Must the Communication Graph of MPC Protocols be an Expander?*

Elette Boyle[†] Ran Cohen[‡‖] Deepesh Data[§‖] Pavel Hubáček[¶‖]

June 22, 2023

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 plain model (no setup) with 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.

*A preliminary version of this work appeared at *CRYPTO 2018* [BCDH18].

[†]Reichman University and NTT Research. E-mail: elette.boyle@runi.ac.il. Supported in part by ISF grant 1861/16, AFOSR Award FA9550-17-1-0069, and ERC Grant no. 307952.

[‡]Reichman University. E-mail: cohenran@runi.ac.il. Some of the work was done while the author was at MIT and Northeastern University and supported in part by Alfred P. Sloan Foundation Award 996698, ISF grant 1861/16, ERC starting grant 638121, NEU Cybersecurity and Privacy Institute, and NSF TWC-1664445.

[§]Meta Platforms, Inc. E-mail: deepesh.data@gmail.com.

[¶]Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic. E-mail: hubacek@iuuk.mff.cuni.cz. Supported by the project 17-09142S of GA ČR, Charles University project UNCE/SCI/004, and Charles University project PRIMUS/17/SCI/9. This work was done under financial support of the Neuron Fund for the support of science.

[‖]This work was done in part while visiting at the FACT Center at Reichman University (formerly IDC Herzliya).

Contents

1 Introduction **1**
- 1.1 Our Results . . . 2
- 1.2 Our Techniques . . . 4
- 1.3 Open Questions . . . 7
- 1.4 Additional Related Work . . . 7

2 Preliminaries **8**

3 Communication Graphs Induced by MPC Protocols **9**
- 3.1 Ensembles of Protocols and Functionalities . . . 10
- 3.2 The Communication Graph of a Protocol's Execution . . . 10
- 3.3 Locality of a Protocol . . . 12
- 3.4 Edge Expansion of a Protocol . . . 13
- 3.5 Discussion on the Definition of Expander Protocols . . . 15
- 3.6 The Adaptive Setting . . . 15

4 MPC with Non-Expanding Communication Graph **16**
- 4.1 Computational Security with Static Corruptions . . . 16
  - 4.1.1 Ideal Functionalities used in the Construction . . . 17
  - 4.1.2 Constructing Non-Expander Protocols . . . 19
- 4.2 Information-Theoretic Security . . . 21
- 4.3 Information-Theoretic MPC with Low Locality . . . 23
- 4.4 Adaptive Corruptions . . . 25

5 Expansion is Necessary for Correct Computation **28**
- 5.1 The Communication Model . . . 29
- 5.2 A Graph-Theoretic Theorem . . . 30
- 5.3 Proof of the Main Theorem (Theorem 5.2) . . . 31
  - 5.3.1 Defining Adversarial Strategies . . . 34
  - 5.3.2 Proving High Entropy of X_{I^*} . . . 39
  - 5.3.3 Proving the Common Output Contains X_{I^*} . . . 46

Bibliography **48**

A Preliminaries (Cont'd) **54**
- A.1 Cryptographic Primitives . . . 54
  - A.1.1 Error-Correcting Secret Sharing . . . 54
  - A.1.2 Committee Election . . . 54
  - A.1.3 Information-Theoretic Signatures . . . 55
  - A.1.4 Averaging Samplers . . . 56
- A.2 Model Definition . . . 57
- A.3 Correlated Randomness Functionalities . . . 60

B MPC with Non-Expanding Communication Graph (Cont'd) **61**
- B.1 Proof of Proposition 4.3 . . . 61
- B.2 Proof of Lemma 4.10 . . . 63

C Expansion is Necessary for Correct Computation (Cont'd) **67**
- C.1 Proof of the Graph-Theoretic Theorem (Theorem 5.6) . . . 68

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 [GMW87, BGW88, CCD88, RB89], 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 [PSL80, LSP82, DS83, FM97, GM93]).

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 Byzantine agreement deterministically against t corruptions, the graph must be $(t+1)$-connected [Dol82, FLM86].[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 [DPPU88, GO08, CGO15, HLP11, HIJ+16]. 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 [KSSV06, BGH13], and secure MPC protocols have been constructed whose communication graphs have degree $o(n)$—and as low as polylog(n) [DKM+17, BGT13, CCG+15, BCP15].[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 [BGT13].

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 [DKM+17, BGT13, CCG+15, BCP15]. 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 [KSSV06, CCG+15], and implicitly in others (e.g., using random graphs [KS09a], pseudorandom graphs [BGT13, BCG21], and averaging samplers [BGH13], 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, where the protocol proceeds in a round-by-round manner.

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 [BGT13]. 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 [CCG+15]) — but all with the common assumption of some form of input-independent *setup* information (such as a *public-key infrastructure*, PKI). 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 5). The setup is used to certify the identities of the members of both committees to the receiving parties, either via a public-key infrastructure for digital signatures (in the computational setting) or correlated randomness for information-theoretic signatures [SHZI02, SASM10] (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 [CCG+15], 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 [BGT13] 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 with common setup assumptions (but *no* private-coin setup as PKI). 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* [PSL80]), 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, even with access to a common reference string, 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 [KS10]. 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 proceed to discuss the technical aspects of the lower bound result. We refer the reader to Section 4.1.2 for an overview of the upper 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

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

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.

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 private-coin setup 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.

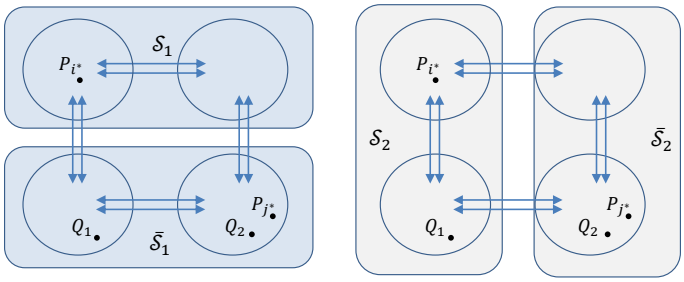

Figure 1: At the end of Phase I the communication graph is partitioned into 4 linear-size "islands" that are connected by sublinear many edges. On the left is the potential cut $(S_1, \bar{S}_1)$ and on the right the potential cut $(S_2, \bar{S}_2)$. If party Q_1 sends a message to party Q_2 then Q_2 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 [DDWY93, FY04, SA96, BF99, KGSR02, SNR04, BJLM06, BM05, ACdH06, FFGS07, KS09b, SZ16], Byzantine agreement [Dol82, DPPU88, BG93, Upf92], and secure multiparty computation [Bei07, BJLM11, BJLM06, CGO15, CGO10, CGO12].

In the setting of *topology-hiding* secure computation (THC) [MOR15, HMTZ16, AM17, ALM17, BBMM18], parties communicate over a partial graph, and the goal is to hide which pairs of honest parties are neighbors. 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). Intuitively, topology-hiding protocols can support non-expanding graphs since sublinear cuts should not be revealed during the protocol. A followup work [BBC+19]

explored this connection, and showed that the classical definition of THC is in fact too strong to hide sublinear cuts in the graph, and demonstrated how to capture this property by a weaker definition called distributional-topology-hiding computation. The separation between these security definitions is based on the distribution of graphs induced by the protocol in Section 3.

Another direction to study the connection between MPC and graph theory, explored in [HIK07, KRS16], is to consider MPC protocols that are based on 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 [KS10] 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 consider adaptive corruptions. However, we highlight three aspects in which the setting in [KS10] is weaker. First, [KS10] realize *Byzantine agreement* which is a weaker functionality than parallel broadcast; indeed, the standard techniques of reducing broadcast to Byzantine agreement requires the sender to first send its input to all other parties who then run BA — when every party acts as the sender this implies a complete communication graph. Second, [KS10] assume hidden channels where the adversary is unaware of honest-to-honest communication. And third, [KS10] 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 [GKKZ11], where atomic message delivery was used to overcome the lower bound of [HZ10]).

Abraham et al. [ACD+19] showed that when the adversary can remove messages of corrupted parties after the fact (i.e., without assuming atomic message delivery), adaptively secure BA requires every honest party to communicate with $\Omega(t)$ neighbors; i.e., every honest party has linear locality. We note that this does not imply our result since a linear degree does not rule out the existence of a sublinear cut.

Paper Organization

Basic notations are presented in Section 2. In Section 3, we provide our formalization of the communication graph induced by an 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. We defer general preliminaries and further details to the appendix.

2 Preliminaries

Notations. In the following we introduce some necessary notation and terminology. For $n, n_1, n_2 \in \mathbb{N}$, let $[n] = \{1, \cdots, n\}$ and $[n_1, n_2] = \{n_1, \cdots, n_2\}$. We denote by κ the security parameter. Let poly denote the set of all positive polynomials and let PPT denote a probabilistic algorithm that runs in *strictly* polynomial time. A function $\nu \colon \mathbb{N} \to \mathbb{R}$ is *negligible* if $\nu(\kappa) < 1/p(\kappa)$ for every $p \in \text{poly}$ and sufficiently large κ. Given a random variable X, we write $x \leftarrow X$ to indicate that x is selected according to X. The statistical distance between two random variables X and Y over a finite set $\mathcal{U}$, denoted $\text{SD}(X, Y)$, is defined as $\frac{1}{2} \cdot \sum_{u \in \mathcal{U}} |\Pr[X = u] - \Pr[Y = u]|$.

Two distribution ensembles $X = \{X(a, \kappa)\}_{a \in \{0,1\}^*, \kappa \in \mathbb{N}}$ and $Y = \{Y(a, \kappa)\}_{a \in \{0,1\}^*, \kappa \in \mathbb{N}}$ are computationally indistinguishable (denoted $X \stackrel{c}{\equiv} Y$) if for every non-uniform polynomial-time distinguisher

$\mathcal{A}$ there exists a function $\nu(\kappa) = \mathsf{negl}(\kappa)$, such that for every $a \in \{0,1\}^*$ and every κ,

$$|\Pr[\mathcal{A}(X(a,\kappa), 1^\kappa) = 1] - \Pr[\mathcal{A}(Y(a,\kappa), 1^\kappa) = 1]| \leq \nu(\kappa).$$

The distribution ensembles X and Y are statistically close (denoted $X \overset{s}{\equiv} Y$) if for every $a \in \{0,1\}^*$ and every κ it holds that $\mathrm{SD}(X(a,\kappa), Y(a,\kappa)) \leq \nu(\kappa)$.

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

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

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 $\{S, \bar{S}\}$. The weight of a cut $\{S, \bar{S}\}$ is defined to be $|\mathsf{edges}_G(S)|$. An α-cut is a cut $\{S, \bar{S}\}$ whose weight is smaller than α, i.e., 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 elaborate on the model in Appendix A.2, and refer the reader to [Can00, Gol04] 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 4.4) we consider *hidden* channels (as introduced in [CCG+15]), 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 [KSSV06, KKK+08, KS10, KLST11] and multiparty computation [BGT13, CCG+15, BCP15] 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 [PSL80, LSP82, DS83, GMW87, BGW88]. We note that for many large-scale protocols, e.g., protocols with low locality [KSSV06, KKK+08, KS10, KLST11, BGT13, BCP15], 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.

[5]Recall that in the synchronous model, every communication round is composed of a *send* phase and a *receive* phase, see Appendix A.2.

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 [KSSV06, KKK+08, KS10, KLST11, BGT13, CCG+15, BCP15]. 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. In Section 3.6, we discuss the *adaptive* setting.

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.

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

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

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

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

- 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 [BGT13], 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 messages to; that is,*

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

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 [Can00].

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 $q = q(n)$ 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 [HLW06, DW10] 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 (see discussion in Section 3.5).

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*

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

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 $h(G_n) \ge \epsilon$ 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 $f(n) \le h_{\pi,D_\pi,\mathcal{C}}(n)$, if for every n that is polynomially related to κ, for every $\boldsymbol{x} = (x_1, \ldots, x_n)$, every auxiliary information z, every $\mathcal{A} \in \mathcal{C}_n$ running with z, and every $\mathcal{I} \subseteq [n]$, it holds that*

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

  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 an upper bound of the edge expansion of π facing adversaries in $\mathcal{C}$, denoted $f(n) \ge h_{\pi,D_\pi,\mathcal{C}}(n)$, 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 auxiliary information z, every $\mathcal{A} \in \mathcal{C}_n$ running with z, and every $\mathcal{I} \subseteq [n]$, it holds that*

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

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 [DS83, FM97, GM93, KSSV06, KKK+08, KLST11, KS10] and in the realm of MPC [GMW87, BGW88, BGT13, CCG+15, BCP15]. 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 not an expander 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 $\alpha(n) \in o(n)$. □

3.5 Discussion on the Definition of Expander Protocols

In addition to edge expansion, there are two other commonly studied notions of expansion in graphs: *spectral expansion* and *vertex expansion* (see [HLW06] for a survey on expander graphs). Spectral expansion is a linear-algebraic definition of expansion in regular graphs, equal to the difference of first and second largest eigenvalues of the graph's adjacency matrix. Vertex (like edge) expansion is combinatorial definitions of expansion, but considers the number of distinct nodes neighboring subsets of the graph as opposed to the number of outgoing edges.

The three metrics are loosely coupled, and analyzing any of them for the communication graphs in our setting seems a natural choice. We choose to study edge expansion, as it is particularly amenable to our new techniques. We further consider an *unscaled* version of edge expansion, as opposed to the more stringent scaled version wherein the expansion ratio $h(G)$ of Equation (1) is defined with an additional scale factor of $(\deg G)^{-1}$. The unscaled variant (while weaker) satisfies the more natural property that it is *monotonic* with respect to the number of edges. That is, by adding more edges to the graph, the value of $h(G)$ cannot decrease; whereas, when dividing $h(G)$ by the degree, adding more edges to the graph might actually decrease the value of $h(G)$, and the graph which was an expander earlier may end up being non-expander. This monotonicity appears more natural in the setting of communication graphs, where adding more edges cannot harm the ability to successfully execute a protocol.

We remark that while our definitional focus is on this form of unscaled edge expansion, our upper bounds (i.e., protocols with non-expanding communication graphs) apply to all aforementioned notions. Considering and extending our lower bound to alternative notions of expansion (spectral/vertex/scaled) is left as an interesting open problem.

3.6 The Adaptive Setting

In the adaptive setting, the definitions of locality of protocols and of protocols with communication graph that forms an expander follow the same spirit as in the static case, however, require a few technical modifications.

Recall that we follow the adaptive model from Canetti [Can00],[6] where an environment machine interacts with the adversary/simulator. In particular, the adversary does not receive auxiliary information at the onset of the protocol; rather the environment acts as an "interactive auxiliary-information provider" and hands the adversary auxiliary information about parties that get corrupted dynamically. In addition, the set of corrupted parties is not defined at the beginning, but generated dynamically during the protocol based on corruption request issued by the adversary, and also after the completion of the protocol, during the post-execution corruption (PEC) phase, based on corruption requests issued by the environment.

Therefore, the required changes to the definitions are two-fold:

1. The parameters for defining an instance of a protocol execution are: the n-party protocol π_n the security parameter κ, the input vector for the parties $\boldsymbol{x} = (x_1, \ldots, x_n)$, the correlated randomness for the parties $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_n)$, the environment $\mathcal{Z}$, the adversary $\mathcal{A}$, the auxiliary input for the environment z, and the random coins for the parties, the adversary, and the environment $\boldsymbol{r} = (r_1, \ldots, r_n, r_{\mathcal{A}}, r_{\mathcal{Z}})$.

[6] We follow the modular composition framework [Can00] for the sake of clarity and simplicity. We note that the definitions can be adjusted to the UC framework [Can01].

We denote by $\mathsf{instance}_{\mathsf{adaptive}}(\pi_n) = (\pi_n, \mathcal{Z}, \mathcal{A}, \kappa, \boldsymbol{x}, \boldsymbol{\rho}, z, \boldsymbol{r})$.

2. The second difference considers the timing where the communication graph is set.

   - After the parties generate their output, and before the PEC phase begins.
   - At the end of the PEC phase, when the environment outputs its decision bit.

   Since the communication graph is fixed at the end of the protocol (before the PEC phase begins), the difference lies in the identity of the corrupted parties. More precisely, an edge that appears in the graph before the PEC phase might not appear after the PEC phase, in case both parties became corrupt. For this reason, we consider the communication graph after the parties generate their outputs and before the PEC phase begins.

All definitions as presented for static case translate to the adaptive setting with the two adjustments presented above.

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 in Section 4.4 to the adaptive-corruption setting.

4.1 Computational Security with Static Corruptions

We start by considering the computational setting with static corruptions.

