusting the protocol from [12] to the information-theoretic setting.

Theorem 1.3 (polylog-locality MPC with information-theoretic security, informal)**.** *For any efficient functionality f and any constant $\epsilon > 0$, there exists a protocol with poly-logarithmic locality in the information-theoretic PKI model, securely realizing f against computationally unbounded adversaries statically corrupting $(1/6-\epsilon)\cdot n$ parties.*

[3] Sublinear locality is impossible in the adaptive setting if the adversary is aware of honest-to-honest communication, since it can simply isolate an honest party from the rest of the protocol.

Lower bounds. On the other hand, we show that in some settings a weak form of expansion *is* a necessity. In fact, we prove a stronger statement, that in these settings the graph must have high connectivity.[4] Our lower bound is in the setting of adaptive corruptions, computational (or information-theoretic) security, and *without* setup assumptions. Our proof relies only on correctness of the protocol and not on any privacy guarantees; namely, we consider the *parallel broadcast* functionality (aka *interactive consistency* [59]), where every party distributes its input to all other parties. We construct an adversarial strategy in this setting such that no protocol can guarantee correctness against this adversary if its induced communication graph at the conclusion of the protocol has any cut with sublinear many crossing edges (referred to as a "sublinear cut" from now on).

Theorem 1.4 (high connectivity is necessary for correct protocols, informal)**.** *Let $t \in \Theta(n)$. Any $t(n)$-resilient protocol for parallel broadcast in the computational setting, tolerating an adaptive, malicious adversary* cannot *maintain an induced communication graph with a sublinear cut.*

Theorem 1.4 in particular implies that the resulting communication graph must have a form of expansion. We note that in a weaker communication model, a weaker form of consensus, namely Byzantine agreement, can be computed in a way that the underlying graph (while still an expander) has low-weight cuts [49]. We elaborate on the differences between the two settings in the related work, Section 1.4.

It is indeed quite intuitive that if a sublinear cut exists in the communication graph of the protocol, and the adversary can adaptively corrupt a linear number of parties $t(n)$, then he could corrupt the parties on the cut and block information flow. The challenge, however, stems from the fact that the cut is not known a priori but is only revealed over time, and by the point at which the cut is identifiable, all necessary information may have already been transmitted across the cut. In fact, even the identity of the cut and visible properties of the communication graph itself can convey information to honest parties about input values without actual bits being communicated. This results in a surprisingly intricate final attack, involving multiple indistinguishable adversaries, careful corruption strategies, and precise analysis of information flow. See below for more detail.

1.2 Our Techniques

We focus on the technical aspects of the lower bound result.

Overview of the attack. Consider an execution of the parallel broadcast protocol over *random* inputs. At a high level, our adversarial strategy, denoted $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$, will select a party P_{i^*} at random and attempt to block its input from being conveyed to honest parties. We are only guaranteed that somewhere in the graph will remain a sublinear cut. Because the identity of the eventual cut is unknown, it cannot be attacked directly. We take the following approach:

1. **Phase I.** Rather, our attack will first "buy time" by corrupting the neighbors of P_{i^*}, and blocking information flow of its input x_{i^*} to the remaining parties. Note that this can only continue up to a certain point, since the degree of P_{i^*} will eventually surpass the corruption threshold (as we prove). But, the benefit of this delay is that in the meantime, the communication graph starts to fill in, which provides more information about the locations of the potential cuts.

[4] More concretely, the graph should be at least $\alpha(n)$-connected for every sublinear function $\alpha(n) \in o(n)$.

For this to be the case, it must be that the parties cannot identify that P_{i^*} is under attack (otherwise, the protocol may instruct many parties to quickly communicate to/from P_{i^*}, forcing the adversary to run out of his "corruption budget" before the remaining graph fills in). The adversary thus needs to fool all honest parties and make each honest party believe that he participates in an honest execution of the protocol. This is done by maintaining two simulated executions: one pretending to be P_{i^*} running on a random input, and another pretending (to P_{i^*}) to be all other parties running on random inputs. Note that for this attack strategy to work it is essential that the parties do not have pre-computed correlated randomness such as PKI.

2. **Phase II.** We show that with noticeable probability, by the time we run out of the Phase I corruption threshold (which is a linear number of parties), *all parties* in the protocol have high (linear) degree. In turn, we prove that the current communication graph can have at most a constant number of sublinear cuts.

   In the remainder of the protocol execution, the adversary will simultaneously attack all of these cuts. Namely, he will block information flow from P_{i^*} across any of these cuts by corrupting the appropriate "bridge" party, giving up on each cut one by one when a certain threshold of edges have already crossed it.

If the protocol is guaranteed to maintain a sublinear cut, then necessarily there will remain at least one cut for which all Phase II communication across the cut has been blocked by the adversary. Morally, parties on the side of this cut opposite P_{i^*} should not have learned x_{i^*}, and thus the *correctness* of the protocol should be violated. Proving this, on the other hand, requires surmounting two notable challenges.

1. We must prove that there still remains an uncorrupted party P_{j^*} on the opposite side of the cut. It is not hard to show that each side of the cut is of linear size, that P_{i^*} has a sublinear number of neighbors across the cut (all of which are corrupted), and that a sublinear number of parties get corrupted in Phase II. Hence, there exists parties across the cut that are not neighbors of P_{i^*} and that are not corrupted in Phase II. However, by the attack strategy, all of the neighbors of the *virtual* P_{i^*} are corrupted in Phase I as well, and this is also a linear size set, which is independent of the real neighbors of P_{i^*}. Therefore, it is not clear that there will actually remain honest parties across the cut by the end of the protocol execution.

2. More importantly, even though we are guaranteed that no bits of communication have been passed along any path from P_{i^*} to P_{j^*}, this does not imply that no *information* about x_{i^*} has been conveyed. For example, since the graph develops as a function of parties' inputs, it might be the case that this situation of P_{j^*} being blocked from P_{i^*}, only occurs when x_{i^*} equals a certain value.

We now discuss how these two challenges are addressed.

Guaranteeing honest parties across the cut. Unexpectedly, we cannot guarantee existence of honest parties across the cut. Instead, we introduce a different adversarial strategy, which we prove *must* have honest parties blocked across a cut from P_{i^*}, and for which there exist honest parties who cannot distinguish which of the two attacks is taking place. More explicitly, we consider the

"dual" version of the original attack, denoted $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$, where party P_{i^*} is *corrupted* and instead pretends to be under attack as per $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ above.

Blocking honest parties from x_{i^*} in $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ does not contradict correctness explicitly on its own, as P_{i^*} is corrupted in this case. It is the combination of both of these attacks that will enable us to contradict correctness. Namely, we prove that:

- Under the attack $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$, there exists a "blocked cut" $(S, \bar{S})$ with uncorrupted parties on both sides. By *agreement*, all uncorrupted parties output the same value y_{i^*} as the i^*'th coordinate of the output vector.
- The view of some of the uncorrupted parties under the attack $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ is identically distributed as that of uncorrupted parties in the original attack $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$. Thus, their output distribution must be the same across the two attacks.
- Since under the attack $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$, the party P_{i^*} is honest, by *completeness*, all uncorrupted parties in $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ must output the *correct* value $y_{i^*} = x_{i^*}$.
- Thus, uncorrupted parties in $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ (who have the same view) must output the correct value x_{i^*} as well.

Altogether, this implies all honest parties in interaction with $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$, in particular P_{j^*} who is blocked across the cut from P_{i^*}, must output $y_{i^*} = x_{i^*}$.

Bounding information transmission about x_{i^*}. The final step is to show that this cannot be the case, since an uncorrupted party P_{j^*} across the cut in $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ does not receive enough information about x_{i^*} to fully specify the input. This demands delicate treatment of the specific attack strategy and analysis, as many "side channel" signals within the protocol can leak information on x_{i^*}. Corruption patterns in Phase II, and their timing, can convey information "across" the isolated cut. In fact, even the event of successfully reaching Phase II may be correlated with the value of x_{i^*}.

For example, say the cut at the conclusion of the protocol is $(S_1, \bar{S}_1)$ with $i^* \in S_1$ and $j^* \in \bar{S}_1$, but at the beginning of Phase II there existed another cut $(S_2, \bar{S}_2)$, for which $S_1 \cap S_2 \neq \emptyset$, $S_1 \cap \bar{S}_2 \neq \emptyset$, $\bar{S}_1 \cap S_2 \neq \emptyset$, and $\bar{S}_1 \cap \bar{S}_2 \neq \emptyset$. Since any "bridge" party in $\bar{S}_2$ that receives a message from S_2, gets corrupted and discards the message, the view of honest parties in $\bar{S}_1$ might change as a result of the corruption related to the cut $(S_2, \bar{S}_2)$, which in turn could depend on x_{i^*}. See Figure 1 for an illustration of this situation.

Ultimately, we ensure that the final view of P_{j^*} in the protocol can be simulated given only "Phase I" information, which is independent of x_{i^*}, in addition to the identity of the final cut in the graph, which reveals only a constant amount of additional entropy.

Additional subtleties. The actual attack and its analysis are even more delicate. For example, it is important that the degree of the "simulated P_{i^*}," by the adversarial strategy $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$, will reach the threshold faster than the real P_{i^*}. In addition, in each of these cases, the threshold, and so the transition to the next phase, could possibly be reached in a middle of a round, requiring detailed treatment.

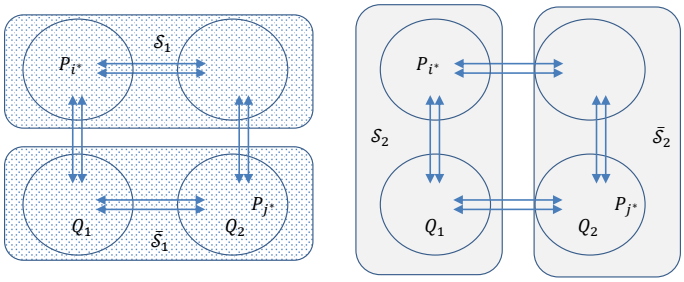

Figure 1: At the end of Phase I the communication graph is partitioned into 4 linear-size "islands" that are connected by sublinear many edges. On the left is the potential cut $(S_1, \bar{S}_1)$ and on the right the potential cut $(S_2, \bar{S}_2)$. If party Q_1 sends a message to party Q_2 then Q_2 gets corrupted and discards the message. This event can be identified by all parties in $\bar{S}_1$, and in particular by P_{j^*} by the end of the protocol.

1.3 Open Questions

This work leaves open many interesting lines of future study.

- Bridging the gap between upper and lower bounds. This equates to identifying the core properties that necessitate graph expansion versus not. Natural candidates suggested by our work are existence of setup information and adaptive corruptions in the hidden or visible (yet private) channels model.
- What other graph properties are necessary (or not) to support secure computation? Our new definitional framework may aid in this direction.
- Our work connects graph theory and secure protocols, giving rise to further questions and design principles. For example, can good constructions of expanders give rise to new communication-efficient MPC? On the other hand, can necessity of expansion (in certain settings) be used to argue new communication complexity lower bounds?

1.4 Additional Related Work

Communication graphs induced by fault-tolerant protocols is a field that has been intensively studied in various aspects.

In the fixed-graph model, where the parties can communicate over a pre-determined partial graph, there have been many work for realizing secure message transmission [29, 35, 61, 52, 65, 1, 34, 48, 64], Byzantine agreement [27, 31, 7, 66], and secure multiparty computation [5, 9, 8, 20, 17, 18].