Theorem 4.1 (restating Theorem 1.1 and Item 1 of Theorem 1.2)**.** *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[7] 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.*[8]

[7] In the SKI model every pair of parties has a secret random string that is unknown to other parties.
[8] 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. [BMR90] or of Damgård and Ishai [DI05].[9]
- The second part follows using the low-locality MPC protocol of Boyle et al. [BGT13] that exists assuming one-way functions in the PKI model with secure channels and tolerates $t = (1/3 - \delta)n$ static corruptions.[10]
- The third part follows using the low-locality MPC protocol of Chandran et al. [CCG+15] 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 of 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 $\mathsf{Verify}_{\mathtt{vk}_1,\ldots,\mathtt{vk}_m}(\mu, \sigma)$ the verification algorithm that for every $i \in [m]$ computes $b_i \leftarrow \mathsf{Verify}_{\mathtt{vk}_i}(\mu, \sigma_i)$, 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 [MOR01, BGLS03, LMRS04, LOS+13] in order to obtain better communication complexity, however, we use the notation above both for simplicity and as a step toward 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 (see Definition A.1). 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 formal description of the functionality can be found in Figure 2.

[9] Generic honest-majority MPC protocols require a broadcast channel or some form of trusted setup assumptions [CL17, CHOR18]. The PKI assumption in Theorem 4.1 is sufficient in the computational setting. Looking ahead, in the information-theoretic setting (Section 4.2) additional adjustments are required.

[10] In [BGT13] public-key encryption is also assumed, but as we show in Section 4.3, this assumption can be removed.

The functionality $f^{(t',n')}_{\mathsf{elect\text{-}share}}$

The m-party functionality $f^{(t',n')}_{\mathsf{elect\text{-}share}}$ is parametrized by a signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ and a (t', n') ECSS scheme $(\mathsf{Share}, \mathsf{Recon})$, and proceeds with parties $\mathcal{P}_1 = \{P_1, \ldots, P_m\}$ as follows.

1. Every party P_i sends a pair of values $(x_i, \mathtt{sk}_i)$ as its input, where x_i is the actual input value and $\mathtt{sk}_i$ is a signing key. (If P_i didn't send a valid input, set it to the default value, e.g., zero.)
2. Sample uniformly at random two subsets (committees) $\mathcal{C}_1, \mathcal{C}_2 \subseteq [m]$ of size n'.
3. Sign each subset as $\sigma_1 = \mathsf{Sign}_{\mathtt{sk}_1,\ldots,\mathtt{sk}_m}(\mathcal{C}_1)$ and $\sigma_2 = \mathsf{Sign}_{\mathtt{sk}_1,\ldots,\mathtt{sk}_m}(\mathcal{C}_2)$.
4. For every $i \in [m]$, secret share x_i as $(s_i^1, \ldots, s_i^{n'}) \leftarrow \mathsf{Share}(x_i)$.
5. For every $j \in [n']$, set $\boldsymbol{s}_j = (s_1^j, \ldots, s_m^j)$.
6. Denote $\mathcal{C}_1 = \{i_{(1,1)}, \ldots, i_{(1,n')}\}$. For every $i \notin \mathcal{C}_1$ set the output of P_i to be $\mathcal{C}_1$. For every $i = i_{(1,j)} \in \mathcal{C}_1$ (for some $j \in [n']$), set the output of P_i to be $(\mathcal{C}_1, \sigma_1, \mathcal{C}_2, \sigma_2, \boldsymbol{s}_j)$.

Figure 2: The Elect-and-Share functionality

The Reconstruct-and-Compute functionality. The Reconstruct-and-Compute functionality, $f^{(\mathtt{vk}_1,\ldots,\mathtt{vk}_m)}_{\mathsf{recon\text{-}compute}}$, is an m-party functionality. Denote the party-set by $\{P_{m+1}, \ldots, P_{2m}\}$.[11] 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 formal description of the functionality can be found in Figure 4.

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. The formal description of the functionality can be found in Figure 3.

The functionality $f^{\mathcal{C}_1}_{\mathsf{out\text{-}dist}}$

The m-party functionality $f^{\mathcal{C}_1}_{\mathsf{out\text{-}dist}}$ is parametrized by a subset $\mathcal{C}_1 \subseteq [m]$, and proceeds with parties $\mathcal{P}_1 = \{P_1, \ldots, P_m\}$ as follows.

1. Every party P_i, with $i \in \mathcal{C}_1$, gives input value y_i, a party P_i with $i \notin \mathcal{C}_1$ gives the empty input ϵ.
2. Denote $y = \mathsf{majority}\{y_i \mid i \in \mathcal{C}_1\}$ (choose arbitrarily if the majority is not unique).
3. Output y to every $P_i \in \mathcal{P}_1$.

Figure 3: The Output-Distribution functionality

[11] We use the notation $\{P_{m+1}, \ldots, P_{2m}\}$ instead of the more standard $\{P_1, \ldots, P_m\}$ for consistency with the protocol π^{ne}_n for $n = 2m$ (described in Figure 6), where parties $\{P_{m+1}, \ldots, P_{2m}\}$ invoke the functionality.

The functionality $f_{\text{recon-compute}}$

The m-party functionality $f_{\text{recon-compute}}^{(\mathtt{vk}_1,\ldots,\mathtt{vk}_m)}$ is parametrized by a signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, a (t', n') ECSS scheme $(\mathsf{Share}, \mathsf{Recon})$, and a vector of verification keys $(\mathtt{vk}_1, \ldots, \mathtt{vk}_m)$, and proceeds with parties $\mathcal{P}_2 = \{P_{m+1}, \ldots, P_{2m}\}$ as follows.

1. Every party P_{m+i} sends a pair of values (x_{m+i}, z_{m+i}) as its input, where x_{m+i} is the actual input value and either $z_{m+i} = \epsilon$ or $z_{m+i} = (\mathcal{C}_{m+i}, \sigma_{m+i}, \boldsymbol{s}_{m+i})$.
2. For every P_{m+i} that provided $z_{m+i} \neq \epsilon$, verify that $\mathsf{Verify}_{\mathtt{vk}_1,\ldots,\mathtt{vk}_m}(\mathcal{C}_{m+i}, \sigma_{m+i}) = 1$ (ignore invalid inputs). If there is no subset $\mathcal{C}_2 \subseteq [m]$ of size n' with an accepting signature, or if there exists more than one such subset, then abort. Otherwise, denote $\mathcal{C}_2 = \{i_{(2,1)} \ldots, i_{(2,n')}\}$.
3. For every $i = i_{(2,j)} \in \mathcal{C}_2$, let $\boldsymbol{s}_{m+i_{(2,j)}} = (s_1^j, \ldots, s_m^j)$ be the input provided by P_{m+i}. (If P_{m+i} provided invalid input, set $\boldsymbol{s}_{m+i_{(2,j)}}$ to be the default value, e.g., the zero vector.)
4. For every $i \in [m]$, reconstruct $x_i = \mathsf{Recon}(s_i^1, \ldots, s_i^{n'})$.
5. Compute $y = f(x_1 \ldots, x_m, x_{m+1}, \ldots, x_{2m})$.
6. Output y to every $P_{m+i} \in \mathcal{P}_2$.

Figure 4: The Reconstruct-and-Compute functionality

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 6). 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 5). The left subset will invoke the Elect-and-Share functionality, that elects two subsets $\mathcal{C}_1, \mathcal{C}_2 \subseteq [m]$ of size $n' = \log^2(n)$.[12] 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, $\mathcal{C}_2$ will transfer the output value to $\mathcal{C}_1$, and the left subset will invoke the Output-Distribution functionality in order to distribute the output value to all the parties.

Lemma 4.2. *Let $f = \{f_n\}_{n\in 2\mathbb{N}}$,[13] 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 $(f_{\text{elect-share}}, f_{\text{recon-compute}}, f_{\text{out-dist}})$ 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.*

[12] We note that any $n' \in \omega(\log n)$ will do.

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

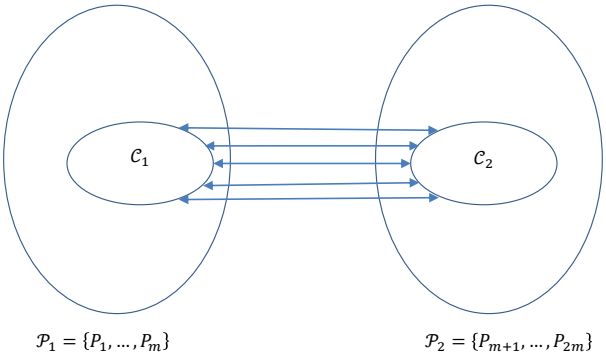

Figure 5: The non-expanding subsets in the protocol π^{ne}. The sets $\mathcal{C}_1$ and $\mathcal{C}_2$ are of poly-logarithmic size and the sets $\mathcal{P}_1$ and $\mathcal{P}_2$ are of linear size. The number of edges between $\mathcal{P}_1$ and $\mathcal{P}_2$ is poly-logarithmic.

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 $\ell = 2 \cdot \ell_\rho + \log^2(n)$.*

Proof. For $m \in \mathbb{N}$ and $n = 2m$, we construct the n-party protocol π^{ne}_n (see Figure 6) 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^{ne}_n 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^{ne}_n 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 Appendix B.1.

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 $\mathsf{edges}(\mathcal{P}_1, \mathcal{P}_2) = |\mathcal{C}_1| \times |\mathcal{C}_2| = \log^2(n) \cdot \log^2(n)$. 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} (see Definition 3.10). We conclude that the communication graph of π^{ne} is strongly not an expander. □

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 an 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 6: Non-expanding MPC in the $(f_{\mathsf{elect\text{-}share}}, f_{\mathsf{recon\text{-}compute}}, f_{\mathsf{out\text{-}dist}})$-hybrid model

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 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 (see Appendix A.1.3).

Theorem 4.6 (restating Item 2 of Theorem 1.2)**.** *Let $f = \{f_n\}_{n\in\mathbb{N}}$ be an ensemble of functionalities and let $\delta > 0$. 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 Lemma 4.7 (below), which is an information-theoretic variant of Lemma 4.2, 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 [RB89].

- In Section 4.3 we prove information-theoretic variant of the low-locality MPC protocol of Boyle et al. [BGT13] 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. □

The main differences between standard digital signatures and information-theoretic signatures are: (1) the verification key is not publicly known, but rather, must be kept hidden (meaning that each party P_i has a different verification key $\mathtt{vk}_i^j$ with respect to every party P_j), and (2) a bound on the number times that a secret signing key and a secret verification key are used must be a priori known. The latter does not form a problem since, indeed, the number of signatures that are generated in Protocol π_n^{ne} (Figure 6) by any of the signing keys is 2, and likewise, each verification key is used to verify 2 signatures. However, the former requires adjusting the functionality $f_{\mathsf{recon\text{-}compute}}$.

Instead of having the functionality be parametrized by the vector of verification keys $\mathtt{vk}_1, \ldots, \mathtt{vk}_m$ (which, as mentioned above, will not be secure in the information-theoretic setting), each party P_{m+i}, with $i \in [m]$, has a vector of (secret) verification keys $\mathtt{vk}_{m+i}^1, \ldots, \mathtt{vk}_{m+i}^m$ corresponding to the parties in $\mathcal{P}_1$.

In the adjusted functionality, denoted $f_{\mathsf{it\text{-}recon\text{-}compute}}$, we change the functionality $f_{\mathsf{recon\text{-}compute}}$ by having each party P_{m+i} provide an additional input consisting of its verification keys $\mathtt{vk}_{m+i}^1, \ldots, \mathtt{vk}_{m+i}^m$ (and the functionality is no longer parametrized by any value). Now, on each input consisting of a subset $\mathcal{C}_{m+i}$ and corresponding signature-vector σ_{m+i}, the functionality verifies the j'th signature of the set $\mathcal{C}_{m+i}$ using the verification keys $\mathtt{vk}_{m+1}^j, \ldots, \mathtt{vk}_{2m}^j$. If at most t verifications fail, the functionality accepts the committee, whereas if more than t verifications fail, the functionality ignores the subset. Next, the functionality proceeds as in $f_{\mathsf{recon\text{-}compute}}$.

Signature notations. Given an information-theoretically (ℓ_S, ℓ_V)-secure signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, and m tuples of signing and verification keys $(\mathtt{sk}_i, \vec{\mathtt{vk}}_i) \leftarrow \mathsf{Gen}(1^\kappa, n, \ell_S)$, where for every $i \in [m]$ the verification keys are $\vec{\mathtt{vk}}_i = (\mathtt{vk}_1^i, \ldots, \mathtt{vk}_n^i)$, we use the following notations for signing and verifying with multiple keys:

- Given a message μ we denote by $\mathsf{Sign}_{\mathtt{sk}_1,\ldots,\mathtt{sk}_m}(\mu)$ we consider 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 $\mathsf{Verify}_{\vec{\mathtt{vk}}_{m+1},\ldots,\vec{\mathtt{vk}}_{2m}}(\mu, \sigma)$ the following verification algorithm:

  1. For every $i \in [m]$ proceeds as follows:
     (a) For every $j \in [m]$ let $b_i^j \leftarrow \mathsf{Verify}_{\mathtt{vk}_{m+j}^i}(\mu, \sigma_i)$.
     (b) Set $b_i = 1$ if and only if $\sum_{j=1}^m b_i^j \geq m - t$, i.e., even if up to t verification keys reject the signature.
  2. Accepts σ if and only if $\sum_{i=1}^m b_i \geq m - t$, i.e., even if up to t signatures are invalid.

Lemma 4.7. *Let $f = \{f_n\}_{n\in 2\mathbb{N}}$, where f_n is an n-party functionality for $n = 2m$, and let $\delta > 0$. Then, in the $(2,2)$-IT-PKI-hybrid model with secure channels, where a trusted party additionally computes the m-party functionality-ensembles $(f_{\mathsf{elect\text{-}share}}, f_{\mathsf{it\text{-}recon\text{-}compute}}, f_{\mathsf{out\text{-}dist}})$ tolerating $\gamma \cdot m$ corruptions, there exists a protocol ensemble π that securely computes f tolerating static 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 $f_{\mathsf{elect\text{-}share}}, f_{\mathsf{it\text{-}recon\text{-}compute}}, f_{\mathsf{out\text{-}dist}}$ (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 $\ell = 2 \cdot \ell_\rho + \log^2(n)$.*

Proof sketch. The proof of the lemma follows in similar lines as the proof of Lemma 4.2. We highlight the main differences.

- **IT-PKI model.** The protocol is defined in the $(2,2)$-IT-PKI-hybrid model, rather than the PKI-hybrid model, meaning that each party receives a secret signing key along with a secret verification key for every other party. At most two values can signed verified by these keys.
- **Hybrid model.** The protocol is defined in the $f_{\mathsf{it\text{-}recon\text{-}compute}}$-hybrid model, rather than the $f_{\mathsf{recon\text{-}compute}}$-hybrid model, meaning that each party in $\mathcal{P}_2$ sends its vector of secret verification keys as input to the functionality.
- **The simulation.** The simulator generates appropriate signing and verification keys for simulating the $(2,2)$-IT-PKI functionality, and receives the verification keys from the adversaries when emulating the functionality $f_{\mathsf{it\text{-}recon\text{-}compute}}$. No other changes are needed.
- **The hybrid games.** The first hybrid game and the second are statistically close when using $(2,2)$-IT-PKI, rather than computationally indistinguishable (see Claim B.1).

The rest of the proof follows the proof of Lemma 4.2. □

4.3 Information-Theoretic MPC with Low Locality

The protocol of Boyle, Goldwasser, and Tessaro [BGT13] follows a framework common to other protocols achieving low locality (see [KS09a, KLST11, BGH13] and the references therein). First, the parties compute almost-everywhere agreement, that is agreement among at least a $1 - o(1)$ fraction of parties. Next, the parties upgrade to full agreement via a transformation that preserves

low locality. The results in [BGT13] are in the computational setting where the main cryptographic tools that are being used are public-key encryption, digital signatures, and pseudorandom functions (PRF). In this section, we show that the approach of [BGT13] can be adapted to the information-theoretic setting by removing the need of public-key encryption and by substituting other computational primitives by their information-theoretic analogues. Namely, we will use information-theoretic signatures instead of digital signatures and samplers [Zuc97, Gol11] instead of PRF.

Overview of the BGT protocol. The protocol consists of two parts:

1. **Establishing a polylog-degree communication tree.** This part of the protocol requires digital signatures established via a PKI and a PRF.

   - Initially, the parties run the protocol of King et al. [KSSV06] to reach *almost everywhere* agreement on a random seed while establishing a polylog-degree communication tree, and maintaining polylog locality. This part holds information theoretically.
   - Next, *certified* almost everywhere agreement is obtained by having the parties sign the seed and distribute the signatures. Specifically, every party sends its signature on the seed up the tree to the supreme committee, which concatenates all signatures to form a certificate on the seed, and sends it down the tree to (almost all) the parties.
   - Finally, to achieve *full* agreement, every party that received sufficiently many signatures on the seed locally evaluates the PRF on the seed and its identity to get a polylog subset of parties, and sends the certified seed to each party in this set. A party that receives the certified seed can validate the seed is properly signed and that he is a valid recipient of the message.

   Note that the PRF is used for its combinatorics properties and is not needed for security.

2. **Computing the function.** Having established the communication tree, the supreme committee (i.e., the parties assigned to the root) jointly generate keys to a threshold encryption scheme such that each committee member holds a share of the decryption key and the public key is known. Next, they distribute the public encryption key down the tree. Every party encrypts its input using the encryption key and sends it up the tree. Finally, the supreme committee runs a protocol to decrypt the ciphertexts and evaluate the function to obtain the output, which is distributed to all parties.

We now turn to explain how to construct an information-theoretic analogue for this protocol.

Establishing the communication tree information theoretically. This part follows almost immediately from [BGT13]. The digital signatures and the PKI are replaced by information-theoretic signatures and an IT-PKI, where every party signs the κ-bit seed (i.e., one signature operation per party) and has to verify n signatures. As mentioned above, the PRF is used only for its combinatorial properties (mapping each party to a polylog set of neighbors) and not for other security purposes, and so it can be replaced by a sampler with good parameters (this approach was adopted by [BGH13] to construct the first BA protocol for with polylog communication complexity). We provide more information on the samplers that are employed in Appendix A.1.4.

Computing the function information theoretically. Once the communication tree is established, each party must send his input to the supreme-committee members in a way that allows them to compute the function. We replace the public-key encryption used in [BGT13] by secret sharing. To understand this step, we will first explain the structure of the communication tree.

For any $n \in \mathbb{N}$, the communication tree from [KSSV06] is a graph $G = G(n)$ in which every node is labeled by subsets of $[n]$ that satisfies the following properties:

- G is a tree of height $l^* \in O(\log n / \log\log n)$. Each node from level $\ell > 0$ has $\log n$ nodes from level $\ell - 1$ as its children.
- Each node of G is labeled by a subset polylog(n) parties.
- Each party is assigned to polylog(n) nodes at each level.

King et al. [KSSV06] showed that for any $t = \beta n$ static corruptions, all but a $3/\log n$ fraction of the leaf nodes have a good path up to the root node (i.e., a path on which each committee contains a majority of honest parties). As observed in [BGT13], this implies that for a $1 - o(1)$ fraction of the parties, majority of the leaf nodes that they are assigned to are good. In addition, each leaf node is connected to polylog(n) parties as determined by the sampler.

The high-level idea now is to let each party P_i, with associated leaf-nodes $v_1, \ldots, v_k$, secret share its input as $(s_{i,1}, \ldots, s_{i,k}) \leftarrow \mathsf{Share}(x_i)$ and send the share $s_{i,j}$ to v_j. Each leaf-node will send the received shares up the tree until to supreme committee (the parties associated with the root) receive all shares, reconstruct all inputs, and compute the function. Clearly, this idea does not provide any privacy, since the adversary may have corrupted parties in many leaves, thus recover the honest parties' inputs values. To overcome this problem, instead of sending $s_{i,j}$ in the clear, each P_i will secret share each share as $(s^1_{i,j}, \ldots, s^m_{i,j}) \leftarrow \mathsf{Share}(s_{i,j})$, where m is the size of a committee associated to a leaf node, and send $s^h_{i,j}$ to the h'th party in v_j. Stated differently, each leaf-node will hold the shares in a secret-shared form.

The next part of the protocol proceeds recursively. For every node v in level ℓ and every child node u of v in level $\ell - 1$, the parties associated with u and with v will run a secure protocol for the following functionality: For each of the shared values held by the parties associated with u, they enter the secret shares as input; the functionality reconstructs the value, reshares it, and outputs the new shares to the parties associated with v.

In order to implement this functionality using BGW, we require that the union of the parties associated with u and with v will have a $2/3$ majority. Such a majority is guaranteed with overwhelming probability if the total fraction of corruptions is $1/6 - \epsilon$, for an arbitrary small constant ϵ. We thus proved the following theorem.

Theorem 4.8 (restating Theorem 1.3)**.** *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.*

4.4 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 obstacle 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 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 is inherent when the communication lines are visible to the adversary; therefore, we turn to the hidden-channels setting that was used by [CCG+15], where the adversary does not learn whether a message is sent between two honest parties (see Appendix A.2).

Theorem 4.9 (restating Items 3 and 4 of Theorem 1.2)**.** *Let $f = \{f_n\}_{n\in\mathbb{N}}$ be an ensemble of functionalities, let $\delta > 0$, let $\beta < 1/8 - \delta$, and let $t = \beta \cdot n$. 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. *Assuming the existence of trapdoor permutations with a reverse domain sampler, f can be securely computed by a protocol ensemble π in the PKI and SKI 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.*[14]
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 Lemma 4.10 (below), which is an adaptively secure variant of Lemma 4.2, by instantiating the hybrid functionalities using 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. [CDD+99] or Damgård and Ishai [DI05].
- The second part follows using the adaptively secure protocol of Chandran et al. [CCG+15].
- The third part follows using information-theoretic signatures via the same adjustments that were employed in Section 4.2, and using the protocol of Cramer et al. [CDD+99]. □

Hiding the subsets $\mathcal{C}_1$ and $\mathcal{C}_2$ from the parties requires adjusting the ideal functionalities that are used in Section 4.1. We now describe the adjusted functionalities.

The Adaptive-Elect-and-Share functionality. The Adaptive-Elect-and-Share m-party functionality, $f^{(t',n')}_{\text{a-elect-share}}$, is defined in a similar way as the Elect-and-Share functionality (Figure 2) with the following difference. Instead of outputting the set $\mathcal{C}_1$ to all parties and the set $\mathcal{C}_2$ to parties in $\mathcal{C}_1$, the functionality outputs for every party in $\mathcal{C}_1$ an index of a single party in $\mathcal{C}_2$ (and signs the values). Parties outside of $\mathcal{C}_1$ receive no output. The formal description of the functionality can be found in Figure 7.

[14] We note that the adaptively secure protocols in [CCG+15] are proven in a model with atomic simultaneous multi-send operations [HZ10, GKKZ11] and secure erasures.

The functionality $f^{(t',n')}_{\mathsf{a\text{-}elect\text{-}share}}$

The m-party functionality $f^{(t',n')}_{\mathsf{a\text{-}elect\text{-}share}}$ is parametrized by a signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ and a (t', n') ECSS scheme $(\mathsf{Share}, \mathsf{Recon})$, and proceeds with parties $\mathcal{P}_1 = \{P_1, \ldots, P_m\}$ as follows.

1. Every party P_i sends a pair of values $(x_i, \mathtt{sk}_i)$ as its input, where x_i is the actual input value and $\mathtt{sk}_i$ is a signing key. (If P_i didn't send a valid input, set it to the default value, e.g., zero.)
2. Sample uniformly at random two subsets (committees) $\mathcal{C}_1, \mathcal{C}_2 \subseteq [m]$ of size n'. Denote $\mathcal{C}_1 = \{i_{(1,1)}, \ldots, i_{(1,n')}\}$ and $\mathcal{C}_2 = \{i_{(2,1)}, \ldots, i_{(2,n')}\}$.
3. For every $j \in [n']$, sign $\sigma_{1,j} = \mathsf{Sign}_{\mathtt{sk}_1,\ldots,\mathtt{sk}_m}(i_{(1,j)})$ and $\sigma_{2,j} = \mathsf{Sign}_{\mathtt{sk}_1,\ldots,\mathtt{sk}_m}(m + i_{(2,j)})$.
4. For every $i \in [m]$, secret share x_i as $(s_i^1, \ldots, s_i^{n'}) \leftarrow \mathsf{Share}(x_i)$.
5. For every $j \in [n']$, set $\boldsymbol{s}_j = (s_1^j, \ldots, s_m^j)$.
6. For every $i = i_{(1,j)} \in \mathcal{C}_1$ (for some $j \in [n']$), set the output of P_i to be $(i_{2,j}, \sigma_{1,j}, \sigma_{2,j}, \boldsymbol{s}_j)$. (Parties outside of $\mathcal{C}_1$ receive the empty output ϵ)

Figure 7: The Adaptive-Elect-and-Share functionality

The Adaptive-Reconstruct-and-Compute functionality. The Adaptive-Reconstruct-and-Compute functionality, $f^{(\mathtt{vk}_1,\ldots,\mathtt{vk}_m)}_{\mathsf{a\text{-}recon\text{-}compute}}$, is defined in a similar way as the Reconstruct-and-Compute functionality (Figure 4) with the following difference. Instead of having the potential additional input value consist of a signed subset $\mathcal{C}_2 \subseteq [m]$, it consists of a signed index. The functionality verifies that if a party provided an additional input, then it has a valid signature of its own index, and derives the committee $\mathcal{C}_2$ from the indices with a valid signature. The formal description of the functionality can be found in Figure 8.

The functionality $f_{\mathsf{a\text{-}recon\text{-}compute}}$

The m-party functionality $f^{(\mathtt{vk}_1,\ldots,\mathtt{vk}_m)}_{\mathsf{a\text{-}recon\text{-}compute}}$ is parametrized by a signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, a (t', n') ECSS scheme $(\mathsf{Share}, \mathsf{Recon})$, and a vector of verification keys $(\mathtt{vk}_1, \ldots, \mathtt{vk}_m)$, and proceeds with parties $\mathcal{P}_2 = \{P_{m+1}, \ldots, P_{2m}\}$ as follows.

1. Every party P_{m+i} sends a pair of values (x_{m+i}, z_{m+i}) as its input, where x_{m+i} is the actual input value and either $z_{m+i} = \epsilon$ or $z_{m+i} = (\sigma_{m+i}, \boldsymbol{s}_{m+i})$.
2. For every P_{m+i} with $z_{m+i} \neq \epsilon$, check if $\mathsf{Verify}_{\mathtt{vk}_1,\ldots,\mathtt{vk}_m}(m + i, \sigma_{m+i}) = 1$ (ignore invalid inputs). If exactly n' indices are properly signed, denote as $\mathcal{C}_2 = \{i_{(2,1)} \ldots, i_{(2,n')}\}$; otherwise, abort.
3. For every $i = i_{(2,j)} \in \mathcal{C}_2$, let $\boldsymbol{s}_{m+i_{(2,j)}} = (s_1^j, \ldots, s_m^j)$ be the input provided by P_{m+i}. (If P_{m+i} provided invalid input, set $\boldsymbol{s}_{m+i_{(2,j)}}$ to be the default value, e.g., the zero vector.)
4. For every $i \in [m]$, reconstruct $x_i = \mathsf{Recon}(s_i^1, \ldots, s_i^{n'})$.
5. Compute $y = f(x_1 \ldots, x_m, x_{m+1}, \ldots, x_{2m})$.
6. Output y to every $P_{m+i} \in \mathcal{P}_2$.

Figure 8: The Adaptive-Reconstruct-and-Compute functionality

The Adaptive-Output-Distribution functionality. The Adaptive-Output-Distribution functionality is defined in a similar way as the Output-Distribution functionality (Figure 3) with the following difference. Instead of being parametrized by a subset $\mathcal{C}_1$ that specifies the input providers, the functionality is parametrized by the verification keys $\mathtt{vk}_1, \ldots, \mathtt{vk}_m$, every party that provides an input value must also provide a signature of its own index. The formal description of the functionality can be found in Figure 9.

The functionality $f_{\mathsf{a\text{-}out\text{-}dist}}$

The m-party functionality $f_{\mathsf{a\text{-}out\text{-}dist}}^{(\mathtt{vk}_1,\ldots,\mathtt{vk}_m)}$ is parametrized by a signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ and a vector of verification keys $(\mathtt{vk}_1, \ldots, \mathtt{vk}_m)$, and proceeds with parties $\mathcal{P}_1 = \{P_1, \ldots, P_m\}$ as follows.

1. Every party P_i gives either the empty input ϵ of an input value of the form (σ_i, y_i).
2. For every party P_i that provided a non-empty input, verify that $\mathsf{Verify}_{\mathtt{vk}_1,\ldots,\mathtt{vk}_m}(i, \sigma_i) = 1$ (ignore invalid inputs). Denote by $\mathcal{C}_1$ the set of indices of parties that gave valid inputs.
3. Denote $y = \mathsf{majority}\{y_i \mid i \in \mathcal{C}_1\}$ (choose arbitrarily if the majority is not unique).
4. Output y to every $P_i \in \mathcal{P}_1$.

Figure 9: The Adaptive-Output-Distribution functionality

Lemma 4.10. *Let $f = \{f_n\}_{n\in 2\mathbb{N}}$, where f_n is an n-party functionality for $n = 2m$, and let $\delta > 0$. Then, in the PKI-hybrid model with secure channels, where a trusted party additionally computes the m-party functionality-ensembles $(f_{\mathsf{a\text{-}elect\text{-}share}}, f_{\mathsf{a\text{-}recon\text{-}compute}}, f_{\mathsf{a\text{-}out\text{-}dist}})$ tolerating $\gamma \cdot m$ corruptions, there exists a protocol ensemble π that securely computes f, with computational security, tolerating adaptive PPT βn=adversaries, for $\beta < \min(1/8 - \delta, \gamma/2)$ with the following guarantees:*

1. *The communication graph induced by π is strongly not an expander.*
2. *Denote by f_1, f_2, f_3 the functionalities $f_{\mathsf{a\text{-}elect\text{-}share}}, f_{\mathsf{a\text{-}recon\text{-}compute}}, f_{\mathsf{a\text{-}out\text{-}dist}}$ (resp.). If protocols 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 $\ell = 2 \cdot \ell_\rho + 1$.*

The proof of Lemma 4.10 can be found in Appendix B.2.

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,[15] a PPT adaptive adversary, and focus on parallel multi-valued broadcast (also known as interactive consistency [PSL80]), 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.

[15] The lower bound immediately extends to a setting where the parties have access to a common reference string; we consider the plain setting for simplicity.

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 4.4, 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 (restating Theorem 1.4)**.** *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 the communication graph of 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 toward 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.

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 4.4, 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 [Can00] 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 [Can01]. As we consider *synchronous* protocols, we use the parallel SMT functionality that was formalized in [CCGZ19, CCGZ21].[16]

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^i_j to be the empty string ϵ (the value x^i_j represents the message to be sent from P_i to P_j).*

- ***The input phase.*** *Every party P_i sends a vector of n messages $(v^i_1, \ldots, v^i_n)$. The functionality sets $x^i_j = v^i_j$, 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*

$$l_{\mathsf{psmt}}\left((x^1_1, \ldots, x^1_n), \ldots, (x^n_1, \ldots, x^n_n)\right) = \left((y^1_1, \ldots, y^1_n), \ldots, (y^n_1, \ldots, y^n_n)\right),$$

  where $y^i_j = |x^i_j|$ in case P_j is honest and $y^i_j = x^i_j$ 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 messages are not delivered yet to the honest parties).

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

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 is deferred to Appendix C.1.

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 $\bar{S} = \bigcup_{j \in [\ell] \setminus J} U_j$.*

[16] 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.

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 an $(\alpha(n), n/c)$-partition of G_n, denoted Γ; it holds that $|\Gamma| \leq c$.*
2. *An $(\alpha(n), n/c)$-partition Γ of G_n can be found in (deterministic) polynomial time.*

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 the 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 $\beta n/4$. 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 an $(\alpha(n), n/c)$-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 5.15). 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 [PSL80, LSP82, FLM86] 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,[17] 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 toward 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_n 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 $\textsc{FinalCut}^{\mathsf{corrupt}}$ 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 $\textsc{RedExec}$, the $\textsc{FinalCut}^{\mathsf{corrupt}}$ 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 $\textsc{InputsAndCoins}$. In both adversarial strategies that will be used in the proof, the corrupted parties will operate in a way that indeed induces the red and blue executions, respectively, 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 $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ (see Figures 10 and 11 and Figures 12 and 13, respectively). 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 the protocol over random inputs with adversarial strategy $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$, respectively $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$.