In the setting of *topology-hiding* secure computation [58, 45, 2, 3], parties communicate over a partial graph, and the goal is to hide which pairs of honest parties are neighbors. Intuitively, topology-hiding protocols can support non-expanding graphs since sublinear cuts should not be revealed during the protocol. This is a stronger property than considered in this work, as we do not aim to hide the topology of the graph (in particular, the entire communication graph can be revealed by the conclusion of the protocol). Due to strong lower bounds [58], the current state of the art of topology-hiding secure computation tolerate only semi-honest or fail-stop adversaries, and do not extend to the malicious setting as considered in this work.

Another direction, studied in [43, 53], is to consider MPC protocol that use oblivious transfer (OT), and to analyze the graph structure that is induced by all pairwise OT channels that are used by the protocol.

King and Saia [49] presented a Byzantine agreement protocol that is secure against adaptive corruptions and (while still being an expander) its communication graph has sublinear cuts. Compared to our lower bound (Section 5), both results do not assume any trusted setup, and both support adaptive corruptions and visible communication (i.e., the adversary is aware of honest-to-honest communication). However, we highlight two aspects in which the setting in [49] is weaker. First, [49] realize *Byzantine agreement* which is a weaker functionality than parallel broadcast. Indeed, the standard techniques of reducing broadcast to Byzantine agreement do not support sublinear cuts. The second is the communication model, as [49] consider *atomic message delivery*, meaning that once a party has sent a message to the network, the adversary cannot change the content of the message even by corrupting the sender and before any honest party received it. For more details see [38], where atomic message delivery was used to overcome the lower bound of [44].

Paper Organization

Basic notations are presented in Section 2. In Section 3, we provide our formalization of the communication graph induced by a MPC protocol and related properties. In Section 4, we describe our upper bound results, constructing protocols with non-expanding graphs. In Section 5, we prove our lower bound.

2 Preliminaries

Graph-theoretic notations. Let $G = (V, E)$ be an undirected graph of size n, i.e., $|V| = n$. Given a set $S \subseteq V$, we denote its complement set by $\bar{S}$, i.e., $\bar{S} = V \setminus S$. Given two disjoint subsets $U_1, U_2 \subseteq V$ define the set of all the edges in G for which one end point is in U_1 and the other end point is in U_2 as

$$\mathsf{edges}_G(U_1, U_2) := \{(u_1, u_2) : u_1 \in U_1, u_2 \in U_2, \text{ and } (u_1, u_2) \in E\}.$$

We denote by $|\mathsf{edges}_G(U_1, U_2)|$ the total number of edges going across U_1 and U_2. For simplicity, we denote $\mathsf{edges}_G(S) = \mathsf{edges}_G(S, \bar{S})$. A cut in the graph G is a partition of the vertices V into two non-empty, disjoint sets $\{S, \bar{S}\}$. An α-cut is a cut $\{S, \bar{S}\}$ such that $|\mathsf{edges}_G(S)| \leq \alpha$.

Given a graph $G = (V, E)$ and a node $i \in V$, denote by $G \setminus \{i\} = (V', E')$ the graph obtained by removing node i and all its edges, i.e., $V' = V \setminus \{i\}$ and $E' = E \setminus \{(i, j) \mid j \in V'\}$.

MPC Model. We consider multiparty protocols in the stand-alone, synchronous model, and require security with guaranteed output delivery. We refer the reader to [15, 39] for a precise definition of the model. Throughout the paper we assume malicious adversaries that can deviate from the protocol in an arbitrary manner. We will consider both *static* corruptions, where the set of corrupted parties is fixed at the onset of the protocol, and *adaptive* corruptions, where the adversary can dynamically corrupt parties during the protocol execution, In addition, we will consider both PPT adversaries and computationally unbounded adversaries

Recall that in the synchronous model protocols proceed in rounds, where every round consists of a *send phase* followed by a *receive phase*. The adversary is assumed to be *rushing*, meaning that

he can determine the messages for corrupted parties *after* seeing the messages sent by the honest parties. We assume a complete network of point-to-point channels (broadcast is not assumed), where every party has the ability to send a message to every other party. We will normally consider *secure* (private) channels where the adversary learns that a message has been sent between two honest parties, but not its content. If a public-key encryption is assumed, this assumption can be relaxed to *authenticated* channels, where the adversary can learn the content of all messages (but not change them). For our upper bound in the adaptive setting (Section 4.2.2) we consider *hidden* channels (as introduced in [19]), where the adversary does not even know whether two honest parties have communicated or not.

3 Communication Graphs Induced by MPC Protocols

In this section, we present formal definitions of properties induced by the communication graph of interactive protocols. These definitions are inspired by previous works in distributed computing [50, 47, 49, 51] and multiparty computation [12, 19, 13] that constructed interactive protocols with *low locality*.

3.1 Ensembles of Protocols and Functionalities

In order to capture certain asymptotic properties of the communication graphs of generic n-party protocols, such as edge expansion and locality, it is useful to consider a family of protocols that are parametrized by the number of parties n. This is implicit in many distributed protocols and in generic multiparty protocols, for example [59, 54, 28, 40, 6]. We note that for many large-scale protocols, e.g., protocols with low locality [50, 47, 49, 51, 12, 13], the security guarantees increase with the number of parties, and in fact, the number of parties is assumed to be polynomially related to the security parameter.

Definition 3.1 (protocol ensemble)**.** *Let $f = \{f_n\}_{n\in\mathbb{N}}$ be an ensemble of functionalities, where f_n is an n-party functionality, let $\pi = \{\pi_n\}_{n\in\mathbb{N}}$ be an ensemble of protocols, and let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be an ensemble of classes of adversaries (e.g., $\mathcal{C}_n$ is the class of PPT $t(n)$-adversaries). We say that π securely computes f tolerating adversaries in $\mathcal{C}$ if for every n that is polynomially related to the security parameter κ, it holds that π_n securely computes f_n tolerating adversaries in $\mathcal{C}_n$.*

In Section 4, we will consider several classes of adversaries. We use the following notation for clarity and brevity.

Definition 3.2. *Let $f = \{f_n\}_{n\in\mathbb{N}}$ be an ensemble of functionalities and let $\pi = \{\pi_n\}_{n\in\mathbb{N}}$ be an ensemble of protocols. We say that π securely computes f tolerating adversaries of the form* type *(e.g., static PPT $t(n)$-adversaries, adaptive $t(n)$-adversaries, etc.), if π securely computes f tolerating adversaries in $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$, where for every n, the set $\mathcal{C}_n$ is the class of adversaries of the form* type*.*

3.2 The Communication Graph of a Protocol's Execution

Intuitively, the communication graph induced by a protocol should include an edge (i, j) precisely if parties P_i and P_j exchange messages during the protocol execution. For instance, consider the property of *locality*, corresponding to the maximum degree of the communication graph. When

considering malicious adversaries that can deviate from the protocol using an arbitrary strategy, it is important to consider only messages that are sent by honest parties and messages that are received by honest parties. Otherwise, every corrupted party can send a message to every other corrupted party, yielding a subgraph with degree $\Theta(n)$. We note that restricting the analysis to only consider honest parties is quite common in the analysis of protocols.

Another issue that must be taken under consideration is flooding by the adversary. Indeed, there is no way to prevent the adversary from sending messages from all corrupted parties to all honest parties; however, we wish to only count those message which are actually processed by honest parties. To model this, the *receive* phase of every communication round[5] is composed of two sub-phases:

1. *The filtering sub-phase:* Each party inspects the list of messages received in the previous round, according to specific filtering rules defined by the protocol, and discards the messages that do not pass the filter. The resulting list of messages is appended to the local transcript of the protocol.

2. *The processing sub-phase:* Based on its local transcript, each party computes the next-message function and obtains the list of messages to be sent in the current round along with the list of recipients, and sends them to the relevant parties.

In practice, the filtering procedure should be "lightweight," such as verifying validity of a signature. However, we assume only an abstraction and defer the actual choice of filtering procedure (as well as corresponding discussion) to specific protocol specifications. We note that the above two-phase processing of rounds is implicit in protocols from the literature that achieve low locality [50, 47, 49, 51, 12, 19, 13]. It is also implicit when analyzing the communication complexity of general protocols, where malicious parties can send long messages to honest parties, and honest parties filter out invalid messages before processing them.

We now turn to define the communication graph of a protocol's execution, by which we mean the deterministic instance of the protocol defined by fixing the adversary and all input values and random coins of the parties and the adversarial strategy. We consider protocols that are defined in the correlated-randomness model (e.g., for establishing PKI). This is without loss of generality since by defining the "empty distribution," where every party is given an empty string, we can model also protocols in the plain model. Initially, we focus on the *static* setting, where the set of corrupted parties is determined at the onset of the protocol. We discuss the *adaptive* setting in the full version [11].

Definition 3.3 (protocol execution instance)**.** *For $n \in \mathbb{N}$, let π_n be an n-party protocol, let κ be the security parameter, let $\boldsymbol{x} = (x_1, \ldots, x_n)$ be an input vector for the parties, let $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_n)$ be correlated randomness for the parties, let $\mathcal{A}$ be an adversary, let z be the auxiliary information of $\mathcal{A}$, let $\mathcal{I} \subseteq [n]$ be the set of indices of corrupted parties controlled by $\mathcal{A}$, and let $\boldsymbol{r} = (r_1, \ldots, r_n, r_{\mathcal{A}})$ be the vector of random coins for the parties and for the adversary.*

Denote by $\mathsf{instance}(\pi_n) = (\pi_n, \mathcal{A}, \mathcal{I}, \kappa, \boldsymbol{x}, \boldsymbol{\rho}, z, \boldsymbol{r})$ the list of parameters that deterministically define an execution instance of the protocol π_n.

Note that $\mathsf{instance}(\pi_n)$ fully specifies the entire views and transcript of the protocol execution, including all messages sent to/from honest parties.

[5] Recall that in the synchronous model, every communication round is composed of a *send* phase and a *receive* phase.

Definition 3.4 (communication graph of protocol execution)**.** *For $n \in \mathbb{N}$, let $\mathsf{instance}(\pi_n) = (\pi_n, \mathcal{A}, \mathcal{I}, \kappa, \boldsymbol{x}, \boldsymbol{\rho}, z, \boldsymbol{r})$ be an execution instance of the protocol π_n. We now define the following communication graphs induced by this execution instance. Each graph is defined over the set of n vertices $[n]$.*

- Outgoing communication graph. *The directed graph $G_{\mathsf{out}}(\mathsf{instance}(\pi_n)) = ([n], E_{\mathsf{out}})$ captures all the communication lines that are used by honest parties to send messages. That is,*

$$E_{\mathsf{out}}(\mathsf{instance}(\pi_n)) = \{(i,j) \mid P_i \text{ is honest and sent a message to } P_j\}\,.$$

- Incoming communication graph. *The directed graph $G_{\mathsf{in}}(\mathsf{instance}(\pi_n)) = ([n], E_{\mathsf{in}})$ captures all the communication lines in which honest parties received messages that were processed (i.e., excluding messages that were filtered out). That is,*

$$E_{\mathsf{in}}(\mathsf{instance}(\pi_n)) = \{(i,j) \mid P_j \text{ is honest and processed a message received from } P_i\}\,.$$

- Full communication graph. *The undirected graph $G_{\mathsf{full}}(\mathsf{instance}(\pi_n)) = ([n], E_{\mathsf{full}})$ captures all the communication lines in which honest parties received messages that were processed, or used by honest parties to send messages. That is,*

$$E_{\mathsf{full}}(\mathsf{instance}(\pi_n)) = \{(i,j) \mid (i,j) \in E_{\mathsf{out}} \text{ or } (i,j) \in E_{\mathsf{in}}\}\,.$$

We will sometimes consider ensembles of protocol instances (for $n \in \mathbb{N}$) and the corresponding ensembles of graphs they induce.