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

Adversary $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$

The adversary $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ attacks an n-party protocol $\pi_n = (P_1, \ldots, P_n)$. The adversary is parametrized by $\beta_1 = \beta/2$, by $c = \lceil 4/\beta \rceil$, and by a sublinear function $\alpha(n) \in o(n)$, and proceeds as follows:

Phase I: (ensuring high degree in the "red execution")

1. Choose uniformly at random $i^* \leftarrow [n]$.
2. Emulate in its head:
   - A virtual (blue) execution of π_n with virtual parties $(\tilde{Q}_1, \ldots, \tilde{Q}_n)$, where every $\tilde{Q}_i$, for $i \in [n] \setminus \{i^*\}$, is initialized with a uniformly distributed input $\tilde{x}_i \in_R \{0,1\}^n$ and random coins $\tilde{r}_i$. (Party $\tilde{Q}_{i^*}$ is simulated based on the actions of party P_{i^*}.)
   - A virtual party $\tilde{P}_{i^*}$ with a uniformly distributed input $\tilde{x}_{i^*} \in_R \{0,1\}^n$ and random coins $\tilde{r}_{i^*}$. (Party $\tilde{P}_{i^*}$ is used to simulate a (red) execution with real P_i's.)
3. In every round ρ, receive the leakage from f_{psmt} containing the messages for corrupted parties, and which honest party sends a message to another honest party. Next:
   (a) Proceed with round ρ in the *red* execution as follows. For every $j \in [n] \setminus \{i^*\}$ (in lexicographic order) check the following:
      i. If party P_j sends a message to P_{i^*} then corrupt P_j, learn the message μ, and discard the message. Next, simulate party $\tilde{P}_{i^*}$ as receiving message μ from P_j.
      ii. If virtual party $\tilde{P}_{i^*}$ should send a message μ to party P_j, then corrupt P_j and instruct him to play as an honest party that received the message μ from P_{i^*}.
      iii. Let $G_{\mathsf{red}}(\rho, j)$ be the communication graph of the red execution at round ρ, except for messages from $\tilde{P}_{i^*}$ to $P_{j'}$, and from $\tilde{P}_{j'}$ to P_{i^*}, for $j' > j$. If
      $$\deg_{G_{\mathsf{red}}(\rho,j)}(i^*) \geq (\beta_1/2) \cdot n,$$
      then let $G^{\mathsf{honest}}_{\mathsf{phaseII}} = G_{\mathsf{red}}(\rho, j)$, let $\rho^{\mathsf{honest}}_{\mathsf{phaseII}} = \rho$, and proceed to Phase II.
   (b) Proceed with round ρ in the *blue* execution as follows. For every $j \in [n] \setminus \{i^*\}$ (in lexicographic order) check the following:
      i. If P_{i^*} sends a message to party P_j, corrupt P_j, learn the message μ, and instruct P_j to play as an honest party that does not receive messages from P_{i^*}. In addition, simulate virtual party $\tilde{Q}_{i^*}$ sending the message μ to party $\tilde{Q}_j$.
      ii. If virtual party $\tilde{Q}_j$ sends a message μ to virtual party $\tilde{Q}_{i^*}$ then corrupt P_j and send the message μ to party P_{i^*}.
      iii. Let $G_{\mathsf{blue}}(\rho, j)$ be the communication graph of the blue execution at round ρ, except for messages from $\tilde{Q}_{i^*}$ to $\tilde{Q}_{j'}$, and from $\tilde{Q}_{j'}$ to $\tilde{Q}_{i^*}$, for $j' > j$. If
      $$\deg_{G_{\mathsf{blue}}(\rho,j)}(i^*) \geq (\beta_1/2) \cdot n,$$
      then output $\perp$ and halt (the attack fails).
   (c) In case the protocol completes before Phase II, then halt.

Figure 10: (Phase I of Adversary $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$)

Proof structure. Our proof follows from two main steps. In Lemma 5.7, stated and proven 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 $n/2$ 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,

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

In Lemma 5.15, stated and proven in Section 5.3.3, we show that in an execution of π_n 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.,

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

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,

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

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 course 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 toward P_{i^*} in the so-called "blue execution."[18]

The adversary $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$. At a high level, the adversary $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ (see Figures 10 and 11) 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 $\tilde{x}_{i^*}$ toward all other parties. The adversary corrupts any party that sends a message to P_{i^*}, blocks its message, and simulates the virtual party $\tilde{P}_{i^*}$ receiving this message. Whenever the virtual party $\tilde{P}_{i^*}$ should send a message to some P_j, the adversary corrupts party P_j, and instructs him to proceed as if he received the intended message from $\tilde{P}_{i^*}$.

[18] 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.

Adversary $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$

Phase II: (ensure that P_{i^*} remains isolated until his degree in the *real* execution is high (i.e., not in the blue execution))

1. If for some P_i with $i \neq i^*$, it holds $\deg_{G^{\mathsf{honest}}_{\mathsf{phaseII}}}(i) < (\beta_1/2) \cdot n$, output $\perp$ (the attack fails).
2. Identify an $(\alpha, n/c)$-partition of $G^{\mathsf{honest}}_{\mathsf{phaseII}} \setminus \{i^*\}$, denoted $\Gamma_1 = \{U_1, \ldots, U_\ell\}$ (see Definition 5.5).
3. For every round $\rho \geq \rho^{\mathsf{honest}}_{\mathsf{phaseII}}$, proceed with round ρ in the real execution as follows.
   (a) For every $j \in [n] \setminus \{i^*\}$ (in lexicographic order) check the following:
      i. Let $G_{\mathsf{real}}(\rho, j)$ be the communication graph of the real execution at round ρ, except for messages from P_{i^*} to $P_{j'}$, and from $P_{j'}$ to P_{i^*}, for $j' > j$.
      ii. If P_{i^*} talked to P_j in round ρ, and $\deg_{G_{\mathsf{real}}(\rho,j)}(i^*) < (\beta_1/2) \cdot n$, then corrupt P_j and instruct to play honestly as a party that does not receive messages from P_{i^*}.
      iii. If $\deg_{G_{\mathsf{real}}(\rho,j)}(i^*) \geq (\beta_1/2) \cdot n$, then let $G^{\mathsf{honest}}_{\mathsf{phaseIII}} = G_{\mathsf{real}}(\rho, j)$, let $\rho^{\mathsf{honest}}_{\mathsf{phaseIII}} = \rho$, and proceed to Phase III.
   (b) Let $G_{\mathsf{real}}(\rho)$ be the communication graph of the real execution at round ρ.
   (c) For every $i, j \neq i^*$, such that P_i has sent a message to P_j in round ρ:
      i. If $i \in U_k$ and $j \in U_{k'}$, for $k \neq k'$, and $|\mathsf{edges}_{G_{\mathsf{real}}(\rho)}(U_k, U_{k'})| \leq \alpha(n)$, then corrupt P_j, and instruct P_j to play as an honest party that does not send/receive messages to/from parties outside of U_k (from round ρ and onwards).
   (d) In case the protocol completes before Phase III, then halt.

Phase III: (isolate low-weight cuts until the protocol completes)

1. To add i^* to one of the sets in the partition, consider the minimum index of a set U_k for which P_{i^*} has more than $\alpha(n)$ neighbors as

$$k_{\min} = \min \left\{ k : \left| \mathsf{edges}_{G^{\mathsf{honest}}_{\mathsf{phaseIII}}} (\{i^*\}, U_k) \right| > \alpha(n) \right\}.$$

   Set $V_{k_{\min}} = U_{k_{\min}} \bigcup \{i^*\}$ and $V_k = U_k$ for all $k \neq k_{\min}$. Denote $\Gamma_2 = \{V_1, \ldots, V_\ell\}$.[a]
2. For every round $\rho \geq \rho^{\mathsf{honest}}_{\mathsf{phaseIII}}$, proceed with round ρ in the real execution as follows.
   (a) Let $G_{\mathsf{real}}(\rho)$ be the communication graph of the real execution at round ρ.
   (b) For every i and every $j \neq i^*$, such that P_i has sent a message to P_j in round ρ:
      i. If $i \in V_k$ and $j \in V_{k'}$, for $k \neq k'$, and $|\mathsf{edges}_{G_{\mathsf{real}}(\rho)}(V_k, V_{k'})| \leq \alpha(n)$, then corrupt P_j, and instruct P_j to play as an honest party that does not send/receive messages to/from parties outside of V_k (from round ρ and onwards).

[a] Note that Γ_2 *may not* be an (α, d)-partition of the communication graph.

Figure 11: (Phases II and III of Adversary $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$)

- 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 toward 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.

Recall that according to Definition 3.4, edges in which an honest party receives messages will be considered in the final communication graph if the receiver actually processed the messages. In particular, this means that the messages that are blocked during the red execution are not processed by P_{i^*} and will not be considered in the final graph, whereas the messages sent by corrupted parties to P_{i^*} in the blue execution will be considered as long as P_{i^*} will process them.

Adversary $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$

The adversary $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ attacks an n-party protocol $\pi_n = (P_1, \ldots, P_n)$. The adversary is parametrized by $\beta_1 = \beta/2$, by $c = \lceil 4/\beta \rceil$, and by a sublinear function $\alpha(n) \in o(n)$, and proceeds as follows:

Phase I: (ensuring high degree in the "red execution")

1. Choose uniformly at random $i^* \leftarrow [n]$ and corrupt party P_{i^*}.
2. Emulate in its head:
   (a) A virtual (blue) execution of π_n with virtual parties $(\tilde{Q}_1, \ldots, \tilde{Q}_n)$, where every virtual party $\tilde{Q}_i$, for $i \in [n] \setminus \{i^*\}$, is initialized with a uniformly distributed input $\tilde{x}_i \in_R \{0,1\}^n$ and random coins $\tilde{r}_i$. Virtual party $\tilde{Q}_{i^*}$ is initialized with input x_{i^*} and random coins r_{i^*}.
   (b) A virtual party $\tilde{P}_{i^*}$ with a uniformly distributed input $\tilde{x}_{i^*} \in_R \{0,1\}^n$ and random coins $\tilde{r}_{i^*}$. (Party $\tilde{P}_{i^*}$ is used to simulate a (red) execution with real P_i's.)
3. In every round ρ, receive the leakage from f_{psmt} containing the messages for corrupted parties, and which honest party sends a message to another honest party. Next:
   (a) Proceed with round ρ in the *red* execution $(P_1, \ldots, \tilde{P}_{i^*}, \ldots, P_n)$ as follows. For every $j \in [n] \setminus \{i^*\}$ (in lexicographic order) check the following:
      i. If party P_j sends a message μ to P_{i^*}, then simulate $\tilde{P}_{i^*}$ as receiving message μ from P_j.
      ii. If virtual party $\tilde{P}_{i^*}$ generates a message μ for P_j, send μ to P_j on behalf of P_{i^*}.
      iii. Let $G_{\mathsf{red}}(\rho, j)$ be the communication graph of the red execution at round ρ, except for message from $\tilde{P}_{i^*}$ to $P_{j'}$, and from $\tilde{P}_{j'}$ to P_{i^*}, for $j' > j$. If
      $$\deg_{G_{\mathsf{red}}(\rho,j)}(i^*) \geq (\beta_1/2) \cdot n,$$
      then let $G^{\mathsf{corrupt}}_{\mathsf{phaseII}} = G_{\mathsf{red}}(\rho, j)$, let $\rho^{\mathsf{corrupt}}_{\mathsf{phaseII}} = \rho$, and proceed to Phase II.
   (b) Proceed with round ρ in the *blue* execution as follows. Generate all messages for round ρ, and for every $j \in [n] \setminus \{i^*\}$ check the following:
      i. Let $G_{\mathsf{blue}}(\rho, j)$ be the communication graph of the blue execution at round ρ, except for messages from $\tilde{Q}_{i^*}$ to $\tilde{Q}_{j'}$, and from $\tilde{Q}_{j'}$ to $\tilde{Q}_{i^*}$, for $j' > j$. If
      $$\deg_{G_{\mathsf{blue}}(\rho,j)}(i^*) \geq (\beta_1/2) \cdot n,$$
      then output $\perp$ and halt (the attack fails).
   (c) In case the protocol completes before Phase II, then halt.

Figure 12: (Phase I of Adversary $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$)

Adversary $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$

Phase II: (hold back any information about x_{i^*} until the degree of P_{i^*} in the *real* execution is high)

1. If for some party P_i, for $i \neq i^*$, it holds that $\deg_{G^{\mathsf{corrupt}}_{\mathsf{phaseII}}}(i) < (\beta_1/2) \cdot n$, then output $\perp$ and halt (the attack fails).
2. Identify an $(\alpha, (\beta_1/2) \cdot n)$-partition of $G^{\mathsf{corrupt}}_{\mathsf{phaseII}} \setminus \{i^*\}$, denoted $\Gamma_1 = \{U_1, \ldots, U_\ell\}$.
3. For every round $\rho \geq \rho^{\mathsf{corrupt}}_{\mathsf{phaseII}}$, proceed with round ρ in the real execution as follows.
   (a) For every $j \in [n] \setminus \{i^*\}$ (in lexicographic order) check the following:
      i. Let $G_{\mathsf{real}}(\rho, j)$ be the communication graph of the real execution at round ρ, except for message from P_{i^*} to $P_{j'}$, and from $P_{j'}$ to P_{i^*}, for $j' > j$.
      ii. If P_j sends a message μ to P_{i^*} in round ρ, and $\deg_{G_{\mathsf{real}}(\rho,j)}(i^*) < (\beta_1/2) \cdot n$, then simulates virtual party $\tilde{Q}_{i^*}$ receiving the message μ from $\tilde{Q}_j$.
      iii. If $\deg_{G_{\mathsf{real}}(\rho,j)}(i^*) \geq (\beta_1/2) \cdot n$, then let $G^{\mathsf{corrupt}}_{\mathsf{phaseIII}} = G_{\mathsf{real}}(\rho, j)$, let $\rho^{\mathsf{corrupt}}_{\mathsf{phaseIII}} = \rho$, and proceed to Phase III.
   (b) Let $G_{\mathsf{real}}(\rho)$ be the communication graph of the real execution at round ρ.
   (c) For every $i, j \neq i^*$, such that P_i has sent a message to P_j in round ρ:
      i. If $i \in U_k$ and $j \in U_{k'}$, for $k \neq k'$, and $|\mathsf{edges}_{G_{\mathsf{real}}(\rho)}(U_k, U_{k'})| \leq \alpha(n)$, then corrupt P_j, and instruct P_j to play as an honest party that does not send/receive messages to/from parties outside of U_k (from round ρ and onwards).
   (d) In case the protocol completes before Phase III, then halt.

Phase III: (isolate low-weight cuts until the protocol completes)

1. To add i^* to one of the sets in the partition, consider the minimum index of a set U_k for which P_{i^*} has more than $\alpha(n)$ neighbors as

$$k_{\min} = \min \left\{ k : \left| \mathsf{edges}_{G^{\mathsf{honest}}_{\mathsf{phaseIII}}} \left(\{i^*\}, U_k \right) \right| > \alpha(n) \right\}.$$

   Set $V_{k_{\min}} = U_{k_{\min}} \bigcup \{i^*\}$ and $V_k = U_k$ for all $k \neq k_{\min}$. Denote $\Gamma_2 = \{V_1, \ldots, V_\ell\}$.[a]
2. For every round $\rho \geq \rho^{\mathsf{corrupt}}_{\mathsf{phaseIII}}$, proceed with round ρ in the real execution as follows.
   (a) Let $G_{\mathsf{real}}(\rho)$ be the communication graph of the real execution at round ρ.
   (b) For every i and every $j \neq i^*$, such that P_i has sent a message to P_j in round ρ:
      i. If $i \in V_k$ and $j \in V_{k'}$, for $k \neq k'$, and $|\mathsf{edges}_{G_{\mathsf{real}}(\rho)}(V_k, V_{k'})| \leq \alpha(n)$, then corrupt P_j, and instruct P_j to play as an honest party that does not send/receive messages to/from parties outside of V_k (from round ρ and onwards).

[a] Note that Γ_2 *may not* be an (α, d)-partition of the communication graph.

Figure 13: (Phases II and III of Adversary $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$)

Phase II begins when the degree of the virtual party $\tilde{P}_{i^*}$ in the *red* execution is at least $\beta n/4$; 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 $\beta n/4$.

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., $\beta n/2$. 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 $\beta n/4$ corruptions for the red execution and $\beta n/4$ corruptions for the blue execution. We show that with a noticeable probability, once $\tilde{P}_{i^*}$ has degree $\beta n/4$ 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 $(\alpha(n), \beta n/4)$ partition of this graph into a constant number of linear-size subsets that are connected with sublinear many edges, denoted $\Gamma_1 = \{U_1, \ldots, U_\ell\}$ (in particular, this partition is independent of x_{i^*}). In Phase II, the adversary continues blocking outgoing messages from P_{i^*} toward the real honest parties, until the degree of P_{i^*} in the real execution is $\beta n/4$. In addition, $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ 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 $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$. The adversary $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ (see Figures 12 and 13) corrupts the randomly chosen party P_{i^*}, and emulates the operations of an honest P_{i^*} that is being attacked by $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$.

In Phase I, the adversary $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ induces two honestly distributed executions, by simulating an honest execution of a virtual party $\tilde{P}_{i^*}$ over a random input $\tilde{x}_{i^*}$ toward 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). Note that only P_{i^*} is corrupted, hence all messages that are sent by other parties will be considered in the final communication graph, as well as messages sent by P_{i^*} that are processed by the receivers.

In Phase II, the adversary continues simulating the corrupted P_{i^*} toward 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, $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ 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), $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ 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 Proving High Entropy of X_{I^*}

We now proceed to prove the first of the two core lemmata.

Lemma 5.7. *Consider an execution of π_n on random inputs $(X_1, \ldots, X_n)$ for the parties with adversary $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$, 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*

$$H(X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2, \text{VIEW}_{j^*}^{\mathsf{corrupt}}) \geq n/2,$$

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

Proof. We start by showing that for a randomly chosen $I^* \in [n]$, if party P_{I^*} does not send messages to sufficiently many parties, the adversary $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ can violate *validity* of π_n.

Claim 5.8. *Let $G_{\mathsf{phaseII}}^{\mathsf{corrupt}}$ denote the random variable representing the graph of the red execution $(P_1, \ldots, \tilde{P}_{I^*}, \ldots, P_n)$ at the conclusion of Phase I with adversary $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$. Then*

$$\Pr\left[\deg_{G_{\mathsf{phaseII}}^{\mathsf{corrupt}}}(I^*) < (\beta_1/2) \cdot n\right] \leq \mathsf{negl}(n).$$

Proof. Consider an execution of π_n with adversary $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ on random inputs $(X_1, \ldots, X_n)$, with random coins $(R_1, \ldots, R_n)$ for the parties and $(\tilde{X}_1, \ldots, \tilde{X}_n, \tilde{R}_1, \ldots, \tilde{R}_n, I^*)$ for the adversary. Denote $\Pr\left[\deg_{G_{\mathsf{phaseII}}^{\mathsf{corrupt}}}(I^*) < (\beta_1/2) \cdot n\right] = \epsilon(n)$.

In this case, the view of all honest parties (i.e., all parties but P_{I^*}) is identically distributed as their view in an honest execution of the protocol π_n on input $(X_1, \ldots, X_{I^*-1}, \tilde{X}_{I^*}, X_{I^*+1}, \ldots, X_n)$, and in particular, all honest parties output $Y_{I^*}^{\mathsf{corrupt}} = \tilde{X}_{I^*}$ as the I^*'th coordinate of the common output, except for a negligible probability.

This is not sufficient for contradicting validity, since P_{I^*} is corrupted, hence, all that is required is *agreement* on the coordinate $Y_{I^*}^{\mathsf{corrupt}}$. However, consider the adversary $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ (Figures 10 and 11) that instead of corrupting P_{I^*}, isolates him by corrupting all his neighbors.[19]

Because the protocol is defined in the plain model, and the parties do not share correlated randomness (such as PKI), the adversary $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ indeed manages to isolate party P_{I^*} such that:

1. The view of all honest parties, except for party P_{I^*}, is distributed identically as in an honest execution of π_n on inputs $(X_1, \ldots, X_{I^*-1}, \tilde{X}_{I^*}, X_{I^*+1}, \ldots, X_n)$ and random coins $(R_1, \ldots, R_{I^*-1}, \tilde{R}_{I^*}, R_{I^*+1}, \ldots, R_n)$. Denote the communication graph of this distribution by $G_{\mathsf{red}}^{\mathsf{honest}}$.
2. The view of party P_{I^*} is distributed identically as in an honest execution π_n on inputs $(\tilde{X}_1, \ldots, \tilde{X}_{I^*-1}, X_{I^*}, \tilde{X}_{I^*+1}, \ldots, \tilde{X}_n)$ and random coins $(\tilde{R}_1, \ldots, \tilde{R}_{I^*-1}, R_{I^*}, \tilde{R}_{I^*+1}, \ldots, \tilde{R}_n)$. Denote the communication graph of this distribution by $G_{\mathsf{blue}}^{\mathsf{honest}}$.

[19] For now, we are only interested in Phase I of $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$.

Since both executions are distributed like honest executions on random inputs, it holds that

$$\Pr\left[\deg_{G^{\mathsf{honest}}_{\mathsf{red}}}(I^*) < (\beta_1/2)\cdot n\right] = \Pr\left[\deg_{G^{\mathsf{honest}}_{\mathsf{blue}}}(I^*) < (\beta_1/2)\cdot n\right] = \epsilon(n).$$

Therefore, with a non-negligible probability, the number of parties corrupted by $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ is bounded by

$$\deg_{G^{\mathsf{honest}}_{\mathsf{red}}}(I^*) + \deg_{G^{\mathsf{honest}}_{\mathsf{blue}}}(I^*) < \beta_1 \cdot n.$$

Hence, with a non-negligible probability, the execution of π_n with $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ will complete in Phase I, where the view of the set of honest parties, except for P_{I^*} is identically distributed as in an honest execution where P_{I^*} has input $\tilde{X}_{I^*}$, and therefore the I^*'th coordinate of the common output will be $Y_{I^*}^{\mathsf{honest}} = \tilde{X}_{I^*}$. However, since $\beta_1 \cdot n < \beta \cdot n = t$, it follows from the *validity* property, that the I^*'th coordinate of the common output is $Y_{I^*}^{\mathsf{honest}} = X_{I^*}$ except for negligible probability. Furthermore, for every party that is not a neighbour of P_{I^*} it holds that the party is honest and that its view is identically distributed in an execution with $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ as in an execution with $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$, therefore, by *agreement*, $Y_{I^*}^{\mathsf{corrupt}} = Y_{I^*}^{\mathsf{honest}}$ except for negligible probability. Since both X_{I^*} and $\tilde{X}_{I^*}$ are random elements in $\{0,1\}^n$, it holds that $X_{I^*} \neq \tilde{X}_{I^*}$ with a noticeable probability, and we derive a contradiction. □

From Claim 5.8 it follows that with overwhelming probability, $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ will not complete the attack in Step 3c of Phase I. We now turn to bound from below the probability of event $\mathcal{E}_1 \cap \mathcal{E}_2$. Recall events $\mathcal{E}_1$ and $\mathcal{E}_2$ as defined in Section 5.3. Loosely speaking, $\mathcal{E}_1$ is the event that P_{I^*} is the last party to reach high-degree threshold in both red and blue executions. $\mathcal{E}_2$ is the event that P_{I^*} reaches the degree threshold in the red execution before the blue execution.

Claim 5.9. *Consider an execution of protocol π_n on random inputs with adversary $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$. Then*

1. $\Pr[\mathcal{E}_1] \geq 1/n^2 - \mathsf{negl}(n)$.
2. $\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] = 1/2$.
3. $\Pr[\mathcal{E}_2 \cap \mathcal{E}_1] \geq 1/2n^2 - \mathsf{negl}(n)$.

Proof. By Claim 5.8, with overwhelming probability, a random party will have degree $(\beta_1/2)\cdot n$ in an honest execution over random inputs, by the end of the protocol. Thus, *all* parties must reach degree $(\beta_1/2)\cdot n$ with overwhelming probability. By choosing I^* uniformly from $[n]$, we conclude that P_{I^*} will be the *last* party to do so with probability $1/n - \mathsf{negl}(n)$. Since both the red and the blue executions are independent honest executions over random inputs, it follows that P_{I^*} will be last in *both* executions with probability $1/n^2 - \mathsf{negl}(n)$. The second part follows by symmetry, and the third part by definition of conditional probability. □

From Claim 5.9 it follows that with a noticeable probability, the adversary $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ will proceed to Phase III. We now show that the size of the partition Γ_1 (set at the beginning of Phase II) in this case is constant, and only depends on β. In particular, we wish to argue that the identity of the final remaining sublinear cut in the graph cannot reveal too much information about the input X_{I^*}. (Recall that RedExec is the execution with X_{I^*} replaced by $\tilde{X}_{I^*}$.) The proof follows from the graph-theoretic theorem (Theorem 5.6), stated in Section 5.2. We start by looking at the communication graph at the beginning of Phase II *without* the chosen party P_{I^*} (which depends only on the red execution). Party P_{I^*} is added to one of the sets in the partition based on his edges

at the end of Phase II. Finally, we show that given the red-execution graph there are at most 2^{2c} possible choices for the final cut.

Claim 5.10. *Let $c > 1$ be a constant integer satisfying $\beta_1/2 \geq 1/c$. Then, for sufficiently large n, in an execution with $\mathcal{A}_n^{\textsf{corrupt-}i^*}$, conditioned on the event $\mathcal{E}_1 \cap \mathcal{E}_2$, it holds that:*

1. *There exists an $(\alpha(n), n/c)$-partition Γ_1 of $G^{\textsf{corrupt}}_{\textsf{phaseII}} \setminus \{I^*\}$ of size at most c.*
2. *At the end of Phase II there exists a set $U_k \in \Gamma_1$ such that $|\textsf{edges}_{G^{\textsf{corrupt}}_{\textsf{phaseIII}}}(\{I^*\}, U_k)| > \alpha(n)$.*
3. *Conditioned on the event $\mathcal{E}_1 \cap \mathcal{E}_2$ and on RedExec, the random variable $\textsc{FinalCut}^{\textsf{corrupt}}$ has at most $2c$ bits of entropy, i.e.,*

$$H\left(\textsc{FinalCut}^{\textsf{corrupt}} \mid \mathcal{E}_1 \cap \mathcal{E}_2, \textsc{RedExec}\right) \leq 2c.$$

Proof. For $n \in \mathbb{N}$, denote by $G^{\textsf{corrupt}}_{\textsf{phaseII}}(n)$ the "red graph" that is obtained by π_n running on input/randomness from RedExec with $\mathcal{A}_n^{\textsf{corrupt-}i^*}$. From the definition of the events $\mathcal{E}_1$ and $\mathcal{E}_2$, it holds that the degree of every node in the red graph $G^{\textsf{corrupt}}_{\textsf{phaseII}}(n)$ is greater or equal to $(\beta_1/2)\cdot n \geq n/c$. By removing the red node I^* (corresponding to the virtual party $\tilde{P}_{I^*}$ running on input $\tilde{X}_{I^*}$), we obtain the graph $G_{n-1} = G^{\textsf{corrupt}}_{\textsf{phaseII}}(n) \setminus \{I^*\}$ of size $n-1$. Since by removing I^*, the degree of each node reduces by at most 1, it holds that the degree of each vertex in G_{n-1} is at least $n/c - 1$. By applying Theorem 5.6 on the ensemble of graphs $\{G_{n-1}\}_{n\in\mathbb{N}}$, for sufficiently large n, there exists a $(\alpha(n), n/c)$-partition of G_n, denoted $\Gamma_1 = \{U_1, \ldots, U_\ell\}$, of size $\ell < c$.[20]

From the definition of the events $\mathcal{E}_1$ and $\mathcal{E}_2$, it holds that the degree of P_{I^*} in the blue execution will reach the threshold $(\beta_1/2)\cdot n$ (yet not before P_{I^*} reaches the threshold in the red execution). It follows that the attack will enter Phase III. If for every $k \in [\ell]$, it holds that $|\textsf{edges}_{G^{\textsf{corrupt}}_{\textsf{phaseIII}}}(\{I^*\}, U_k)| \leq \alpha(n)$, then the total number of neighbors of P_{I^*} is bounded by $\ell \cdot \alpha(n) \in o(n)$, and we get a contradiction.

Finally, since Γ_1 is an $(\alpha(n), n/c)$-partition of the $(n-1)$-size graph $G^{\textsf{corrupt}}_{\textsf{phaseII}}(n) \setminus \{I^*\}$, it holds that every $\alpha(n)$-cut $\{S_n, \bar{S}_n\}$ of $G^{\textsf{corrupt}}_{\textsf{phaseII}}(n) \setminus \{I^*\}$ can be represented as $S_n = \bigcup_{k\in A} U_k$ and $\bar{S}_n = \bigcup_{k\in[\ell]\setminus A} U_k$ for some $A \subsetneq [\ell]$. There are at most 2^c such subsets. Next, consider the n-size graph $G^{\textsf{corrupt}}_{\textsf{phaseII}}(n)$. When adding the node I^* (and its edges) back to $G^{\textsf{corrupt}}_{\textsf{phaseII}}(n) \setminus \{I^*\}$, the number of potential $\alpha(n)$-cuts at most doubles, since for every potential $\alpha(n)$-cut $\{S, \bar{S}\}$ in $G^{\textsf{corrupt}}_{\textsf{phaseII}}(n) \setminus \{I^*\}$, either $\{S \cup I^*, \bar{S}\}$ is an $\alpha(n)$-cut in $G^{\textsf{corrupt}}_{\textsf{phaseII}}(n)$, or $\{S, \bar{S} \cup I^*\}$ is an $\alpha(n)$-cut in $G^{\textsf{corrupt}}_{\textsf{phaseII}}(n)$ (or neither options induces an $\alpha(n)$-cut in $G^{\textsf{corrupt}}_{\textsf{phaseII}}(n)$). Therefore, there are at most 2^{2c} potential $\alpha(n)$-cuts in $G^{\textsf{corrupt}}_{\textsf{phaseII}}(n)$. It follows that $2c$ bits are sufficient to describe the support of $\textsc{FinalCut}^{\textsf{corrupt}}$ given RedExec (which in particular fully specifies the graph $G^{\textsf{corrupt}}_{\textsf{phaseII}}(n)$). □

In Claims 5.11 to 5.14, we prove that there exists an honest party P_{j^*} that receives a bounded amount of information about X_{I^*} by the end of the protocol's execution. For $n \in \mathbb{N}$, denote by $G^{\textsf{corrupt}}_{\textsf{end}}(n)$ the final communication graph of π_n when running on input/randomness from InputsAndCoins with $\mathcal{A}_n^{\textsf{corrupt-}i^*}$.

[20] To be more precise, applying Theorem 5.6 on $\{G_{n-1}\}_{n\in\mathbb{N}}$ gives an $(\alpha(n-1), (n-1)/c)$-partition. For clarity, we abuse the notation and write $(\alpha(n), n/c)$. This will not affect the subsequent calculations.

Claim 5.11. *Condition on the event $\mathcal{E}_1 \cap \mathcal{E}_2$. Then, at the conclusion of the protocol execution with $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ there exists an $\alpha(n)$-cut, denoted $\{S_n, \bar{S}_n\}$, in the induced graph $G_{end}^{\mathsf{corrupt}}(n)$, and $j^* \in [n]$ such that party P_{j^*} is honest, $I^* \in S_n$, and $j^* \in \bar{S}_n$.*

Proof. By assumption, the communication graph of π is not $\alpha(n)$-edge-connected; therefore, by definition, for sufficiently large n there exists a cut $\{S_n, \bar{S}_n\}$ of weight at most $\alpha(n)$ in $G_{\mathsf{end}}^{\mathsf{corrupt}}(n)$. Let $\{S_n, \bar{S}_n\}$ be such a cut, and assume without loss of generality that $I^* \in S_n$. We now prove that there exists an honest party in the complement set $\bar{S}_n$.

Let $\Gamma_1 = \{U_1, \ldots, U_\ell\}$ be the $(\alpha(n), n/c)$-partition of $G_{\mathsf{phaseII}}^{\mathsf{corrupt}}(n) \setminus \{I^*\}$ defined in Phase II of the attack. By Item 2 of Claim 5.10, at the end of the Phase III there exists $k \in [\ell]$ for which $|\mathsf{edges}(\{I^*\}, U_k)| \geq \alpha(n)$.

Since $\{S_n, \bar{S}_n\}$ is an $\alpha(n)$-cut of $G_{\mathsf{end}}^{\mathsf{corrupt}}(n)$, it holds that $\{S_n \setminus \{I^*\}, \bar{S}_n\}$ is an $\alpha(n)$-cut of $G_{\mathsf{phaseII}}^{\mathsf{corrupt}} \setminus \{I^*\}$ (since any α-cut in a graph remains an α-cut when removing additional edges). It holds that $S_n \setminus \{I^*\} = \bigcup_{k \in A} U_k$ for some $\emptyset \neq A \subsetneq [\ell]$. Similarly, $\bar{S}_n = \bigcup_{k \in [\ell] \setminus A} U_k$. Recall that in an execution with $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ we have the following corruption pattern:

- During Phase I only P_{I^*} gets corrupted.
- During Phase II, for every pair of $U_k, U_{k'} \in \Gamma_1$, at most $\alpha(n)$ communicating parties get corrupted.
- During Phase III, for every pair of $V_k, V_{k'} \in \Gamma_2$, at most $\alpha(n)$ communicating parties get corrupted. (Recall that $V_k \in \Gamma_2$ either equals U_k or equals $U_k \cup \{I^*\}$.)

It follows that there are at most $\ell^2 \cdot \alpha(n)$ corrupted parties in $\bar{S}_n$. By definition, $|\bar{S}_n| \geq n/c$ (since $|U_k| \geq n/c$, for each $k \in [\ell]$). By Item 1 of Claim 5.10, $\ell \leq c$ for some constant c, meaning that there is a linear number of parties in each side of the cut, but only a sublinear number of corruptions, and the claim follows. □

We now prove that the view of an honest $P_{j^*} \in \bar{S}_n$ can be perfectly simulated given: (1) All inputs and random coins that were used in the *red* execution. This information is captured in the random variable RedExec, and deterministically determines the partition Γ_1 at the beginning of Phase II. (2) The identities of the parties in $\bar{S}_n$. This information is captured in the random variable $\text{FinalCut}^{\mathsf{corrupt}}$.