Looking ahead, in subsequent sections we will consider the full communication graph G_{full}. Apart from making the presentation clear, the graphs G_{out} and G_{in} are used for defining G_{full} above, and the locality of a protocol in Definition 3.5. Note that G_{out} and G_{in} are interesting in their own right, and can be used for a fine-grained analysis of the communication graph of protocols in various settings, e.g., when transmitting messages is costly but receiving messages is cheap (or vice versa). We leave it open as an interesting problem to study various graph properties exhibited by these two graphs.

3.3 Locality of a Protocol

We now present a definition of communication locality, aligning with that of [12], with respect to the terminology introduced above.

Definition 3.5 (locality of a protocol instance)**.** *Let $\mathsf{instance}(\pi_n) = (\pi_n, \kappa, \boldsymbol{x}, \boldsymbol{\rho}, \mathcal{A}, z, \mathcal{I} \subseteq [n], \boldsymbol{r})$ be an execution instance as in Definition 3.4. For every honest party P_i we define the locality of party P_i to be the number of parties from which P_i received and processed messages, or sent message to; that is,*

$$\ell_i(\mathsf{instance}(\pi_n)) = \left|\{j \mid (i,j) \in G_{\mathsf{out}}\} \cup \{j \mid (j,i) \in G_{\mathsf{in}}\}\right|.$$

The locality of $\mathsf{instance}(\pi_n)$ is defined as the maximum locality of an honest party, i.e.,

$$\ell(\mathsf{instance}(\pi_n)) = \max_{i \in [n] \setminus \mathcal{I}} \{\ell_i(\mathsf{instance}(\pi_n))\}\,.$$

We proceed by defining locality as a property of a protocol ensemble. The protocol ensemble is parametrized by the number of parties n. To align with standard notions of security where asymptotic measurements are with respect to the security parameter κ, we consider the situation where the growth of n and κ are polynomially related.

Definition 3.6 (locality of a protocol)**.** *Let $\pi = \{\pi_n\}_{n\in\mathbb{N}}$ be a family of protocols in the correlated-randomness model with distribution $D_\pi = \{D_{\pi_n}\}_{n\in\mathbb{N}}$, and let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be a family of adversary classes. We say that π has locality $\ell(n)$ facing adversaries in $\mathcal{C}$ if for every n that is polynomially related to κ it holds that for every input vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, every auxiliary information z, every adversary $\mathcal{A} \in \mathcal{C}_n$ running with z, and every set of corrupted parties $\mathcal{I} \subseteq [n]$, it holds that*

$$\Pr\left[\ell(\pi_n, \mathcal{A}, \mathcal{I}, \kappa, \boldsymbol{x}, z) > \ell(n)\right] \leq \mathsf{negl}(\kappa),$$

where $\ell(\pi_n, \mathcal{A}, \mathcal{I}, \kappa, \boldsymbol{x}, z)$ is the random variable corresponding to $\ell(\pi_n, \mathcal{A}, \mathcal{I}, \kappa, \boldsymbol{x}, \boldsymbol{\rho}, z, \boldsymbol{r})$ when $\boldsymbol{\rho}$ is distributed according to D_{π_n} and $\boldsymbol{r}$ is uniformly distributed.

The following proposition follows from the sequential composition theorem of Canetti [15].

Proposition 3.7 (composition of locality)**.** *Let $f = \{f_n\}_{n\in\mathbb{N}}$ and $g = \{g_n\}_{n\in\mathbb{N}}$ be ensembles of n-party functionalities.*

- *Let $\varphi = \{\varphi_n\}_{n\in\mathbb{N}}$ be a protocol ensemble that securely computes f with locality ℓ_φ tolerating adversaries in $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$.*
- *Let $\pi = \{\pi_n\}_{n\in\mathbb{N}}$ be a protocol that securely computes g with locality ℓ_π in the f-hybrid model, tolerating adversaries in $\mathcal{C}$, using q calls to the ideal functionality.*

Then protocol $\pi^{f\mapsto\varphi}$, that is obtained from $\{\pi_n\}$ by replacing all ideal calls to f_n with the protocol φ_n, is a protocol ensemble that securely computes g in the real model, tolerating adversaries in $\mathcal{C}$, with locality at most $\ell_\pi + q \cdot \ell_\varphi$.

3.4 Edge Expansion of a Protocol

The measure of complexity we study for the communication graph of interactive protocols will be that of *edge expansion* (see discussion below). We refer the reader to [46, 30] for more background on expanders. We consider a definition of edge expansion which satisfies a natural monotonic property, where adding more edges cannot decrease the graph's measure of expansion.

Definition 3.8. *(edge expansion of a graph) Given an undirected graph $G = (V, E)$, the* edge expansion ratio of G*, denoted $h(G)$, is defined as*

$$h(G) = \min_{\{S\subseteq V : |S| \leq \frac{|V|}{2}\}} \frac{|\mathsf{edges}(S)|}{|S|}\ , \tag{1}$$

where $\mathsf{edges}(S)$ denotes the set of edges between S and its complement $\bar{S} = V \setminus S$.

Definition 3.9. *(family of expander graphs) A sequence $\{G_n\}_{n\in\mathbb{N}}$ of graphs is a* family of expander graphs *if there exists a constant $\epsilon > 0$ such that $h(G_n) \geq \epsilon$ for all n.*

We now consider the natural extension of graph expansion to the setting of protocol-induced communication graph.

Definition 3.10. *(bounds on edge expansion of a protocol) Let $\pi = \{\pi_n\}_{n\in\mathbb{N}}$, $D_\pi = \{D_{\pi_n}\}_{n\in\mathbb{N}}$, and $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be as in Definition 3.6.*

- *A function $f(n)$ is a lower bound of the edge expansion of π facing adversaries in $\mathcal{C}$, denoted $f(n) \leq h_{\pi,D_\pi,\mathcal{C}}(n)$, if for every n that is polynomially related to κ, for every $\boldsymbol{x} = (x_1,\ldots,x_n)$, every $\mathcal{A} \in \mathcal{C}_n$ running with z, and every $\mathcal{I} \subseteq [n]$, it holds that*

$$\Pr\left[h(G_{\mathsf{full}}(\pi_n,\mathcal{A},\mathcal{I},\kappa,\boldsymbol{x},z)) \leq f(n)\right] \leq \mathsf{negl}(\kappa),$$

  where $G_{\mathsf{full}}(\pi_n,\mathcal{A},\mathcal{I},\kappa,\boldsymbol{x},z)$ is the random variable $G_{\mathsf{full}}(\pi_n,\mathcal{A},\mathcal{I},\kappa,\boldsymbol{x},\boldsymbol{\rho},z,\boldsymbol{r})$, when $\boldsymbol{\rho}$ is distributed according to D_{π_n} and $\boldsymbol{r}$ is uniformly distributed.

- *A function $f(n)$ is a upper bound of the edge expansion of π facing adversaries in $\mathcal{C}$, denoted $f(n) \geq h_{\pi,D_\pi,\mathcal{C}}(n)$, if there exists a polynomial relation between n and κ such that for infinitely many n it holds that for every $\boldsymbol{x} = (x_1,\ldots,x_n)$, every $\mathcal{A} \in \mathcal{C}_n$ running with z, and every $\mathcal{I} \subseteq [n]$, it holds that*

$$\Pr\left[h(G_{\mathsf{full}}(\pi_n,\mathcal{A},\mathcal{I},\kappa,\boldsymbol{x},z)) \geq f(n)\right] \leq \mathsf{negl}(\kappa).$$

Definition 3.11 (expander protocol)**.** *Let $\pi = \{\pi_n\}_{n\in\mathbb{N}}$, $D_\pi = \{D_{\pi_n}\}_{n\in\mathbb{N}}$, and $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be as in Definition 3.6. We say that the communication graph of π is an expander, facing adversaries in $\mathcal{C}$, if there exists a constant function $\epsilon(n) > 0$ such that $\epsilon(n) \leq h_{\pi,D_\pi,\mathcal{C}}(n)$.*

We note that most (if not all) secure protocols in the literature are expanders according to Definition 3.11, both in the realm of distributed computing [28, 32, 36, 50, 47, 51, 49] and in the realm of MPC [40, 6, 12, 19, 13]. Proving that a protocol is not an expander according to this definition requires showing an adversary for which the edge expansion is sub-constant. Looking ahead, both in our constructions of protocols that are not expanders (Section 4) and in our lower bound, showing that non-expander protocols can be attacked (Section 5), we use a stronger definition, that requires that the edge expansion is sub-constant facing *all* adversaries, see Definition 3.12 below. While it makes our positive results stronger, we leave it as an interesting open question to attack protocols that do not satisfy Definition 3.11.

Definition 3.12 (strongly non-expander protocol)**.** *Let $\pi = \{\pi_n\}_{n\in\mathbb{N}}$, $D_\pi = \{D_{\pi_n}\}_{n\in\mathbb{N}}$, and $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be as in Definition 3.6. We say that the communication graph of π is strongly not an expander, facing adversaries in $\mathcal{C}$, if there exists a sub-constant function $\alpha(n) \in o(1)$ such that $\alpha(n) \geq h_{\pi,D_\pi,\mathcal{C}}(n)$.*

We next prove a useful observation that will come into play in Section 5, stating that if the communication graph of π is strongly not an expander, then there must exist a sublinear cut in the graph.

Lemma 3.13. *Let $\pi = \{\pi_n\}_{n\in\mathbb{N}}$ be a family of protocols in the correlated-randomness model with distribution $D_\pi = \{D_{\pi_n}\}_{n\in\mathbb{N}}$, and let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be such that $\mathcal{C}_n$ is the class of adversaries corrupting at most $\beta \cdot n$ parties, for a constant $0 < \beta < 1$.*

Assuming the communication graph of π is strongly non-expanding facing adversaries in $\mathcal{C}$, there exists a sublinear function $\alpha(n) \in o(n)$ such that for infinitely many n's the full communication graph of π_n has an $\alpha(n)$-cut with overwhelming probability.

Proof. Since the full communication graph of π is strongly not an expander, there exists a sub-constant function $\alpha'(n) \in o(1)$ such that there exists a polynomial relation between n and κ such that for infinitely many n's it holds that for every input $\boldsymbol{x} = (x_1, \ldots, x_n)$ and every adversary $\mathcal{A} \in \mathcal{C}_n$ and every set of corrupted parties $\mathcal{I}$,

$$\Pr\left[h(G_{\mathsf{full}}(\pi_n, \mathcal{A}, \mathcal{I}, \kappa, \boldsymbol{x}, z)) > \alpha'(n)\right] \leq \mathsf{negl}(\kappa).$$

This means that for these n's, with overwhelming probability there exists a subset $S_n \subseteq [n]$ of size at most $n/2$ for which

$$\frac{|\mathsf{edges}(S_n)|}{|S_n|} \leq \alpha'(n).$$

Since $|S_n| \leq n/2$ it holds that

$$|\mathsf{edges}(S_n)| \leq \alpha'(n) \cdot \frac{n}{2}.$$

We define $\alpha(n) = \alpha'(n) \cdot \frac{n}{2}$, and the claim thus holds for $\alpha(n) \in o(n)$. □

4 MPC with Non-Expanding Communication Graph

In this section, we show that in various standard settings, the communication graph of an MPC protocol is *not* required to be an expander graph, even when the communication locality is poly-logarithmic. In Section 4.1, we focus on static corruptions and computational security. In Section 4.2, we extend the construction to the information-theoretic setting and to the adaptive-corruption setting. The proof for these extensions can be found in the full version [11].