Claim 5.12. *Conditioned on the event $\mathcal{E}_1 \cap \mathcal{E}_2$, the view $\text{view}_{j^*}^{\mathsf{corrupt}}$ of honest party P_{j^*} at the end of the protocol is simulatable by RedExec and by $\text{FinalCut}^{\mathsf{corrupt}}$.*

Proof. Let $\{S_n, \bar{S}_n\}$ be the $\alpha(n)$-cut that is guaranteed to exist at the end of the protocol (the "minimal" such cut according to some canonical ordering). Assume that $I^* \in S_n$, and let $j^* \in \bar{S}_n$ be an index of the honest party that exists by Claim 5.11. The view of P_{j^*} is defined as its input, random coins, and the messages it received during the protocol.[21]

During Phase I the view of P_{j^*} is identically distributed as a view of an honest party in an honest execution over inputs $(X_1, \ldots, \tilde{X}_{I^*}, \ldots, X_n)$ and random coins $(R_1, \ldots, \tilde{R}_{I^*}, \ldots, R_n)$ (without loss of generality, we can assume that the joint view of all honest parties is exactly $(X_1, \ldots, \tilde{X}_{I^*}, \ldots, X_n)$ and $(R_1, \ldots, \tilde{R}_{I^*}, \ldots, R_n)$, i.e., all the information that deterministically defines the red execution).

[21] Formally, in every round every party receives a vector of n messages from f_{psmt}, where some may be the empty string. Therefore, a party also knows the identity of the sender of every message.

Indeed, this can be easily simulated given the random variable RedExec by running an honest execution until all parties have degree $(\beta_1/2) \cdot n$. Furthermore, the partition $\Gamma_1 = (U_1, \ldots, U_\ell)$ is deterministically determined by RedExec, as well as the view of every honest party at the end of Phase I (except for P_{I^*}).

Next, consider the parties in $\bar{S}_n$ (which are determined by the random variable $\text{FinalCut}^{\mathsf{corrupt}}$). By Theorem 5.6, for large enough n it holds that $\bar{S}_n = \cup_{k \in A} U_k$ for some nonempty $A \subsetneq [\ell]$. That is, given $\Gamma_1 = \{U_1, \ldots, U_\ell\}$, the random variable $\text{FinalCut}^{\mathsf{corrupt}}$ is fully specified by $A \subseteq [\ell]$, which can be described as an element of $\{0,1\}^\ell$. As before, and without loss of generality, for every $j \in \bar{S}_n$, the view of party P_j at the end of Phase I can also be set as $(X_1, \ldots, \tilde{X}_{I^*}, \ldots, X_n)$ and $(R_1, \ldots, \tilde{R}_{I^*}, \ldots, R_n)$.

Observe that given the final cut $\{S_n, \bar{S}_n\}$, it holds that during Phases II and III every message that is being sent to a party in $\bar{S}_n$ by a party in S_n is ignored. Therefore, it may seem that the simulation can be resumed by continuing an honest execution of all parties in $\bar{S}_n$ based on their joint view at the end of Phase I. However, this approach will not suffice since the adversary in the real execution blocks any message that is sent by parties in different U_k's, even if both parties are members of $\bar{S}_n$. To simulate this behavior, it is important to know exactly when to block a message sent between different U_k's and when to keep it. Indeed, this is the reason for keeping track of the number of edges between every pair of U_k's and blocking message only until the threshold $\alpha(n)$ is reached.

The simulation of Phases II and III therefore proceeds by running the protocol honestly for every P_j, with $j \in \bar{S}_n$, however, with the following two exceptions. First, every party is simulated as if he does not receive any message from parties outside of $\bar{S}_n$, and whenever he is instructed to send a message to a party outside of $\bar{S}_n$, a dummy party is simulated as receiving this message. Second, any message that is sent between two parties P_{j_1} and P_{j_2} is discarded whenever $j_1 \in U_k$ and $j_2 \in U_{k'}$ (respectively, $j_1 \in V_k$ and $j_2 \in V_{k'} \setminus \{I^*\}$), for some $k \neq k'$, and it holds that $|\mathsf{edges}(U_k, U_{k'})| < \alpha(n)$ (respectively, $|\mathsf{edges}(V_k, V_{k'})| < \alpha(n)$).

The above simulation identically emulates the view of very honest party in $\bar{S}_n$, since the adversary indeed discards every message that is sent from some party in S_n to some party in $\bar{S}_n$, as well as from parties in different U_k's, but otherwise behaves honestly. □

The core of the proof (Claim 5.14) is showing that a constant number of bits suffices to describe $\text{FinalCut}^{\mathsf{corrupt}}$. We note that while RedExec is independent of X_{I^*} conditioned on $\mathcal{E}_1$ and on I^* (as shown in Claim 5.13 below), there is some correlation between $\text{FinalCut}^{\mathsf{corrupt}}$ and X_{I^*}.

Claim 5.13. *Conditioned on I^* and on the event $\mathcal{E}_1$, the random variables X_{I^*} and RedExec are independent. That is,*

$$I\left(X_{I^*}; \textit{RedExec} \mid \mathcal{E}_1, I^*\right) = 0.$$

Proof. Recall that over the sampling of

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

the random variable RedExec is defined as

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

Similarly, define

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

Note that (even) given the value of I^*, RedExec and BlueExec are simply uniform distributions, independent of each other. That is, for every $i^* \in [n]$

$$I\left(\textsc{BlueExec}\ ;\ \textsc{RedExec} \mid I^* = i^*\right) = 0.$$

This in turn implies that

$$I\left(\textsc{BlueExec}\ ;\ \textsc{RedExec} \mid I^*\right) = 0.$$

We observe that given I^*, the event $\mathcal{E}_1$ decomposes as a conjunction of independent events. Namely, consider the (deterministic) predicate

$$\textsc{LastParty}\left(\left(x_{-i^*}, x_{i^*}, r_{-i^*}, r_{i^*}\right), i^*\right),$$

which simulates an honest execution of π_n with corresponding inputs and randomness, and outputs 1 if party P_{i^*} is the last party to reach degree $\beta n/4$. Then,

$$\mathcal{E}_1 = \left(\textsc{LastParty}\left(\textsc{BlueExec}, I^*\right) = 1\right) \wedge \left(\textsc{LastParty}\left(\textsc{RedExec}, I^*\right) = 1\right).$$

It follows that

$$\begin{aligned} I\left(X_{I^*}\ ;\ \textsc{RedExec} \mid I^*, \mathcal{E}_1\right) &\leq I\left(\textsc{BlueExec}\ ;\ \textsc{RedExec} \mid I^*, \mathcal{E}_1\right) \\ &= I\left(\begin{array}{c|c} \textsc{BlueExec}\ ;\ \textsc{RedExec} & \begin{array}{c} I^* \\ \textsc{LastParty}\left(\textsc{BlueExec}, I^*\right) = 1 \\ \textsc{LastParty}\left(\textsc{RedExec}, I^*\right) = 1 \end{array} \end{array}\right). \end{aligned}$$

We will show that the last term is equal to zero. This proves our claim, as mutual information cannot be negative.

$$\begin{aligned} 0 &= I\left(\textsc{BlueExec}\ ;\ \textsc{RedExec} \mid I^*\right) \\ &= I\left(\textsc{BlueExec}, \textsc{LastParty}\left(\textsc{BlueExec}, I^*\right)\ ;\ \textsc{RedExec}, \textsc{LastParty}\left(\textsc{RedExec}, I^*\right) \mid I^*\right) \\ &\geq I\left(\begin{array}{c|c} \textsc{BlueExec}\ ;\ \textsc{RedExec} & \begin{array}{c} I^* \\ \textsc{LastParty}\left(\textsc{BlueExec}, I^*\right) \\ \textsc{LastParty}\left(\textsc{RedExec}, I^*\right) \end{array} \end{array}\right), \end{aligned}$$

where the second equality follows from the fact that LastParty is a deterministic function of inputs and randomness of all the parties and I^*. This implies that

$$I\left(\begin{array}{c|c} \textsc{BlueExec}\ ;\ \textsc{RedExec} & \begin{array}{c} I^* \\ \textsc{LastParty}\left(\textsc{BlueExec}, I^*\right) \\ \textsc{LastParty}\left(\textsc{RedExec}, I^*\right) \end{array} \end{array}\right) = 0.$$

Finally, since the event $\mathcal{E}_1$ occurs with non-zero probability, it holds that

$$I\left(\begin{array}{c|c} \textsc{BlueExec}\ ;\ \textsc{RedExec} & \begin{array}{c} I^* \\ \textsc{LastParty}\left(\textsc{BlueExec}, I^*\right) = 1 \\ \textsc{LastParty}\left(\textsc{RedExec}, I^*\right) = 1 \end{array} \end{array}\right) = 0,$$

This concludes the proof of Claim 5.13. □

Claim 5.14. *For sufficiently large n, conditioned on the event $\mathcal{E}_1 \cap \mathcal{E}_2$, the input X_{I^*} retains $n/2$ bits of entropy given the view of honest party P_{j^*}, i.e.,*

$$H(X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2, \text{VIEW}_{j^*}^{\mathsf{corrupt}}) \geq n/2.$$

Proof. First, since $\Pr[\mathcal{E}_1] \geq 1/n^2$, it holds that

$$H(X_{I^*} \mid \mathcal{E}_1) \geq n - \log(n^2). \tag{2}$$

Indeed, for an arbitrary $x \in \{0,1\}^n$,

$$\Pr[X_{I^*} = x \mid \mathcal{E}_1] = \frac{\Pr[X_{I^*} = x, \mathcal{E}_1]}{\Pr[\mathcal{E}_1]} \leq \frac{\Pr[X_{I^*} = x]}{\Pr[\mathcal{E}_1]} \leq \frac{n^2}{2^n},$$

where the last inequality uses the fact that X_{I^*} is uniformly distributed in $\{0,1\}^n$ and that $\Pr[\mathcal{E}_1] \geq 1/n^2$. Equation (2) follows by the following computation.

$$\begin{aligned}
H(X_{I^*} \mid \mathcal{E}_1) &= \sum_{x \in \{0,1\}^n} \Pr[X_{I^*} = x \mid \mathcal{E}_1] \cdot \log\left(\frac{1}{\Pr[X_{I^*} = x \mid \mathcal{E}_1]}\right) \\
&\geq \sum_{x \in \{0,1\}^n} \Pr[X_{I^*} = x \mid \mathcal{E}_1] \cdot \log\left(\frac{2^n}{n^2}\right) \\
&= \log\left(\frac{2^n}{n^2}\right) \cdot \underbrace{\sum_{x \in \{0,1\}^n} \Pr[X_{I^*} = x \mid \mathcal{E}_1]}_{=1} \\
&= n - \log(n^2).
\end{aligned}$$

Since I^* represents an element of the set $[n]$, the support of I^* is $\{0,1\}^{\log(n)}$, and it holds that

$$H(X_{I^*} \mid \mathcal{E}_1, I^*) \geq H(X_{I^*} \mid \mathcal{E}_1) - \log(n). \tag{3}$$

In addition, since conditioned on $\mathcal{E}_1$ and I^*, the random variables RedExec and X_{I^*} are independent (Claim 5.13), it holds that

$$\begin{aligned}
H(X_{I^*} \mid \mathcal{E}_1, I^*, \text{RedExec}) &= H(X_{I^*} \mid \mathcal{E}_1, I^*) - I(X_{I^*}; \text{RedExec} \mid \mathcal{E}_1, I^*) \\
&= H(X_{I^*} \mid \mathcal{E}_1, I^*).
\end{aligned} \tag{4}$$

Next, since $\Pr[\mathcal{E}_2 \mid \mathcal{E}_1] = 1/2$ (Claim 5.9), it follows that

$$H(X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2, I^*, \text{RedExec}) \geq 2 \cdot H(X_{I^*} \mid \mathcal{E}_1, I^*, \text{RedExec}) - n - 2. \tag{5}$$

To prove this, we define an indicator random variable Ind_2 for the event $\mathcal{E}_2$, i.e., $\Pr[\text{Ind}_2 = 1] = \Pr[\mathcal{E}_2]$, and use the fact that since Ind_2 is an indicator random variable, the mutual information $I(X_{I^*}; \text{Ind}_2 \mid \text{RedExec}, \mathcal{E}_1, I^*)$ cannot be more than 1. Therefore,

$$\begin{aligned}
H(X_{I^*} \mid \mathcal{E}_1, I^*, \text{RedExec}, \text{Ind}_2) &= H(X_{I^*} \mid \mathcal{E}_1, I^*, \text{RedExec}) - I(X_{I^*}; \text{Ind}_2 \mid \mathcal{E}_1, I^*, \text{RedExec}) \\
&\geq H(X_{I^*} \mid \mathcal{E}_1, I^*, \text{RedExec}) - 1.
\end{aligned}$$

Equation (5) now follows by the following computation.

$$\begin{aligned}
H(X_{I^*} \mid \mathcal{E}_1, I^*, \textsc{RedExec}) - 1 &\leq H(X_{I^*} \mid \mathcal{E}_1, I^*, \textsc{RedExec}, \textsc{Ind}_2) \\
&= \Pr[\textsc{Ind}_2 = 1 \mid \mathcal{E}_1] \cdot H(X_{I^*} \mid \mathcal{E}_1, I^*, \textsc{RedExec}, \textsc{Ind}_2 = 1) \\
&\quad + \Pr[\textsc{Ind}_2 = 0 \mid \mathcal{E}_1] \cdot \underbrace{H(X_{I^*} \mid \mathcal{E}_1, I^*, \textsc{RedExec}, \textsc{Ind}_2 = 0)}_{\leq n} \\
&\leq \underbrace{\Pr[\mathcal{E}_2 \mid \mathcal{E}_1]}_{=1/2} \cdot H(X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2, I^*, \textsc{RedExec}) + \underbrace{\Pr[\neg\mathcal{E}_2 \mid \mathcal{E}_1]}_{=1/2} \cdot n \\
&\leq 1/2 \cdot H(X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2, I^*, \textsc{RedExec}) + n/2.
\end{aligned}$$

By Claim 5.10, the support of $\textsc{FinalCut}^{\mathsf{corrupt}}$ is $\{0,1\}^{2c}$ and it holds that

$$\begin{aligned}
&H(X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2, I^*, \textsc{RedExec}, \textsc{FinalCut}^{\mathsf{corrupt}}) \\
&\quad = H(X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2, I^*, \textsc{RedExec}) - I\left(X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2, I^*, \textsc{RedExec}, \textsc{FinalCut}^{\mathsf{corrupt}}\right) \\
&\quad \geq H(X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2, I^*, \textsc{RedExec}) - H(\textsc{FinalCut}^{\mathsf{corrupt}} \mid \mathcal{E}_1 \cap \mathcal{E}_2, I^*, \textsc{RedExec}) \\
&\quad \geq H(X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2, I^*, \textsc{RedExec}) - 2c. \qquad (6)
\end{aligned}$$

The first inequality holds since the conditional mutual information is upper-bounded by the conditional entropy term. Finally, by Claim 5.12 it holds that

$$H(X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2, \textsc{view}_{j^*}^{\mathsf{corrupt}}) \geq H(X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2, I^*, \textsc{RedExec}, \textsc{FinalCut}^{\mathsf{corrupt}}). \qquad (7)$$

By combining Equations (2) to (7) together, it holds that

$$\begin{aligned}
H(X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2, \textsc{view}_{j^*}^{\mathsf{corrupt}}) &\geq H(X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2, I^*, \textsc{RedExec}, \textsc{FinalCut}^{\mathsf{corrupt}}) \\
&\geq H(X_{I^*} \mid \mathcal{E}_1 \cap \mathcal{E}_2, I^*, \textsc{RedExec}) - 2c \\
&\geq 2 \cdot H(X_{I^*} \mid \mathcal{E}_1, I^*, \textsc{RedExec}) - n - 2 - 2c \\
&= 2 \cdot H(X_{I^*} \mid \mathcal{E}_1, I^*) - n - 2 - 2c \\
&\geq 2 \cdot H(X_{I^*} \mid \mathcal{E}_1) - 2 \cdot \log(n) - n - 2 - 2c \\
&\geq 2n - 2 \cdot \log(n^2) - 2 \cdot \log(n) - n - 2 - 2c \\
&\geq n - 2 \cdot (\log(n^2) + \log(n) + 1 + c).
\end{aligned}$$

The claim follows for sufficiently large n. □

This concludes the proof of Lemma 5.7. □

5.3.3 Proving the Common Output Contains X_{I^*}

We now turn to the second main lemma of the proof. We show that although party P_{I^*} is corrupted in the execution with $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$, all honest parties must output its initial input value X_{I^*} at the conclusion of the protocol. This is done by analyzing an execution with the dual adversary, $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ (described in Section 5.3.1), that does not corrupt the chosen party P_{I^*} but simulates the attack by $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ toward the honest parties.

Lemma 5.15. *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.,*

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

Proof. The first part of the lemma follows (almost) from the *validity* property of parallel broadcast. It is only left to prove that the event $\mathcal{E}_1 \cap \mathcal{E}_2$ is non-negligible in an execution with $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$. However, as $\mathcal{E}_1$ and $\mathcal{E}_2$ are events on honest executions (mirrored by both $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ and $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ in Phase I), this follows by Claim 5.9. We spell this out in Claims 5.16 and 5.17.