4.1 Computational Security with Static Corruptions

We start by considering the computational setting with static corruptions.

Theorem 4.1. *Let $f = \{f_n\}_{n\in\mathbb{N}}$ be an ensemble of functionalities, let $\delta > 0$, and assume that one-way functions exist. Then, the following holds in the PKI-hybrid model with secure channels:*

1. *Let $\beta < 1/4 - \delta$ and let $t(n) = \beta \cdot n$. Then, f can be securely computed by a protocol ensemble π tolerating static PPT $t(n)$-adversaries such that the communication graph of π is strongly not an expander.*

2. *Let $\beta < 1/6 - \delta$ and let $t(n) = \beta \cdot n$. Then, f can be securely computed by a protocol ensemble π tolerating static PPT $t(n)$-adversaries such that (1) the communication graph of π is strongly not an expander, and (2) the locality of π is poly-logarithmic in n.*

3. *Let $\beta < 1/4 - \delta$, let $t(n) = \beta \cdot n$, and assume in addition the* secret-key infrastructure (SKI) *model*[6] *and the existence of public-key encryption schemes. Then, f can be securely computed by a protocol ensemble π tolerating static PPT $t(n)$-adversaries such that (1) the communication graph of π is strongly not an expander, and (2) the locality of π is poly-logarithmic in n.*[7]

[6] In the SKI model every pair of parties has a secret random string that is unknown to other parties.
[7] This item hold in the authenticated-channels model, since we assume PKE.

Proof. The theorem follows from Lemma 4.2 (below) by instantiating the hybrid functionalities using existing MPC protocols from the literature.

- The first part follows using honest-majority MPC protocols that exist assuming one-way functions in the secure-channels model, e.g., the protocol of Beaver et al. [4] or of Damgård and Ishai [25].
- The second part follows using the low-locality MPC protocol of Boyle et al. [12] that exists assuming one-way functions in the PKI model with secure channels and tolerates $t = (1/3-\delta)n$ static corruptions.
- The third part follows using the low-locality MPC protocol of Chandran et al. [19] that exists assuming public-key encryption in the PKI and SKI model with authenticated channels and tolerates $t < n/2$ static corruptions.

□

4.1.1 Ideal Functionalities used in the Construction

The proof to Theorem 4.1 relies on Lemma 4.2 (below). We start by defining the notations and the ideal functionalities that will be used in the protocol considered in Lemma 4.2.

Signature notations. Given a signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ and m pairs of signing and verification keys $(\mathtt{sk}_i, \mathtt{vk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$ for $i \in [m]$, we use the following notations for signing and verifying with multiple keys:

- Given a message μ we denote by $\mathsf{Sign}_{\mathtt{sk}_1,\ldots,\mathtt{sk}_m}(\mu)$ the vector of m signatures $\sigma = (\sigma_1, \ldots, \sigma_m)$, where $\sigma_i \leftarrow \mathsf{Sign}_{\mathtt{sk}_i}(\mu)$.
- Given a message μ and a signature $\sigma = (\sigma_1, \ldots, \sigma_m)$, we denote by $\mathsf{Verify}_{\mathtt{vk}_{m+1},\ldots,\mathtt{vk}_{2m}}(\mu, \sigma)$ the verification algorithm that for every $i \in [m]$ computes $b_i \leftarrow \mathsf{Verify}_{\mathtt{vk}_{m+i}}(\mu, \sigma_i)$, and accepts the signature σ if and only if $\sum_{i=1}^{m} b_i \geq m - t$, i.e., even if up to t signatures are invalid.

We note that it is possible to use multi-signatures or aggregated signatures [57, 10, 56, 55] in order to obtain better communication complexity, however, we use the notation above both for simplicity and as a step towards the information-theoretic construction in the following section.

The Elect-and-Share functionality. In the Elect-and-Share m-party functionality, $f_{\mathsf{elect\text{-}share}}^{(t',n')}$, every party P_i has a pair of inputs $(x_i, \mathtt{sk}_i)$, where $x_i \in \{0,1\}^*$ is the "actual input" and $\mathtt{sk}_i$ is a private signing key. The functionality starts by electing two random subsets $\mathcal{C}_1, \mathcal{C}_2 \subseteq [m]$ of size n', and signing each subset using all signing keys. In addition, every input value x_i is secret shared using a (t', n') error-correcting secret-sharing scheme. Every party receives as output the subset $\mathcal{C}_1$, whereas a party P_i, for $i \in \mathcal{C}_1$, receives an additional output consisting of a signature on $\mathcal{C}_1$, the signed subset $\mathcal{C}_2$, along with one share for each one of the m input values.

The Reconstruct-and-Compute functionality. The Reconstruct-and-Compute functionality, $f_{\mathsf{recon\text{-}compute}}^{(\mathtt{vk}_1,\ldots,\mathtt{vk}_m)}$, is an m-party functionality. Denote the party-set by $\{P_{m+1},\ldots,P_{2m}\}$. Every party P_{m+i} has an input value $x_{m+i} \in \{0,1\}^*$, and a potential additional input value consisting of a signed subset $\mathcal{C}_2 \subseteq [m]$ and a vector of m shares. The functionality starts by verifying the signatures, where every invalid input is ignored. The signed inputs should define a single subset $\mathcal{C}_2 \subseteq [m]$ (otherwise the functionality aborts), and the functionality uses the additional inputs of parties P_{m+i}, for every $i \in \mathcal{C}_2$, in order to reconstruct the m-tuple $(x_1,\ldots,x_m)$. Finally, the functionality computes $y = f(x_1,\ldots,x_{2m})$ and hands y as the output for every party.

The Output-Distribution functionality. The m-party Output-Distribution functionality is parametrized by a subset $\mathcal{C}_1 \subseteq [m]$. Every party P_i, with $i \in \mathcal{C}_1$, hands in a value, and the functionality distributes the majority of these inputs to all the parties.

4.1.2 Constructing Non-Expander Protocols

High-level overview of the protocol. Having defined the ideal functionalities, we are ready to present the main lemma. We start by describing the underlying idea behind the non-expanding MPC protocol π_n^{ne} (Figure 3). At the onset of the protocol, the party-set is partitioned into two subsets of size $m = n/2$, a left subset and a right subset (see Figure 2). The left subset will invoke the Elect-and-Share functionality, that elects two subsets $\mathcal{C}_1, \mathcal{C}_2 \subseteq [m]$ of size $n' = \log^2(n)$. The parties in the left subset corresponding to $\mathcal{C}_1$ and the parties in the right subset corresponding to $\mathcal{C}_2$ will form a "bridge." The parties in $\mathcal{C}_1$ will receive shares of all inputs values of parties in the left subset, and transfer them to $\mathcal{C}_2$. Next, the right subset of parties will invoke the Reconstruct-and-Compute functionality, where each party hands its input value, and parties in $\mathcal{C}_2$ additionally provide the shares they received from $\mathcal{C}_1$. The functionality reconstructs the left-subset's inputs, computes the function f and hands the output to the right subset. Finally, $\mathcal{C}_2$ transfers the output value to $\mathcal{C}_1$, and the left subset invoke the Output-Distribution functionality in order to distribute the output value to all the parties.

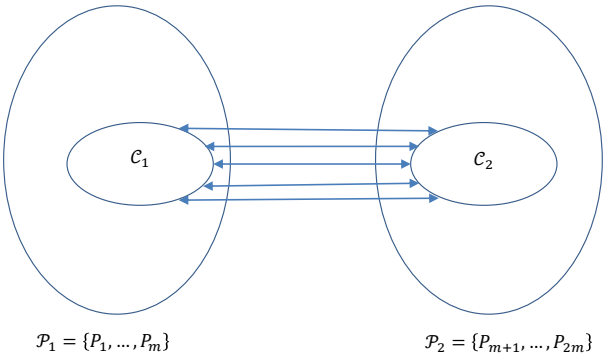

Figure 2: The non-expanding subsets in the protocol π^{ne}. The sets $\mathcal{C}_1$ and $\mathcal{C}_2$ are of poly-logarithmic size and the sets $\mathcal{P}_1$ and $\mathcal{P}_2$ are of linear size. The number of edges between $\mathcal{P}_1$ and $\mathcal{P}_2$ is poly-logarithmic.

Protocol π_n^{ne}

- **Hybrid Model:** The protocol is defined in the $(f_{\mathsf{elect\text{-}share}}, f_{\mathsf{recon\text{-}compute}}, f_{\mathsf{out\text{-}dist}})$-hybrid model.
- **Common Input:** A (t', n') ECSS scheme $(\mathsf{Share}, \mathsf{Recon})$, a signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, and a partition of the party-set $\mathcal{P} = \{P_1, \ldots, P_n\}$ into $\mathcal{P}_1 = \{P_1, \ldots, P_m\}$ and $\mathcal{P}_2 = \mathcal{P} \setminus \mathcal{P}_1$.
- **PKI:** Every party P_i, for $i \in [n]$, has signature keys $(\mathtt{sk}_i, \mathtt{vk}_i)$; the signing key $\mathtt{sk}_i$ is private, whereas the vector of verification keys $(\mathtt{vk}_1, \ldots, \mathtt{vk}_n)$ is public and known to all parties.
- **Private Input:** Every party P_i, for $i \in [n]$, has private input $x_i \in \{0,1\}^*$.
- **The Protocol:**

1. The parties in $\mathcal{P}_1$ invoke $f_{\mathsf{elect\text{-}share}}^{(t',n')}$, where every $P_i \in \mathcal{P}_1$ sends input $(x_i, \mathtt{sk}_i)$, and receives back output consisting of a committee $\mathcal{C}_1 = \{i_{(1,1)}, \ldots, i_{(1,n')}\} \subseteq [m]$. Every party P_i with $i = i_{(1,j)} \in \mathcal{C}_1$, receives additional output consisting of a signature σ_1 on $\mathcal{C}_1$, a committee $\mathcal{C}_2 = \{i_{(2,1)}, \ldots, i_{(2,n')}\} \subseteq [m]$, a signature σ_2 on $\mathcal{C}_2$, and a vector $\boldsymbol{s}_j = (s_1^j, \ldots, s_m^j)$.
2. For every $j \in [n']$, party $P_{i_{(1,j)}}$ sends $(\mathcal{C}_1, \sigma_1, \mathcal{C}_2, \sigma_2)$ to every party in $\mathcal{C}_2$, and $\boldsymbol{s}_j$ only to $P_{m+i_{(2,j)}}$. A party $P_{m+i} \in \mathcal{P}_2$ that receives a message $(\mathcal{C}_1, \sigma_1, \mathcal{C}_2, \sigma_2)$ from $P_j \in \mathcal{P}_1$ will discard the message in the following cases:
   (a) If $i \notin \mathcal{C}_2$ or $j \notin \mathcal{C}_1$.
   (b) If $\mathsf{Verify}_{\mathtt{vk}_1,\ldots,\mathtt{vk}_m}(\mathcal{C}_1, \sigma_1) = 0$ or $\mathsf{Verify}_{\mathtt{vk}_1,\ldots,\mathtt{vk}_m}(\mathcal{C}_2, \sigma_2) = 0$.
3. The parties in $\mathcal{P}_2$ invoke $f_{\mathsf{recon\text{-}compute}}^{(\mathtt{vk}_1,\ldots,\mathtt{vk}_m)}$, where $P_{m+i} \in \mathcal{P}_2$ sends input (x_{m+i}, z_{m+i}) such that for $i \notin \mathcal{C}_2$, set $z_{m+i} = \epsilon$, and for $i = i_{(2,j)} \in \mathcal{C}_2$, set $z_{m+i} = (\mathcal{C}_2, \sigma_2, \boldsymbol{s}_j)$. Every party in $\mathcal{P}_2$ receives back output y.
4. For every $j \in [n']$, party $P_{m+i_{(2,j)}}$ sends y to party $P_{i_{(1,j)}}$. In addition, every party in $\mathcal{P}_2$ outputs y and halts.
5. The parties in $\mathcal{P}_1$ invoke $f_{\mathsf{out\text{-}dist}}^{\mathcal{C}_1}$, where party P_i, with $i \in \mathcal{C}_1$, has input y, and party P_i, with $i \notin \mathcal{C}_1$ has the empty input ϵ. Every party in $\mathcal{P}_1$ receives output y, outputs it, and halts.

Figure 3: Non-expanding MPC in the $(f_{\mathsf{elect\text{-}share}}, f_{\mathsf{recon\text{-}compute}}, f_{\mathsf{out\text{-}dist}})$-hybrid model

Lemma 4.2. *Let $f = \{f_n\}_{n \in 2\mathbb{N}}$,[8] where f_n is an n-party functionality for $n = 2m$, let $\delta > 0$, and assume that one-way functions exist. Then, in the PKI-hybrid model with secure channels, where a trusted party additionally computes the m-party functionality-ensembles $(f_{\mathsf{elect\text{-}share}}, f_{\mathsf{recon\text{-}compute}}, f_{\mathsf{out\text{-}dist}})$ tolerating $\gamma \cdot m$ corruptions, there exists a protocol ensemble π that securely computes f tolerating static PPT βn-adversaries, for $\beta < \min(1/4 - \delta, \gamma/2)$, with the following guarantees:*

1. *The communication graph of π is strongly not an expander.*
2. *Denote by f_1, f_2, f_3 the functionality-ensembles $f_{\mathsf{elect\text{-}share}}, f_{\mathsf{recon\text{-}compute}}, f_{\mathsf{out\text{-}dist}}$ (resp.). If protocol-ensembles ρ_1, ρ_2, ρ_3 securely compute f_1, f_2, f_3 (resp.) with locality $\ell_\rho = \ell_\rho(m)$, then $\pi^{f_i \to \rho_i}$ (where every call to f_i is replaced by an execution of ρ_i) has locality $\ell = 2 \cdot \ell_\rho + \log^2(n)$.*

[8] For simplicity, we consider even n's. Extending the statement to any n is straightforward, however, adds more details.

Proof. For $m \in \mathbb{N}$ and $n = 2m$, we construct the n-party protocol π_n^{ne} (see Figure 3) in the $(f_{\mathsf{elect\text{-}share}}, f_{\mathsf{recon\text{-}compute}}, f_{\mathsf{out\text{-}dist}})$-hybrid model. The parameters for the protocol are $n' = \log^2(n)$ and $t' = (1/2 - \delta) \cdot n'$. We start by proving in Proposition 4.3 that the protocol π_n^{ne} securely computes f_n. Next, in Proposition 4.4 we prove that the communication graph of π^{ne} is strongly not an expander. Finally, in Proposition 4.5 we prove that by instantiating the functionalities $(f_{\mathsf{elect\text{-}share}}, f_{\mathsf{recon\text{-}compute}}, f_{\mathsf{out\text{-}dist}})$ using low-locality protocols, the resulting protocol has low locality.

Proposition 4.3. *For sufficiently large n, the protocol π_n^{ne} securely computes the function f_n, tolerating static PPT βn-adversaries, in the $(f_{\mathsf{elect\text{-}share}}, f_{\mathsf{recon\text{-}compute}}, f_{\mathsf{out\text{-}dist}})$-hybrid model.*

The proof of Proposition 4.3 can be found in the full version [11].

Proposition 4.4. *The communication graph of the Protocol π^{ne} is strongly not an expander, facing static PPT βn-adversaries.*

Proof. For $n = 2m$, consider the set $\mathcal{P}_1 = \{P_1, \ldots, P_m\}$ and its complement $\mathcal{P}_2 = \mathcal{P} \setminus \mathcal{P}_1$. For any input vector and for every static PPT βn-adversary it holds that with overwhelming probability that $|\mathcal{P}_1| = n/2$ and $\mathsf{edges}(\mathcal{P}_1, \mathcal{P}_2) = \log^2(n) \cdot \log^2(n)$. Therefore, considering the function

$$f(n) = \frac{2\log^4(n)}{n},$$

it holds that $f(n) \in o(1)$ and $f(n)$ is an upper bound of the edge expansion of π^{ne}. We conclude that the communication graph of π^{ne} is strongly not an expander. □

Proposition 4.5. *Let ρ_1, ρ_2, ρ_3, and $\pi^{f_i \to \rho_i}$ be the protocols defined in Lemma 4.2, and let $\ell_\rho = \ell_\rho(m)$ be the upper bound of the locality of ρ_1, ρ_2, ρ_3. Then $\pi^{f_i \to \rho_i}$ has locality $\ell = 2 \cdot \ell_\rho + \log^2(n)$.*

Proof. Every party in $\mathcal{P}_1$ communicates with ℓ_ρ parties when executing ρ_1, and with at most another ℓ_ρ parties when executing ρ_3. In addition, every party in $\mathcal{C}_1$ communicates with all $n' = \log^2(n)$ parties in $\mathcal{C}_2$. Similarly, every party in $\mathcal{P}_2$ communicates with ℓ_ρ parties when executing ρ_2, and parties in $\mathcal{C}_2$ communicates with all n' parties in $\mathcal{C}_1$. It follows that maximal number of parties that a party communicates with during the protocol is $2 \cdot \ell_\rho + \log^2(n)$. □

This concludes the proof of Lemma 4.2. □

4.2 Additional Results

4.2.1 Information-Theoretic Security

The protocol in Section 4.1 relies on digital signatures, hence, security is guaranteed only in the presence of computationally bounded adversaries. Next, we gain security facing all-powerful adversaries by using information-theoretic signatures. We prove the following theorem in the full version [11].

Theorem 4.6. *Let $f = \{f_n\}_{n \in \mathbb{N}}$ be an ensemble of functionalities and let $\delta > 0$. Then, the following holds in the IT-PKI-hybrid model with secure channels:*

1. *Let $\beta < 1/4 - \delta$ and let $t = \beta \cdot n$. Then, f can be t-securely computed by a protocol ensemble π tolerating static $t(n)$-adversaries such that the communication graph of π is strongly not an expander.*

2. *Let $\beta < 1/12 - \delta$ and let $t = \beta \cdot n$. Then, f can be t-securely computed by a protocol ensemble π tolerating static $t(n)$-adversaries such that (1) the communication graph of π is strongly not an expander, and (2) the locality of π is poly-logarithmic in n.*

Proof. The theorem follows from an information-theoretic variant of Lemma 4.2, stated and proven in the full version [11], by instantiating the hybrid functionalities using appropriate MPC protocols.

- The first part follows using honest-majority MPC protocols that exist in the secure-channels model, e.g., the protocol of Rabin and Ben-Or [60].
- In the full version [11] we prove information-theoretic variant of the low-locality MPC protocol of Boyle et al. [12] in the IT-PKI model with secure channels that tolerates $t < (1/6 - \delta) \cdot n$ static corruptions. The second part follows from that protocol.

□

4.2.2 Adaptive Corruptions

In this section, we focus on the adaptive setting, where the adversary can corrupt parties dynamically, based on information gathered during the course of the protocol.

Adjusting Lemma 4.2 to the adaptive setting is not straightforward, since once the subsets $\mathcal{C}_1$ and $\mathcal{C}_2$ are known to the adversary he can completely corrupt them. A first attempt to get around this problem, is not to reveal the entire subsets in the output of the Elect-and-Share functionality, but rather, let each party in $\mathcal{C}_1$ learn the identity of a single party in $\mathcal{C}_2$ with which he will communicate. This way, if a party in $\mathcal{C}_1$ (resp. $\mathcal{C}_2$) gets corrupted, only one additional party in $\mathcal{C}_2$ (resp. $\mathcal{C}_1$) is revealed to the adversary. This solution comes with the price of tolerating a smaller fraction of corrupted parties, namely, $(1/8 - \delta)$ fraction.

This solution, however, is still problematic in the adaptive setting if the adversary can monitor the communication lines, even when they are completely private (as in the secure-channels setting). The reason is that once the adversary sees the communication that is sent between $\mathcal{C}_1$ and $\mathcal{C}_2$ he can completely corrupt both subsets. This problem is inherent when the communication lines are visible to the adversary, therefore, we turn to the hidden-channels setting that was used by Chandran et al. [19], where the adversary does not learn whether a message is sent between two honest parties.

Theorem 4.7. *Let $f = \{f_n\}_{n \in \mathbb{N}}$ be an ensemble of functionalities, let $\delta > 0$, let $\beta < 1/8 - \delta$, and let $t = \beta \cdot n$. Then, the following holds in the hidden-channels model:*

1. *Assuming the existence of one-way functions, f can be securely computed by a protocol ensemble π in the PKI model tolerating adaptive PPT $t(n)$-adversaries such that the communication graph of π is strongly not an expander.*
2. *Assume in addition the SKI model and non-committing encryption. Then, f can be securely computed by a protocol ensemble π in the PKI model tolerating adaptive PPT $t(n)$-adversaries such that (1) the communication graph of π is strongly not an expander, and (2) the locality of π is poly-logarithmic in n.*
3. *f can be securely computed by a protocol ensemble π in the IT-PKI model tolerating adaptive $t(n)$-adversaries such that the communication graph of π is strongly not an expander.*

Proof. The theorem follows from an adaptively secure variant of Lemma 4.2, stated and proven in the full version [11], by instantiating the hybrid functionalities using existing MPC protocols from the literature.

- The first part follows using an adaptively secure honest-majority MPC protocol in the secure-channels model, e.g., Cramer et al. [24] or Damgård and Ishai [25].
- The second part follows using the adaptively secure low-locality MPC protocol of Chandran et al. [19].
- The third part follows using information-theoretic signatures via the same adjustments that were employed in Section 4.2.1, and using the information-theoretic protocol of Cramer et al. [24].

□

5 Expansion is Necessary for Correct Computation

In this section, we show that in certain natural settings there exist functionalities such that the final communication graph of any MPC protocol that securely computes them *must* be an expander. In fact, we prove a stronger statement, that removing a sublinear number of edges from such graphs will not disconnect them. We consider the plain model, in which parties do not have any trusted setup assumptions, a PPT adaptive adversary, and focus on parallel multi-valued broadcast (also known as interactive consistency [59]), where every party has an input value, and all honest parties agree on a common output vector, such that if P_i is honest then the i'th coordinate equals P_i's input. In particular, our proof does not rely on any privacy guarantees of the protocol, merely its correctness.

For simplicity, and without loss of generality, we assume the security parameter is the number of parties n.