Claim 5.16. *Let $G_{\mathsf{phaseII}}^{\mathsf{honest}}$ denote the random variable representing the graph of the red execution $(P_1, \ldots, \tilde{P}_{I^*}, \ldots, P_n)$ at the conclusion of Phase I with adversary $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$. Then*

$$\Pr\left[\deg_{G_{\mathsf{phaseII}}^{\mathsf{honest}}}(I^*) < (\beta_1/2) \cdot n\right] \leq \mathsf{negl}(n).$$

Proof. As proven in Claim 5.8, in an execution on random inputs with adversary $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$, it holds that the degree of every party reaches $(\beta_1/2) \cdot n$ except for negligible probability. □

Claim 5.17. *Consider an execution of protocol π_n on random inputs with adversary $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$. Then*

$$\Pr\left[\mathcal{E}_2 \cap \mathcal{E}_1\right] \geq 1/2n^2 - \mathsf{negl}(n).$$

Proof. By Claim 5.16 every party has degree $(\beta_1/2) \cdot n$ with overwhelming probability. The rest of the proof follows exactly as the proof of Claim 5.9. □

This complete the proof of the first part of the Lemma 5.15.

For proving the second part of the lemma, we show that in an execution with $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ there exists an honest party whose view, and in particular his output, is identically distributed as in an execution with $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$. This follows from a simple counting argument.

First, denote by C_n^{honest} the random variable representing the set of corrupted parties in an execution with $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$. Since $\beta < 1/3$ and there are at most βn corrupted parties, it holds that $|C_n^{\mathsf{honest}}| < n/3$. Note that by construction, $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ corrupts during Phase I all neighbors of the virtual party $\tilde{P}_{I^*}$ in the red execution and all neighbors of P_{I^*} in the blue execution, and in Phases II and III, all parties that belong to some $U_k \in \Gamma_1$ (respectively, $V_k \in \Gamma_2$) and receive messages from parties in $U_{k'}$ (respectively, $V_{k'}$) for $k \neq k'$ (until there are $\alpha(n)$ such edges, and except for P_{I^*}).

Second, denote by C_n^{corrupt} the random variable representing the following set of parties in an execution with $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$ (note that these parties are not necessarily corrupted here): All neighbors of the virtual party $\tilde{P}_{I^*}$ in the red execution and all neighbors of P_{I^*} in the blue execution during Phase I, and all parties that get corrupted during Phases II and III, i.e., all parties that belong

to some $U_k \in \Gamma_1$ (respectively, $V_k \in \Gamma_2$) and receive messages from parties in $U_{k'}$ (respectively, $V_{k'}$) for $k \neq k'$ (until there are $\alpha(n)$ such edges, and except for P_{I^*}). By symmetry, it holds that $|C_n^{\mathsf{corrupt}}| < n/3$.

For every value of InputsAndCoins, the size of the union of these sets $C_n^{\mathsf{honest}} \cup C_n^{\mathsf{corrupt}}$ is at most $2n/3$. By the constructions of $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ and $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$, the view of any party outside of this union is identically distributed in an execution with $\mathcal{A}_n^{\mathsf{honest}\text{-}i^*}$ as in an execution with $\mathcal{A}_n^{\mathsf{corrupt}\text{-}i^*}$, conditioned on the event $\mathcal{E}_1 \cap \mathcal{E}_2$. Indeed, in both executions the view of every party outside of $C_n^{\mathsf{honest}} \cup C_n^{\mathsf{corrupt}}$ is distributed according to an honest execution on random inputs during Phase I. During Phase II, the view is distributed according the continuation of the protocol under omission failures between every pair $U_k, U_{k'} \in \Gamma_1$. During Phase III, the view is distributed according the continuation of the protocol under omission failures between every pair $V_k, V_{k'} \in \Gamma_2$, with the exception that the internal state of a random party P_{I^*} is replaced by a view of P_{I^*} in a honest execution with an independently distributed random inputs for all parties.

This concludes the proof of Lemma 5.15. □

Bibliography

[ACD+19] Ittai Abraham, T.-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In *Proceedings of the 38th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 317–326, 2019.

[ACdH06] Saurabh Agarwal, Ronald Cramer, and Robbert de Haan. Asymptotically optimal two-round perfectly secure message transmission. In *Advances in Cryptology – CRYPTO 2006*, pages 394–408, 2006.

[ALM17] Adi Akavia, Rio LaVigne, and Tal Moran. Topology-hiding computation on all graphs. In *Advances in Cryptology – CRYPTO 2017, part I*, pages 447–467, 2017.

[AM17] Adi Akavia and Tal Moran. Topology-hiding computation beyond logarithmic diameter. In *Advances in Cryptology – EUROCRYPT 2017, part III*, pages 609–637, 2017.

[BBC+19] Marshall Ball, Elette Boyle, Ran Cohen, Tal Malkin, and Tal Moran. Is information-theoretic topology-hiding computation possible? In *Proceedings of the 17th Theory of Cryptography Conference, TCC 2019, part I*, pages 502–530, 2019.

[BBMM18] Marshall Ball, Elette Boyle, Tal Malkin, and Tal Moran. Exploring the boundaries of topology-hiding computation. In *Advances in Cryptology – EUROCRYPT 2018, part III*, pages 294–325, 2018.

[BCDH18] Elette Boyle, Ran Cohen, Deepesh Data, and Pavel Hubáček. Must the communication graph of MPC protocols be an expander? In *Advances in Cryptology – CRYPTO 2018, part III*, pages 243–272, 2018.

[BCG21] Elette Boyle, Ran Cohen, and Aarushi Goel. Breaking the O($\sqrt{}$ n)-bit barrier: Byzantine agreement with polylog bits per party. In *Proceedings of the 40th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 319–330, 2021.

[BCP15] Elette Boyle, Kai-Min Chung, and Rafael Pass. Large-scale secure computation: Multi-party computation for (parallel) RAM programs. In *Advances in Cryptology – CRYPTO 2015, part II*, pages 742–762, 2015.

[Bea91] Donald Beaver. Foundations of secure interactive computing. In *Advances in Cryptology – CRYPTO '91*, pages 377–391, 1991.

[Bei07] Amos Beimel. On private computation in incomplete networks. *Distributed Computing*, 19(3):237–252, 2007.

[BF99] Amos Beimel and Matthew K. Franklin. Reliable communication over partially authenticated networks. *Theoretical Computer Science*, 220(1):185–210, 1999.

[BG93] Piotr Berman and Juan A. Garay. Fast consensus in networks of bounded degree. *Distributed Computing*, 7(2):67–73, 1993.

[BGH13] Nicolas Braud-Santoni, Rachid Guerraoui, and Florian Huc. Fast byzantine agreement. In *Proceedings of the 32th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 57–64, 2013.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology – EUROCRYPT 2003*, pages 416–432, 2003.

[BGT11] Elette Boyle, Shafi Goldwasser, and Yael Tauman Kalai. Leakage-resilient coin tossing. In *Proceedings of the 25th International Symposium on Distributed Computing (DISC)*, pages 181–196, 2011.

[BGT13] Elette Boyle, Shafi Goldwasser, and Stefano 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.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 1988.

[BJLM06] Markus Bläser, Andreas Jakoby, Maciej Liśkiewicz, and Bodo Manthey. Private computation: k-connected versus 1-connected networks. *Journal of Cryptology*, 19(3):341–357, 2006.

[BJLM11] Markus Bläser, Andreas Jakoby, Maciej Liśkiewicz, and Bodo Manthey. Privacy in non-private environments. *Theory Comput. Syst.*, 48(1):211–245, 2011.

[BM05] Amos Beimel and Lior Malka. Efficient reliable communication over partially authenticated networks. *Distributed Computing*, 18(1):1–19, 2005.

[BMR90] Donald Beaver, Silvio Micali, and Phillip 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.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[Can01] Ran 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.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 11–19, 1988.

[CCG+15] Nishanth Chandran, Wutichai Chongchitmate, Juan A. Garay, Shafi Goldwasser, Rafail Ostrovsky, and Vassilis 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.

[CCGZ19] Ran Cohen, Sandro Coretti, Juan A. Garay, and Vassilis Zikas. Probabilistic termination and composability of cryptographic protocols. *Journal of Cryptology*, 32(3):690–741, 2019.

[CCGZ21] Ran Cohen, Sandro Coretti, Juan A. Garay, and Vassilis Zikas. Round-preserving parallel composition of probabilistic-termination cryptographic protocols. *Journal of Cryptology*, 34(2):12, 2021.

[CDD+99] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Advances in Cryptology – EUROCRYPT '99*, pages 311–326, 1999.

[CDF01] Ronald Cramer, Ivan Damgård, and Serge Fehr. On the cost of reconstructing a secret, or VSS with optimal reconstruction phase. In *Advances in Cryptology – CRYPTO 2001*, pages 503–523, 2001.

[CFOR12] Alfonso Cevallos, Serge Fehr, Rafail Ostrovsky, and Yuval Rabani. Unconditionally-secure robust secret sharing with compact shares. In *Advances in Cryptology – EUROCRYPT 2012*, pages 195–208, 2012.

[CGO10] Nishanth Chandran, Juan A. Garay, and Rafail 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.

[CGO12] Nishanth Chandran, Juan A. Garay, and Rafail 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.

[CGO15] Nishanth Chandran, Juan A. Garay, and Rafail Ostrovsky. Almost-everywhere secure computation with edge corruptions. *Journal of Cryptology*, 28(4):745–768, 2015.

[CHOR18] Ran Cohen, Iftach Haitner, Eran Omri, and Lior Rotem. Characterization of secure multiparty computation without broadcast. *Journal of Cryptology*, 31(2):587–609, 2018.

[CL17] Ran Cohen and Yehuda Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. *Journal of Cryptology*, 30(4):1157–1186, 2017.

[DDWY93] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. *Journal of the ACM*, 40(1):17–47, 1993.

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *Advances in Cryptology – CRYPTO 2005*, pages 378–394, 2005.

[DKM+17] Varsha Dani, Valerie King, Mahnush Movahedi, Jared Saia, and Mahdi Zamani. Secure multiparty computation in large networks. *Distributed Computing*, 30(3):193–229, 2017.

[Dol82] Danny Dolev. The byzantine generals strike again. *J. Algorithms*, 3(1):14–30, 1982.

[DPPU88] Cynthia Dwork, David Peleg, Nicholas Pippenger, and Eli Upfal. Fault tolerance in networks of bounded degree. *SIAM Journal on Computing*, 17(5):975–988, 1988.

[DS83] Danny Dolev and Raymond Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

[DW10] Zeev Dvir and Avi Wigderson. Monotone expanders: Constructions and applications. *Theory of Computing*, 6(1):291–308, 2010.

[Fei99] Uriel Feige. Noncryptographic selection protocols. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 142–153, 1999.

[FFGS07] Matthias Fitzi, Matthew K. Franklin, Juan A. Garay, and Harsha Vardhan Simhadri. Towards optimal and efficient perfectly secure message transmission. In *Proceedings of the Fourth Theory of Cryptography Conference, TCC 2007*, pages 311–322, 2007.

[FLM86] Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1(1):26–39, 1986.

[FM97] Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997.

[FY04] Matthew K. Franklin and Moti Yung. Secure hypergraphs: Privacy from partial broadcast. *SIAM Journal on Discrete Mathematics*, 18(3):437–450, 2004.

[GKKZ11] Juan A. Garay, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. Adaptively secure broadcast, revisited. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 179–186, 2011.

[GL90] Shafi Goldwasser and Leonid A. Levin. Fair computation of general functions in presence of immoral majority. In *Advances in Cryptology – CRYPTO '90*, pages 77–93, 1990.

[GM93] Juan A. Garay and Yoram 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.

[GMW87] Oded Goldreich, Silvio Micali, and Avi 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.

[GO08] Juan A. Garay and Rafail Ostrovsky. Almost-everywhere secure computation. In *Advances in Cryptology – EUROCRYPT 2008*, pages 307–323, 2008.

[Gol04] Oded Goldreich. *Foundations of Cryptography – VOLUME 2: Basic Applications*. Cambridge University Press, 2004.

[Gol11] Oded Goldreich. A sample of samplers: A computational perspective on sampling. In *Studies in Complexity and Cryptography*, pages 302–332. Springer, 2011.

[GVZ06] Ronen Gradwohl, Salil P. Vadhan, and David Zuckerman. Random selection with an adversarial majority. In *Advances in Cryptology – CRYPTO 2006*, pages 409–426, 2006.

[HIJ+16] Shai Halevi, Yuval Ishai, Abhishek Jain, Eyal Kushilevitz, and Tal 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.

[HIK07] Danny Harnik, Yuval Ishai, and Eyal Kushilevitz. How many oblivious transfers are needed for secure multiparty computation? In *Advances in Cryptology – CRYPTO 2007*, pages 284–302, 2007.

[HLP11] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In *Advances in Cryptology – CRYPTO 2011*, pages 132–150, 2011.

[HLW06] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc.*, 43(4):439–561, 2006.

[HMTZ16] Martin Hirt, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Network-hiding communication and applications to multi-party protocols. In *Advances in Cryptology – CRYPTO 2016, part II*, pages 335–365, 2016.

[HZ10] Martin Hirt and Vassilis Zikas. Adaptively secure broadcast. In *Advances in Cryptology – EUROCRYPT 2010*, pages 466–485, 2010.

[IOZ14] Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In *Advances in Cryptology – CRYPTO 2014, part II*, pages 369–386, 2014.

[KGSR02] M. V. N. Ashwin Kumar, Pranava R. Goundan, K. Srinathan, and C. Pandu 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.

[KKK+08] Bruce M. Kapron, David Kempe, Valerie King, Jared Saia, and Vishal 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.

[KLST11] Valerie King, Steven Lonargan, Jared Saia, and Amitabh 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.

[KRS16] Ranjit Kumaresan, Srinivasan Raghuraman, and Adam Sealfon. Network oblivious transfer. In *Advances in Cryptology – CRYPTO 2016, part II*, pages 366–396, 2016.

[KS09a] Valerie King and Jared 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.

[KS09b] Kaoru Kurosawa and Kazuhiro Suzuki. Truly efficient 2-round perfectly secure message transmission scheme. *IEEE Transactions on Information Theory*, 55(11):5223–5232, 2009.

[KS10] Valerie King and Jared 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.

[KSSV06] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 990–999, 2006.

[LMRS04] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In *Advances in Cryptology – EUROCRYPT 2004*, pages 74–90, 2004.

[LOS+13] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures, multisignatures, and verifiably encrypted signatures without random oracles. *Journal of Cryptology*, 26(2):340–373, 2013.

[LSP82] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[MOR01] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: extended abstract. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS)*, pages 245–254, 2001.

[MOR15] Tal Moran, Ilan Orlov, and Silas Richelson. Topology-hiding computation. In *Proceedings of the 12th Theory of Cryptography Conference, TCC 2015, part I*, pages 159–181, 2015.

[MR91] Silvio Micali and Phillip Rogaway. Secure computation (abstract). In *Advances in Cryptology – CRYPTO '91*, pages 392–404, 1991.

[PSL80] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.

[RB89] Tal Rabin and Michael 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.

[SA96] Hasan Md. Sayeed and Hosame Abu-Amara. Efficient perfectly secure message transmission in synchronous networks. *Information and Control*, 126(1):53–61, 1996.

[SASM10] Takenobu Seito, Tadashi Aikawa, Junji Shikata, and Tsutomu Matsumoto. Information-theoretically secure key-insulated multireceiver authentication codes. In *Progress in Cryptology - AFRICACRYPT 2010*, pages 148–165, 2010.

[SHZI02] Junji Shikata, Goichiro Hanaoka, Yuliang Zheng, and Hideki Imai. Security notions for unconditionally secure signature schemes. In *Advances in Cryptology – EUROCRYPT 2002*, pages 434–449, 2002.

[SNR04] K. Srinathan, Arvind Narayanan, and C. Pandu Rangan. Optimal perfectly secure message transmission. In *Advances in Cryptology – CRYPTO 2004*, pages 545–561, 2004.

[SS11] Colleen Swanson and Douglas R. Stinson. Unconditionally secure signature schemes revisited. In *Proceedings of the 5th International Conference on Information Theoretic Security ICITS*, pages 100–116, 2011.

[SZ16] Gabriele Spini and Gilles 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.

[Upf92] Eli 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.

[VY92] Vijay V. Vazirani and Mihalis Yannakakis. Suboptimal cuts: Their enumeration, weight and number (extended abstract). In *Proceedings of the 19th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 366–377, 1992.

[YWS10] Li-Pu Yeh, Biing-Feng Wang, and Hsin-Hao Su. Efficient algorithms for the problems of enumerating cuts by non-decreasing weights. *Algorithmica*, 56(3):297–312, 2010.

[Zuc97] David Zuckerman. Randomness-optimal oblivious sampling. *Random Struct. Algorithms*, 11(4):345–367, 1997.