Definition 5.1 (parallel broadcast)**.** *A protocol ensemble $\pi = \{\pi_n\}_{n\in\mathbb{N}}$ is a $t(n)$-resilient, parallel broadcast protocol with respect to input space $\{\{0,1\}^n\}_{n\in\mathbb{N}}$, if there exists a negligible function $\mu(n)$, such that for every $n \in \mathbb{N}$, every party P_i in π_n has input $x_i \in \{0,1\}^n$ and outputs a vector of n values $\boldsymbol{y}_i = (y_1^i, \ldots, y_n^i)$ such that the following is satisfied, except for probability $\mu(n)$. Facing any adaptive, malicious PPT adversary that dynamically corrupts and controls a subset of parties $\{P_j\}_{j\in\mathcal{I}}$, with $\mathcal{I} \subseteq [n]$ of size $|\mathcal{I}| \leq t(n)$, it holds that:*

- ***Agreement.*** *There exists a vector $\boldsymbol{y} = (y_1, \ldots, y_n)$ such that for every party P_i that is honest at the conclusion of the protocol it holds that $\boldsymbol{y}_i = \boldsymbol{y}$.*
- ***Validity.*** *For every party P_i that is honest at the conclusion of the protocol it holds that the i'th coordinate of the common output equals his input value, i.e., $y_i = x_i$.*

Recall that a connected graph is k-edge-connected if it remains connected whenever fewer than k edges are removed. We are now ready to state the main result of this section. We note that as opposed to Section 4.2.2, where we considered adaptive corruptions in the hidden-channels model, this section considers the *parallel secure message transmission (SMT)* model, formally defined in Section 5.1, where the adversary is aware of communication between honest parties, but not of the message content.

Theorem 5.2. *Let $\beta > 0$ be a fixed constant, let $t(n) = \beta \cdot n$, and let $\pi = \{\pi_n\}_{n\in\mathbb{N}}$ be a $t(n)$-resilient, parallel broadcast protocol with respect to input space $\{\{0,1\}^n\}_{n\in\mathbb{N}}$, in the parallel SMT hybrid model (in the computational setting, tolerating an adaptive, malicious PPT adversary). Then, the communication graph of π must be $\alpha(n)$-edge-connected, for every $\alpha(n) \in o(n)$.*

From Theorem 5.2 and Lemma 3.13 (stating that if π is strongly not an expander then there must exist a sublinear cut in the graph) we get the following corollary.

Corollary 5.3. *Consider the setting of Theorem 5.2. If the communication graph of π is strongly not an expander (as per Definition 3.12), then π is not a $t(n)$-resilient parallel broadcast protocol.*

The remainder of this section goes towards proving Theorem 5.2. We start by presenting the communication model in Section 5.1. In Section 5.2, we prove a graph-theoretic theorem that will be used in the core of our proof and may be of independent interest. Then, in Section 5.3 we present the proof of Theorem 5.2. Some of the proofs are deferred to the full version [11].

5.1 The Communication Model

We consider secure communication channels, where the adversary can see that a message has been sent but not its content (in contrast to the hidden-communication model, used in Section 4.2.2, where the communication between honest parties was hidden from the eyes of the adversary). A standard assumption when considering adaptive corruptions is that in addition to being notified that an honest party sent a message, the adversary can corrupt the sender *before* the receiver obtained the message, learn the content of the message, and replace it with another message of its choice that will be delivered to the receiver. Although the original modular composition framework [15] does not give the adversary such power, this ability became standard after the introduction of the *secure message transmission (SMT)* functionality in the UC framework [16]. As we consider *synchronous* protocols, we use the parallel SMT functionality that was formalized in [22, 23].[9]

Definition 5.4 (parallel SMT)**.** *The parallel secure message transmission functionality f_{psmt} is a two-phase functionality. For every $i, j \in [n]$, the functionality initializes a value x_j^i to be the empty string ϵ (the value x_j^i is the message to be sent from P_i to P_j).*

- ***The input phase.*** *Every party P_i sends a vector of n messages $(v_1^i, \ldots, v_n^i)$. The functionality sets $x_j^i = v_j^i$, and provides the adversary with leakage information on the input values. As we consider rushing adversaries, who can determine the messages to be sent by the corrupted parties* after *receiving the messages sent by the honest parties, the leakage function should leak the messages that are to be delivered from honest parties to corrupted parties. Therefore, the leakage function is*

$$l_{\mathsf{psmt}}\left((x_1^1, \ldots, x_n^1), \ldots, (x_1^n, \ldots, x_n^n)\right) = \left(y_1^1, y_2^1, \ldots, y_{n-1}^n, y_n^n\right),$$

where $y_j^i = |x_j^i|$ in case P_j is honest and $y_j^i = x_j^i$ in case P_j is corrupted.

We consider adaptive corruptions, and so, the adversary can corrupt an honest party during the input phase based on this leakage information, and send a new input on behalf of the corrupted party (note that the message are not delivered yet to the honest parties).

[9] We note that by considering secure channels, that hide the content of the messages from the adversary, we obtain a stronger lower bound than, for example, authenticated channels.

- ***The output phase.*** *In the second phase, the messages are delivered to the parties, i.e., party P_i receives the vector of messages $(x_i^1, \ldots, x_i^n)$.*

In addition, we assume that the parties do not have any trusted-setup assumption.

5.2 A Graph-Theoretic Theorem

Our lower-bound proof is based on the following graph-theoretic theorem, which we believe may be of independent interest. We show that every graph in which every node has a linear degree, can be partitioned into a constant number of linear-size sets that are pairwise connected by sublinear many edges. These subsets are "minimal cuts" in the sense that every sublinear cut in the graph is a union of some of these subsets. The proof of the theorem given in the full version [11].

Definition 5.5 ((α, d)-partition)**.** *Let $G = (V, E)$ be a graph of size n. An (α, d)-partition of G is a partition $\Gamma = (U_1, \ldots, U_\ell)$ of V that satisfies the following properties:*

1. *For every $i \in [\ell]$ it holds that $|U_i| \geq d$.*
2. *For every $i \neq j$, there are at most α edges between U_i and U_j, i.e., $|\mathsf{edges}_G(U_i, U_j)| \leq \alpha$.*
3. *For every $S \subseteq V$ such that $\{S, \bar{S}\}$ is an α-cut, i.e., $|\mathsf{edges}_G(S)| \leq \alpha$, it holds that there exists a subset $J \subsetneq [\ell]$ for which $S = \bigcup_{j \in J} U_j$ and $\bar{S} = \bigcup_{j \in [\ell] \setminus J} U_j$.*

In Theorem 5.6 we first show that if every node in the graph has a linear degree $d(n)$, and $\alpha(n)$ is sublinear, then for sufficiently large n there exists an $(\alpha(n), d(n))$-partition of the graph, and moreover, the partition can be found in polynomial time.

Theorem 5.6. *Let $c > 1$ be a constant integer, let $\alpha(n) \in o(n)$ be a fixed sublinear function in n, and let $\{G_n\}_{n \in \mathbb{N}}$ be a family of graphs, where $G_n = ([n], E_n)$ is defined on n vertices, and every vertex of G_n has degree at least $\frac{n}{c} - 1$. Then, for sufficiently large n it holds that:*

1. *There exists a $(\alpha(n), n/c)$-partition of G_n, denoted Γ; it holds that $|\Gamma| \leq c$.*
2. *A $(\alpha(n), n/c)$-partition Γ of G_n can be found in (deterministic) polynomial time, given the $n \times n$ adjacency matrix of G_n.*

Note that if for every n there exists an $\alpha(n)$-cut in G_n, then it immediately follows that $|\Gamma| > 1$, i.e., the partition is not the trivial partition of the set of all nodes.

5.3 Proof of Main Theorem (Theorem 5.2)

High-level overview of the attack. For $n \in \mathbb{N}$, consider an execution of the alleged parallel broadcast protocol π_n over uniformly distributed n-bit input values for the parties $(x_1, \ldots, x_n) \in_R (\{0,1\}^n)^n$. We define two ensembles of adversarial strategies $\{\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}\}_{n \in \mathbb{N}}$ and $\{\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}\}_{n \in \mathbb{N}}$ (described in full in Section 5.3.1).

The adversary $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ corrupts a random party P_{i^*}, and simulates an honest execution on a random input $\tilde{x}_{i^*}$ until P_{i^*} has degree $\beta/4$. Next, $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ switches the internal state of P_{i^*} with a view that is consistent with an honest execution over the initial input x_{i^*}, where all other parties have random inputs. The adversary $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ continues by computing the $(\alpha(n), n/c)$-partition $\{U_1, \ldots, U_\ell\}$ of the communication graph, (where c is a constant depending only on β – this is

possible due to Theorem 5.6), and blocking every message that is sent between every pair of U_i's. In Lemma 5.7, we show that there exist honest parties that at the conclusion of the protocol have received a bounded amount of information on the initial input value x_{i^*}.

The second adversary, $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$, is used for showing that under the previous attack, every honest party will eventually output the initial input value x_{i^*} (Lemma 5.8). This is done by having $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ corrupt all the neighbors of P_{i^*}, while keeping P_{i^*} honest, and simulate the previous attack to the remaining honest parties.

We show that there exist honest parties whose view is identically distributed under both attacks, and since they output x_{i^*} in the latter, they must also output x_{i^*} in the former. By combining both of these lemmata, we then derive a contradiction.

Proof of Theorem 5.2. First, since we consider the plain model, without any trusted setup assumptions, known lower bounds [59, 54, 33] state that parallel broadcast cannot be computed for $t(n) \geq n/3$, therefore, we can focus on $0 < \beta < 1/3$, i.e., the case where $t(n) = \beta \cdot n < n/3$.

Assume toward a contradiction that π is $t(n)$-resilient parallel broadcast protocol in the above setting, and that there exists a sublinear function $\alpha(n) \in o(n)$ such that the communication graph of π is not $\alpha(n)$-edge-connected, i.e., for sufficiently large n there exists a cut $\{S_n, \bar{S}_n\}$ of weight at most $\alpha(n)$.

Notations. We start by defining a few notations. For a fixed n,[10] consider the following independently distributed random variables

$$\textsc{InputsAndCoins} = \left(X_1, \ldots, X_n, R_1, \ldots, R_n, \tilde{X}_1, \ldots, \tilde{X}_n, \tilde{R}_1, \ldots, \tilde{R}_n, I^*\right),$$

where for every $i \in [n]$, each X_i and $\tilde{X}_i$ take values uniformly at random in the input space $\{0,1\}^n$, each R_i and $\tilde{R}_i$ take values uniformly at random in $\{0,1\}^*$, and I^* takes values uniformly at random in $[n]$. During the proof, (X_i, R_i) represent the pair of input and private randomness of party P_i, whereas $(\tilde{X}_1, \ldots, \tilde{X}_n, \tilde{R}_1, \ldots, \tilde{R}_n, I^*)$ correspond to the random coins of the adversary (used in simulating the two executions towards the honest parties). Unless stated otherwise, all probabilities are taken over these random variables.

Let $\textsc{RedExec}$ be a random variable defined as

$$\textsc{RedExec} := \left(X_{-I^*}, \tilde{X}_{I^*}, R_{-I^*}, \tilde{R}_{I^*}\right).$$

That is, $\textsc{RedExec}$ contains X_i and R_i for $i \in [n] \setminus \{I^*\}$, along with $\tilde{X}_{I^*}$ and $\tilde{R}_{I^*}$. We denote by the "red execution" an *honest* protocol execution when the inputs and private randomness of the parties are $(X_{-I^*}, \tilde{X}_{I^*}, R_{-I^*}, \tilde{R}_{I^*})$. We denote by the "blue execution" an *honest* protocol execution when the inputs and private randomness of the parties are $(\tilde{X}_{-I^*}, X_{I^*}, \tilde{R}_{-I^*}, R_{I^*})$. Note that such a sample fully determines the view and transcript of all parties in an *honest* simulated execution of π_n.

Let $\textsc{FinalCut}^{\mathsf{corrupt}}$ be a random variable defined over $2^{[n]} \cup \{\perp\}$. The distribution of $\textsc{FinalCut}^{\mathsf{corrupt}}$ is defined by running protocol π until its conclusion with adversary $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ (defined in Section 5.3.1) on inputs and coins sampled according to $\textsc{InputsAndCoins}$. If at the conclusion of the protocol there is no $\alpha(n)$-cut in the graph, then set the value of $\textsc{FinalCut}^{\mathsf{corrupt}}$

[10] For clarity, we denote the random variables without the notation n.

to be $\perp$; otherwise, set the value to be the identity of the smallest $\alpha(n)$-cut $\{S, \bar{S}\}$ in the communication graph according to some canonical ordering on the $\alpha(n)$-cuts. We will prove that conditioned on the value of RedExec, then $\text{FinalCut}^{\mathsf{corrupt}}$ can only take one of a constant number of values depending only on β (and not on n).

Let $\mathcal{E}_1$ denote the event that P_{I^*} is the last among all the parties to reach degree $\beta n/4$ in both the red and the blue honest executions of the protocol. More precisely, the event that P_{I^*} reaches degree $\beta n/4$ in both executions, and if it has reached this degree in round ρ in the red (blue) execution, then all parties in the red (blue) execution have degree at least $\beta n/4$ in round ρ.

Let $\mathcal{E}_2$ denote the event that the degree of P_{I^*} reaches $\beta n/4$ in the red execution before, or at the same round as, in the blue execution. Note that $\mathcal{E}_1$ and $\mathcal{E}_2$ are events with respect to two honest executions of the protocol (the red execution and the blue execution) that are defined according to InputsAndCoins. In the adversarial stategies that are used in the proof, the corrupted parties operate in a way that indeed induces the red and blue executions, and so, the events $\mathcal{E}_1$ and $\mathcal{E}_2$ are well defined in an execution of the protocol with those adversarial strategies.

In Section 5.3.1, we formally describe two adversarial strategies, $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ and $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$. We denote by $Y_{I^*}^{\mathsf{corrupt}}$, respectively $Y_{I^*}^{\mathsf{honest}}$, the random variable that corresponds to the I^*'th coordinate of the common output of honest parties, when running over random inputs with adversarial strategy $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$, respectively $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$.

Proof structure. Our proof follows from two main steps. In Lemma 5.7, stated in Section 5.3.2, we show that in an execution of π_n on random inputs with adversary $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$, it holds that (1) $\Pr[\mathcal{E}_1 \cap \mathcal{E}_2] \geq 1/2n^2 - \mathsf{negl}(n)$, and that (2) conditioned on the event $\mathcal{E}_1 \cap \mathcal{E}_2$, there exists an honest party P_{j^*} such that X_{I^*}, conditioned on $\mathcal{E}_1 \cap \mathcal{E}_2$ and on the view of P_{j^*} at the conclusion of the protocol, still retains at least $n/4$ bits of entropy. This means, in particular, that P_{j^*} will output the value X_{I^*} only with negligible probability. Hence, by *agreement*, the probability for any of the honest parties to output X_{I^*} in an execution with $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ is negligible. In particular,

$$\Pr\left[Y_{I^*}^{\mathsf{corrupt}} = X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2\right] = \mathsf{negl}(\kappa).$$

In Lemma 5.8, stated in Section 5.3.2, we show that in an execution of π on random inputs with adversary $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$, it holds that (1) with overwhelming probability all honest parties output X_{i^*} (this holds by correctness, since P_{I^*} remains honest), i.e.,

$$\Pr\left[Y_{I^*}^{\mathsf{honest}} = X_{I^*}\right] \geq 1 - \mathsf{negl}(\kappa),$$

and that (2) conditioned on the event $\mathcal{E}_1 \cap \mathcal{E}_2$, there exists an honest party whose view is identically distributed as in an execution with $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$, therefore,

$$\Pr\left[Y_{I^*}^{\mathsf{corrupt}} = Y_{I^*}^{\mathsf{honest}} \mid \mathcal{E}_1 \cap \mathcal{E}_2\right] \geq 1 - \mathsf{negl}(\kappa).$$

From the combination of the two lemmata, we derive a contradiction. □

5.3.1 Defining Adversarial Strategies

As discussed above, the main idea behind the proof is to construct two dual adversarial strategies that will show that on the one hand, the output of all honest parties must contain the initial value

of a randomly chosen corrupted party, and on the other hand, there exist parties that only receive a bounded amount of information on this value during the coarse of the protocol.

We use the following notation for defining the adversarial strategies. Virtual parties that only exist in the head of the adversary are denoted with "tilde". In particular, for a random $i^* \in [n]$, we denote by $\tilde{P}_{i^*}$ a virtual party that emulates the role of P_{i^*} playing with the real parties using a random input in the so-called "red execution," and by $\{\tilde{Q}_i\}_{i \neq i^*}$ virtual parties that emulate an execution over random inputs towards P_{i^*}.[11]

The adversary $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$. At a high level, the adversary $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ chooses a random $i^* \in [n]$ and isolates the honest party P_{i^*}. The adversary $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ consists of three phases. In Phase I, $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ induces two honestly distributed executions.

- The first (red) execution is set by simulating an honest execution of a virtual party $\tilde{P}_{i^*}$ over a random input $\tilde{x}_{i^*}$ towards all other parties. The adversary corrupts any party that sends a message to P_{i^*}, blocks its message, and simulates $\tilde{P}_{i^*}$ receiving this message. Whenever $\tilde{P}_{i^*}$ should send a message to some P_j, the adversary corrupts the P_j, and instructs him to proceed as if he received the intended message from $\tilde{P}_{i^*}$.

- For the second (blue) execution, $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ emulates a virtual execution with virtual parties $(\tilde{Q}_1, \ldots, \tilde{Q}_n) \setminus \{\tilde{Q}_{i^*}\}$ on random inputs towards the honest party P_{i^*}. To do so, whenever P_{i^*} sends a message to P_j in the real execution, the adversary corrupts P_j, instructing him to ignore this message, and simulates this message from P_{i^*} to $\tilde{Q}_j$ in the virtual execution (that is running in the head of the adversary). Whenever a party $\tilde{Q}_j$ sends a message to P_{i^*} in the virtual execution, the adversary corrupts the real party P_j and instructs him to send this message to P_{i^*} in the real execution.

Phase II begins when the degree of P_{i^*} in the *red* execution is at least $(\beta/4) \cdot n$; if P_{i^*} reaches this threshold faster in the *blue* execution, the attack fails. Phase III begins when the degree of P_{i^*} in the *real* execution is at least $(\beta/4) \cdot n$.

Ideally, Phase I will continue until all parties in the real execution have a linear degree, and before the adversary will use half of his "corruption budget", i.e., $(\beta/2) \cdot n$. This would be the case if we were to consider a single honest execution of the protocol, since we show that there always exists a party that will be the last to reach the linear-degree threshold with a noticeable probability. However, as the attack induces two *independent* executions, in which the degree of the parties can grow at different rates, care must be taken. We ensure that even though P_{i^*} runs in the blue execution, by the time P_{i^*} will reach the threshold, all other parties (that participate in the red execution) will already have reached the threshold, and can be partitioned into "minimal" $\alpha(n)$-cuts, as follows.

The adversary allocates $(\beta/4) \cdot n$ corruptions for the red execution and $(\beta/4) \cdot n$ corruptions for the blue execution. We show that with a noticeable probability, once $\tilde{P}_{i^*}$ has degree $(\beta/4) \cdot n$ in the red execution, all other parties in the red execution also have high degree. Consider the communication graph of the red execution without the virtual party $\tilde{P}_{i^*}$ (i.e., after removing the node i^* and its edges); by Theorem 5.6 there exists an $(\alpha(n), (\beta/4)n - 1)$ partition of this graph into

[11] Following the *red pill blue pill* paradigm, in the adversarial strategy $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$, the chosen party P_{i^*} is participating (without knowing it) in the *blue* execution, which is a fake execution that does not happen in the real world. The real honest parties participate in the *red* execution, where the adversary simulates P_{i^*} by running a virtual party.

a constant number of linear-size subsets that are connected with sublinear many edges, denoted $\Gamma = \{U_1, \ldots, U_\ell\}$ (in particular, this partition is independent of x_{i^*}). In Phase II, the adversary continues blocking outgoing messages from P_{i^*} towards the real honest parties, until the degree of P_{i^*} in the real execution is $\beta n/4$. In addition, $\mathcal{A}_n^{\text{honest-}i^*}$ blocks any message that is sent between two subsets in the partition, by corrupting the recipient and instructing him to ignore messages from outside of his subset.

In Phase III, which begins when P_{i^*} has high degree in the real execution, the adversary adds P_{i^*} to one of the subsets in the partition, in which P_{i^*} has many neighbors, and continues to block messages between different subsets in the partition until the conclusion of the protocol.

We note that special care must be taken in the transition between the phases, since such a transition can happen in a middle of a round, after processing some of the messages, but not all. Indeed, if the transition to the next phase will happen at the end of the round, the adversary may need to corrupt too many parties. For this reason, in Phases I and II, we analyze the messages to and from P_{i^*} one by one, and check whether the threshold has been met after each such message.

The adversary $\mathcal{A}_n^{\text{corrupt-}i^*}$. The adversary $\mathcal{A}_n^{\text{corrupt-}i^*}$ corrupts the randomly chosen party P_{i^*}, and emulates the operations of an honest P_{i^*} that is being attacked by $\mathcal{A}_n^{\text{honest-}i^*}$.

In Phase I, the adversary $\mathcal{A}_n^{\text{corrupt-}i^*}$ induces two honestly distributed executions, by simulating an honest execution of a virtual party $\tilde{P}_{i^*}$ over a random input $\tilde{x}_{i^*}$ towards all other honest parties (the red execution), and furthermore, runs in its mind a virtual execution over the initial input x_{i^*} and random inputs $\tilde{x}_i$ for $i \neq i^*$ (the blue execution). This phase continues until $\tilde{P}_{i^*}$ has degree $\beta n/4$ in the red execution (no parties other than P_{i^*} are being corrupted). If all other parties in the red execution have high degree, then the adversary finds the partition of the red graph as in the previous attack (the partition is guaranteed by Theorem 5.6).

In Phase II, the adversary continues simulating the corrupted P_{i^*} towards the real honest parties until the degree of P_{i^*} in the real execution is $\beta n/4$; however, his communication is based on the view in the blue execution at the end of Phase I (this is no longer an honest-looking execution). During this phase, $\mathcal{A}_n^{\text{corrupt-}i^*}$ blocks any message that is sent between two subsets in the partition.

In Phase III, that begins when P_{i^*} has high degree (in the real execution), $\mathcal{A}_n^{\text{corrupt-}i^*}$ adds P_{i^*} to one of the subsets in the partition, in which P_{i^*} has many neighbors, and continues to block messages between different subsets in the partition until the conclusion of the protocol.

5.3.2 The Core Lemmata

In the full version [11] we prove the following core lemmata that conclude the proof of the theorem.

Lemma 5.7. *Consider an execution of π_n on random inputs $(X_1, \ldots, X_n)$ for the parties with adversary $\mathcal{A}_n^{\text{corrupt-}i^*}$, and the events $\mathcal{E}_1$ and $\mathcal{E}_2$ as defined in Section 5.3. Then, it holds that:*

1. $\Pr[\mathcal{E}_1 \cap \mathcal{E}_2] \geq 1/2n^2 - \mathsf{negl}(n)$.
2. *Conditioned on the event $\mathcal{E}_1 \cap \mathcal{E}_2$, there exists an honest party P_{J^*} such that*

$$H(X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2, \text{VIEW}_{J^*}^{\text{corrupt}}) \geq n/4,$$

where $\text{VIEW}_{J^*}^{\text{corrupt}}$ is the random variable representing the view of P_{J^*} at the end of the protocol.

Lemma 5.8. *Consider an execution of π_n on random inputs $(X_1, \ldots, X_n)$ for the parties with adversary $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$. Then, conditioned on the event $\mathcal{E}_1 \cap \mathcal{E}_2$ it holds that:*

1. *The I^*'th coordinate of the common output $Y_{I^*}^{\mathsf{honest}}$ equals the initial input X_{I^*} of P_{I^*}, except for negligible probability, i.e.,*

$$\Pr\left[Y_{I^*}^{\mathsf{honest}} = X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2\right] \geq 1 - \mathsf{negl}(n).$$

2. *The I^*'th coordinate of the common output $Y_{I^*}^{\mathsf{honest}}$ in an execution with $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ equals the I^*'th coordinate of the common output $Y_{I^*}^{\mathsf{corrupt}}$ in an execution with $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$, except for negligible probability, i.e.,*

$$\Pr\left[Y_{I^*}^{\mathsf{honest}} = Y_{I^*}^{\mathsf{corrupt}} \mid \mathcal{E}_1 \cap \mathcal{E}_2\right] \geq 1 - \mathsf{negl}(n).$$

References

[1] S. Agarwal, R. Cramer, and R. de Haan. Asymptotically optimal two-round perfectly secure message transmission. In *Advances in Cryptology – CRYPTO 2006*, pages 394–408, 2006.

[2] A. Akavia and T. Moran. Topology-hiding computation beyond logarithmic diameter. In *Advances in Cryptology – EUROCRYPT 2017, part III*, pages 609–637, 2017.

[3] A. Akavia, R. LaVigne, and T. Moran. Topology-hiding computation on all graphs. In *Advances in Cryptology – CRYPTO 2017, part I*, pages 447–467, 2017.

[4] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols (extended abstract). In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 503–513, 1990.

[5] A. Beimel. On private computation in incomplete networks. *Distributed Computing*, 19(3): 237–252, 2007.

[6] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1–10, 1988.

[7] P. Berman and J. A. Garay. Fast consensus in networks of bounded degree. *Distributed Computing*, 7(2):67–73, 1993.

[8] M. Bläser, A. Jakoby, M. Liśkiewicz, and B. Manthey. Private computation: k-connected versus 1-connected networks. *Journal of Cryptology*, 19(3):341–357, 2006.

[9] M. Bläser, A. Jakoby, M. Liśkiewicz, and B. Manthey. Privacy in non-private environments. *Theory Comput. Syst.*, 48(1):211–245, 2011.

[10] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology – EUROCRYPT 2003*, pages 416–432, 2003.

[11] E. Boyle, R. Cohen, D. Data, and P. Hubáček. Must the communication graph of MPC protocols be an expander? *Manuscript, 2018.*

[12] E. Boyle, S. Goldwasser, and S. Tessaro. Communication locality in secure multi-party computation - how to run sublinear algorithms in a distributed setting. In *Proceedings of the 10th Theory of Cryptography Conference, TCC 2013*, pages 356–376, 2013.

[13] E. Boyle, K. Chung, and R. Pass. Large-scale secure computation: Multi-party computation for (parallel) RAM programs. In *Advances in Cryptology – CRYPTO 2015, part II*, pages 742–762, 2015.

[14] N. Braud-Santoni, R. Guerraoui, and F. Huc. Fast byzantine agreement. In *Proceedings of the 32th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 57–64, 2013.

[15] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[16] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, 2001.

[17] N. Chandran, J. A. Garay, and R. Ostrovsky. Improved fault tolerance and secure computation on sparse networks. In *Proceedings of the 37th International Colloquium on Automata, Languages, and Programming (ICALP), part II*, pages 249–260, 2010.

[18] N. Chandran, J. A. Garay, and R. Ostrovsky. Edge fault tolerance on sparse networks. In *Proceedings of the 39th International Colloquium on Automata, Languages, and Programming (ICALP), part II*, pages 452–463, 2012.

[19] N. Chandran, W. Chongchitmate, J. A. Garay, S. Goldwasser, R. Ostrovsky, and V. Zikas. The hidden graph model: Communication locality and optimal resiliency with adaptive faults. In *Proceedings of the 6th Annual Innovations in Theoretical Computer Science (ITCS) conference*, pages 153–162, 2015.

[20] N. Chandran, J. A. Garay, and R. Ostrovsky. Almost-everywhere secure computation with edge corruptions. *Journal of Cryptology*, 28(4):745–768, 2015.

[21] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 11–19, 1988.

[22] R. Cohen, S. Coretti, J. A. Garay, and V. Zikas. Probabilistic termination and composability of cryptographic protocols. In *Advances in Cryptology – CRYPTO 2016, part III*, pages 240–269, 2016.

[23] R. Cohen, S. Coretti, J. Garay, and V. Zikas. Round-preserving parallel composition of probabilistic-termination cryptographic protocols. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 37:1–37:15, 2017.

[24] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Advances in Cryptology – EUROCRYPT '99*, pages 311–326, 1999.

[25] I. Damgård and Y. Ishai. Constant-round multiparty computation using a black-box pseudo-random generator. In *Advances in Cryptology – CRYPTO 2005*, pages 378–394, 2005.

[26] V. Dani, V. King, M. Movahedi, J. Saia, and M. Zamani. Secure multi-party computation in large networks. *Distributed Computing*, 30(3):193–229, 2017.

[27] D. Dolev. The byzantine generals strike again. *J. Algorithms*, 3(1):14–30, 1982.

[28] D. Dolev and R. Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

[29] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *Journal of the ACM*, 40(1):17–47, 1993.

[30] Z. Dvir and A. Wigderson. Monotone expanders: Constructions and applications. *Theory of Computing*, 6(1):291–308, 2010.

[31] C. Dwork, D. Peleg, N. Pippenger, and E. Upfal. Fault tolerance in networks of bounded degree. *SIAM Journal on Computing*, 17(5):975–988, 1988.

[32] P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997.

[33] M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1(1):26–39, 1986.

[34] M. Fitzi, M. K. Franklin, J. A. Garay, and H. V. Simhadri. Towards optimal and efficient perfectly secure message transmission. In *Proceedings of the Fourth Theory of Cryptography Conference, TCC 2007*, pages 311–322, 2007.

[35] M. K. Franklin and M. Yung. Secure hypergraphs: Privacy from partial broadcast. *SIAM Journal on Discrete Mathematics*, 18(3):437–450, 2004.

[36] J. A. Garay and Y. Moses. Fully polynomial byzantine agreement in t+1 rounds. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC)*, pages 31–41, 1993.

[37] J. A. Garay and R. Ostrovsky. Almost-everywhere secure computation. In *Advances in Cryptology – EUROCRYPT 2008*, pages 307–323, 2008.

[38] J. A. Garay, J. Katz, R. Kumaresan, and H. Zhou. Adaptively secure broadcast, revisited. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 179–186, 2011.

[39] O. Goldreich. *Foundations of Cryptography – VOLUME 2: Basic Applications*. Cambridge University Press, 2004.

[40] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 218–229, 1987.

[41] S. Halevi, Y. Lindell, and B. Pinkas. Secure computation on the web: Computing without simultaneous interaction. In *Advances in Cryptology – CRYPTO 2011*, pages 132–150, 2011.

[42] S. Halevi, Y. Ishai, A. Jain, E. Kushilevitz, and T. Rabin. Secure multiparty computation with general interaction patterns. In *Proceedings of the 7th Annual Innovations in Theoretical Computer Science (ITCS) conference*, pages 157–168, 2016.

[43] D. Harnik, Y. Ishai, and E. Kushilevitz. How many oblivious transfers are needed for secure multiparty computation? In *Advances in Cryptology – CRYPTO 2007*, pages 284–302, 2007.

[44] M. Hirt and V. Zikas. Adaptively secure broadcast. In *Advances in Cryptology – EUROCRYPT 2010*, pages 466–485, 2010.

[45] M. Hirt, U. Maurer, D. Tschudi, and V. Zikas. Network-hiding communication and applications to multi-party protocols. In *Advances in Cryptology – CRYPTO 2016, part II*, pages 335–365, 2016.

[46] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc.*, 43(4):439–561, 2006.

[47] B. M. Kapron, D. Kempe, V. King, J. Saia, and V. Sanwalani. Fast asynchronous byzantine agreement and leader election with full information. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1038–1047, 2008.

[48] V. King and J. Saia. From almost everywhere to everywhere: Byzantine agreement with $\tilde{o}(n^{3/2})$ bits. In *Proceedings of the 23th International Symposium on Distributed Computing (DISC)*, pages 464–478, 2009.

[49] V. King and J. Saia. Breaking the $O(n^2)$ bit barrier: scalable byzantine agreement with an adaptive adversary. In *Proceedings of the 29th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 420–429, 2010.

[50] V. King, J. Saia, V. Sanwalani, and E. Vee. Scalable leader election. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 990–999, 2006.

[51] V. King, S. Lonargan, J. Saia, and A. Trehan. Load balanced scalable byzantine agreement through quorum building, with full information. In *Proceedings of the 12th International Conference on Distributed Computing and Networking (ICDCN)*, pages 203–214, 2011.

[52] M. V. N. A. Kumar, P. R. Goundan, K. Srinathan, and C. P. Rangan. On perfectly secure cmmunication over arbitrary networks. In *Proceedings of the 21th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 193–202, 2002.

[53] R. Kumaresan, S. Raghuraman, and A. Sealfon. Network oblivious transfer. In *Advances in Cryptology – CRYPTO 2016, part II*, pages 366–396, 2016.

[54] L. Lamport, R. E. Shostak, and M. C. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[55] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures, multisignatures, and verifiably encrypted signatures without random oracles. *Journal of Cryptology*, 26(2):340–373, 2013.

[56] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In *Advances in Cryptology – EUROCRYPT 2004*, pages 74–90, 2004.

[57] S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures: extended abstract. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS)*, pages 245–254, 2001.

[58] T. Moran, I. Orlov, and S. Richelson. Topology-hiding computation. In *Proceedings of the 12th Theory of Cryptography Conference, TCC 2015, part I*, pages 159–181, 2015.

[59] M. C. Pease, R. E. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.

[60] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 73–85, 1989.

[61] H. M. Sayeed and H. Abu-Amara. Efficient perfectly secure message transmission in synchronous networks. *Information and Control*, 126(1):53–61, 1996.

[62] T. Seito, T. Aikawa, J. Shikata, and T. Matsumoto. Information-theoretically secure key-insulated multireceiver authentication codes. In *Progress in Cryptology - AFRICACRYPT 2010*, pages 148–165, 2010.

[63] J. Shikata, G. Hanaoka, Y. Zheng, and H. Imai. Security notions for unconditionally secure signature schemes. In *Advances in Cryptology – EUROCRYPT 2002*, pages 434–449, 2002.

[64] G. Spini and G. Zémor. Perfectly secure message transmission in two rounds. In *Proceedings of the 14th Theory of Cryptography Conference, TCC 2016-B, part I*, pages 286–304, 2016.

[65] K. Srinathan, A. Narayanan, and C. P. Rangan. Optimal perfectly secure message transmission. In *Advances in Cryptology – CRYPTO 2004*, pages 545–561, 2004.

[66] E. Upfal. Tolerating linear number of faults in networks of bounded degree. In *Proceedings of the 11th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 83–89, 1992.