A Preliminaries (Cont'd)

In Appendix A.1, we define the cryptographic primitives used in the paper, in Appendix A.2, we define the MPC model, and in Appendix A.3, we present the correlated-randomness functionalities that are used in the paper.

A.1 Cryptographic Primitives

A.1.1 Error-Correcting Secret Sharing

We present the definition of error-correcting secret sharing, also known in the literature as robust secret sharing.

Definition A.1. *A (t, n)* error-correcting secret-sharing scheme (ECSS) *over a message space $\mathcal{M}$ consists of a pair of algorithms $(\mathsf{Share}, \mathsf{Recon})$ satisfying the following properties:*

1. ***t-privacy:*** *For every $m \in \mathcal{M}$, and every subset $\mathcal{I} \subseteq [n]$ of size $|\mathcal{I}| \leq t$, the distribution of $\{s_i\}_{i\in\mathcal{I}}$ is independent of m, where $(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(m)$.*
2. ***Reconstruction from up to t erroneous shares:*** *For every $m \in \mathcal{M}$, every $\boldsymbol{s} = (s_1, \ldots, s_n)$, and every $\boldsymbol{s}' = (s'_1, \ldots, s'_n)$ such that $\Pr_{\boldsymbol{S}\leftarrow\mathsf{Share}(m)}[\boldsymbol{S} = \boldsymbol{s}] > 0$ and $|\{i \mid s_i = s'_i\}| \geq n - t$, it holds that $m = \mathsf{Recon}(\boldsymbol{s}')$ (except for a negligible probability).*

ECSS can be constructed information-theoretically, with a negligible positive error probability, when $t < n/2$ [RB89, CDF01, CFOR12].

A.1.2 Committee Election

A committee-election protocol is a protocol for electing a subset (committee) of n' parties out of a set of n parties. In this work we consider electing uniformly at random committees of size $n' = \omega(\log n)$. If the fraction of corrupted parties is constant at the original n-party set, then the fraction of corrupted parties in the committee is only slightly larger. This follows immediately by the analysis of Boyle et al. [BGT11, Lem. 2.6] of Feige' lightest-bin protocol [Fei99].

Lemma A.2 ([BGT11])**.** *For any $n' < n$ and $0 < \beta < 1$, Feige's lightest-bin protocol is a 1-round, n-party protocol for electing a committee $\mathcal{C}$, such that for any set of corrupted parties $\mathcal{I} \subseteq [n]$ of size $t = \beta n$, the following holds.*

1. *$|\mathcal{C}| \leq n'$.*
2. *For every constant $\epsilon > 0$, $\Pr[|\mathcal{C} \setminus \mathcal{I}| \leq (1 - \beta - \epsilon)n'] < \frac{n}{n'} e^{-\frac{\epsilon^2 n'}{2(1-\beta)}}$.*
3. *For every constant $\epsilon > 0$, $\Pr\left[\frac{|\mathcal{C}\cap\mathcal{I}|}{|\mathcal{C}|} \geq \beta + \epsilon\right] < \frac{n}{n'} e^{-\frac{\epsilon^2 n'}{2(1-\beta)}}$.*

The following corollary follows.

Corollary A.3. *Let $\mathcal{C} \subseteq [n]$ be a uniformly random subset of size $n' = \omega(\log n)$. Let $\mathcal{I} \subseteq [n]$ be a set of corrupted parties of size $t = \beta \cdot n$, for a constant $0 < \beta < 1$. Then, except for a negligible probability (in n), it holds that for an arbitrary small $\epsilon > 0$*

$$|\mathcal{C} \cap \mathcal{I}| \leq (\beta + \epsilon) \cdot n'.$$

A.1.3 Information-Theoretic Signatures

Parts of the following section are taken almost verbatim from [IOZ14].

$\mathcal{P}$-verifiable Information-Theoretic Signatures. We recall the definition and construction of information-theoretic signatures [SHZI02, SASM10] but slightly modify the terminology to what we consider to be more intuitive. The signature scheme (in particular the key-generation algorithm) needs to know the total number of verifiers or alternatively the list $\mathcal{P}$ of their identities. Furthermore, as usually for information-theoretic primitives, the key-length needs to be proportional to the number of times that the key is used. Therefore, the scheme is parameterized by two natural numbers ℓ_S and ℓ_V which will be upper bounds on the number of signatures that can be generated and verified, respectively, without violating the security.

A *$\mathcal{P}$-verifiable signature scheme* consists of a triple of randomized algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, where:

1. $\mathsf{Gen}(1^\kappa, n, \ell_S, \ell_V)$ outputs a pair $(\mathtt{sk}, \vec{\mathtt{vk}})$, where $\mathtt{sk} \in \{0,1\}^\kappa$ is a signing key, $\vec{\mathtt{vk}} = (\mathtt{vk}_1, \ldots, \mathtt{vk}_n) \in (\{0,1\}^\kappa)^n$ is a verification-key-vector consisting of (private) verification sub-keys, and $\ell_S, \ell_V \in \mathbb{N}$.

2. $\mathsf{Sign}(m, \mathtt{sk})$ on input a message m and the signing-key $\mathtt{sk}$ outputs a signature $\sigma \in \{0,1\}^{\mathrm{poly}(\kappa)}$.

3. $\mathsf{Verify}(m, \sigma, \mathtt{vk}_i)$ on input a message m, a signature σ and a verification sub-key $\mathtt{vk}_i$, outputs a decision bit $d \in \{0,1\}$.

Definition A.4. *A $\mathcal{P}$-verifiable signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is said to be* information-theoretically (ℓ_S, ℓ_V)-secure *if it satisfies the following properties:*

- *(completeness) A valid signature is accepted from any honest receiver:*

$$\Pr[\mathsf{Gen} \to (\mathtt{sk}, (\mathtt{vk}_1, \ldots, \mathtt{vk}_n)); \quad \text{for } i \in [n] : (\mathsf{Verify}(m, \mathsf{Sign}(m, \mathtt{sk}), \mathtt{vk}_i) = 1)] = 1.$$

- *Let $\mathcal{O}^S_{\vec{\mathtt{sk}}}$ denote a signing oracle (on input m, $\mathcal{O}^S_{\vec{\mathtt{sk}}}$ outputs $\sigma = \mathsf{Sign}(m, \mathtt{sk})$) and $\mathcal{O}^V_{\vec{\mathtt{vk}}}$ denote a verification oracle (on input (m, σ, i), $\mathcal{O}^V_{\vec{\mathtt{vk}}}$ outputs $\mathsf{Verify}(m, \sigma, \mathtt{vk}_i)$). Also, let $\mathcal{A}^{\mathcal{O}^S_{\vec{\mathtt{sk}}}, \mathcal{O}^V_{\vec{\mathtt{vk}}}}$ denote a computationally unbounded adversary that makes at most ℓ_S calls to $\mathcal{O}^S_{\vec{\mathtt{sk}}}$ and at most ℓ_V calls to $\mathcal{O}^V_{\vec{\mathtt{vk}}}$, and gets to see the verification keys indexed by some subset $\mathcal{I} \subsetneq [n]$. The following properties hold:*

  - *(unforgeability) $\mathcal{A}^{\mathcal{O}^S_{\vec{\mathtt{sk}}}, \mathcal{O}^V_{\vec{\mathtt{vk}}}}$ cannot generate a valid signature on message m' of his choice, other than the one he queries $\mathcal{O}^S_{\vec{\mathtt{sk}}}$ on (except with negligible probability). Formally,*

$$\Pr\left[\begin{array}{c} \mathsf{Gen} \to (\mathtt{sk}, \vec{\mathtt{vk}}); \text{ for some } \mathcal{I} \subsetneq [n] \text{ chosen by } \mathcal{A}^{\mathcal{O}^S_{\vec{\mathtt{sk}}}, \mathcal{O}^V_{\vec{\mathtt{vk}}}} : \\ \left(A^{\mathcal{O}^S_{\vec{\mathtt{sk}}}, \mathcal{O}^V_{\vec{\mathtt{vk}}}}(\vec{\mathtt{vk}}|_{\mathcal{I}}) \to (m, \sigma, j)\right) \quad \wedge \quad (m \text{ was not queried to } \mathcal{O}^S_{\vec{\mathtt{sk}}}) \quad \wedge \\ (j \in [n] \setminus \mathcal{I}) \quad \wedge \quad (\mathsf{Verify}(m, \sigma, \mathtt{vk}_j) = 1) \end{array}\right] = \mathsf{negl}(\kappa).$$

– *(consistency)*[22] $\mathcal{A}^{\mathcal{O}^S_{sk},\mathcal{O}^V_{\vec{vk}}}$ *cannot create a signature that is accepted by some (honest) party and rejected by some other even after seeing* ℓ_S *valid signatures and verifying* ℓ_V *signatures (except with negligible probability). Formally,*

$$\Pr\left[\begin{array}{c} \mathit{Gen} \to (\mathit{sk}, \vec{\mathit{vk}});\ \text{for some } \mathcal{I} \subsetneq [n] \text{ chosen by } \mathcal{A}^{\mathcal{O}^S_{sk},\mathcal{O}^V_{\vec{vk}}}(\mathit{sk}) : \\ A^{\mathcal{O}^S_{sk},\mathcal{O}^V_{\vec{vk}}}(\mathit{sk}, \vec{\mathit{vk}}|_{\mathcal{I}}) \to (m,\sigma) \\ \exists i,j \in [n] \setminus \mathcal{I} \text{ s.t. } \mathit{Verify}(m,\sigma,\mathit{vk}_i) \neq \mathit{Verify}(m,\sigma,\mathit{vk}_j) \end{array}\right] = \mathit{negl}(\kappa).$$

In [SHZI02, SS11] a signature scheme satisfying the above notion of security was constructed. These signatures have a deterministic signature generation algorithm Sign. In the following (Figure 14) we describe the construction from [SHZI02] (as described by [SS11] but for a single signer). We point out that the keys and signatures in the described scheme are elements of a sufficiently large finite field F (i.e., $|F| = O(2^{\mathrm{poly}(\kappa)})$); one can easily derive a scheme for strings of length $\ell = \mathrm{poly}(\kappa)$ by applying an appropriate encoding: e.g., map the i'th element of F to the i'th string (in the lexicographic order) and vice versa. We say that a value σ is *a valid signature* on message m (with respect to a given key setup $(sk, \vec{\mathtt{vk}})$), if for every honest P_i it holds that $\mathsf{Verify}(m, \sigma, \mathtt{vk}_i) = 1$.

Theorem A.5 ([SS11])**.** *Assuming* $|F| = \Omega(2^\kappa)$ *and* $\ell_S = poly(\kappa)$ *the above signature scheme (Figure 14) is an information-theoretically* $(\ell_S, poly(\kappa))$*-secure* $\mathcal{P}$-verifiable signature scheme.

A.1.4 Averaging Samplers

Samplers [Zuc97, Gol11] were used in distributed protocols as a technique for *universe reduction* [GVZ06, KLST11, BGH13]. Specifically, they allow to sample points of a given universe such that the probability of hitting any particular subset approximately matches its density.

Definition A.6 ([KLST11, BGH13])**.** *A function* $\mathsf{Samp}\colon X \to Y$ *is a* (θ,δ)*-sampler if for any set* $S \subseteq Y$*, at most a* δ *fraction of the inputs* $x \in X$ *satisfy*

$$\frac{|\mathsf{Samp}(x) \cap S|}{|S|} > \frac{|S|}{|Y|} + \theta.$$

The constructions of samplers in [KLST11, BGH13] provide the additional guarantee that the sampled subsets do not have "large" intersections. This is an important property when the sampler is used to select committees (quorums), so that no committee member ends up being overloaded. Specifically, let $H(x,i) = \mathsf{Samp}(x \cdot n + i)$ for $x \in X$ and $i \in [n]$, and denote by $H^{-1}(x,i)$ the set of nodes y such that $i \in H(x,y)$. We say that a node i is d-overloaded by H if for some constant a, there is exists $x \in X$ such that $|H^{-1}(x,i)| > a \cdot d$. Samplers that are not overloading can be constructed with the following parameters.

Lemma A.7 ([KLST11, BGH13])**.** *For every constant* c*, for* $\delta = |X|^{-1}$*, and any* $\theta > 0$*, there is a* (θ,δ)*-sampler* $H : X \times [n] \to [n]^d$ *with* $d = O\left(\frac{\log(1/\delta)}{\theta^2}\right)$ *such that for all* $x \in X$*, no* $i \in [n]$ *is* d*-overloaded.*

[22] This property is often referred to as transferability.

Key Generation: The algorithm for key generation $\mathsf{Gen}(1^\kappa, n, \ell_S)$ is as follows:

1. For $(j,k) \in \{0,\ldots,n-1\} \times \{0,\ldots,\ell_S\}$, choose $a_{ij} \in_R F$ uniformly at random and set the signing key to be (the description of) the multi-variate polynomial

$$\mathtt{sk} := f(y_1,\ldots,y_{n-1},x) = \sum_{k=0}^{\ell_S} a_{0,k}x^k + \sum_{j=1}^{n-1}\sum_{k=0}^{\ell_S} a_{j,k}y_j x^k.$$

2. For $i \in [n]$, choose vector $\vec{v}_i = (v_{i,1},\ldots,v_{i,n-1}) \in_R F^{n-1}$ uniformly at random and set the i'th verification key to be

$$\mathtt{vk}_i = (\vec{v}_i, f_{\vec{v}_i}(x)),$$

where $f_{\vec{v}_i}(x) = f(v_{i,1},\ldots,v_{i,n-1},x)$.

Signature Generation: The algorithm for signing a message $m \in F$, given the above signing key, is (a description of) the following polynomial

$$\mathsf{Sign}(m,\mathtt{sk}) := g(y_1,\ldots,y_{n-1}) := f(y_1,\ldots,y_{n-1},m)$$

Signature Verification: The algorithm for verifying a signature $\sigma = g(y_1,\ldots,y_n)$ on a given message m using the i'th verification key is

$$\mathsf{Verify}(m,\sigma,\mathtt{vk}_i) = \begin{cases} 1, & \text{if } g(\vec{v}_i) = f_{\vec{v}_i}(m) \\ 0, & \text{otherwise} \end{cases}$$

Figure 14: Construction of information-theoretic signatures [SS11]

A.2 Model Definition

We provide the basic definitions for secure multiparty computation according to the real/ideal paradigm, for further details see [Gol04] (which in turn follows [GL90, Bea91, MR91, Can00]). Informally, a protocol is secure according to the real/ideal paradigm, if whatever an adversary can do in the real execution of protocol, can be done also in an ideal computation, in which an uncorrupted trusted party assists the computation. We consider security with *guaranteed output delivery*, meaning that the ideal-model adversary cannot prematurely abort the ideal computation. For the sake of clarity, we focus in this section on the simpler case of *static* adversaries, that decide on the set of corrupted parties before the protocol begins. The case of *adaptive* adversaries, that can decide which party to corrupt based on information gathered during the course of the protocol, follows in similar lines, but is more technically involved. We refer the reader to [Can00] for a precise definition of adaptively secure MPC.

Definition A.8 (functionalities)**.** *An n-party functionality is a random process that maps vectors of n inputs to vectors of n outputs. Given an n-party functionality $f\colon (\{0,1\}^*)^n \to (\{0,1\}^*)^n$, let $f_i(\boldsymbol{x})$ denote its i'th output coordinate, i.e., $f_i(\boldsymbol{x}) = f(\boldsymbol{x})_i$.*

Real-model execution. An n-party protocol $\pi = (P_1,\ldots,P_n)$ is an n-tuple of probabilistic interactive Turing machines. The term *party* P_i refers to the i'th interactive Turing machine. Each party P_i starts with input $x_i \in \{0,1\}^*$ and random coins $r_i \in \{0,1\}^*$. An *adversary* $\mathcal{A}$ is another

probabilistic interactive Turing machine describing the behavior of the corrupted parties. It starts the execution with input that contains the identities of the corrupted parties, their private inputs, and an additional auxiliary input. The parties execute the protocol in a *synchronous* network. That is, the execution proceeds in rounds: each round consists of a *send phase* (where parties send their messages for this round) followed by a *receive phase* (where they receive messages from other parties). The adversary is assumed to be *rushing*, which means that it can see the messages the honest parties send in a round before determining the messages that the corrupted parties send in that round.

We consider the *point-to-point (communication) model*, where all parties are connected via a *fully connected point-to-point network*. We emphasize that although every party has the ability to send a message to every other party, and to receive a message from every other party, we will focus on protocols where each party will only communicate with a subset of the parties. We consider three models for the communication lines between the parties: In the *authenticated-channels* model, the communication lines are assumed to be ideally authenticated but not private (and thus the adversary cannot modify messages sent between two honest parties, but can read them). In the *secure-channels* model, the communication lines are assumed to be ideally private (and thus the adversary cannot read or modify messages sent between two honest parties, but he learns the *size* of the message that was sent on the channel). In the *hidden-channels* model, the communication lines are assumed to hide the very fact that a message has been sent on the channel (and thus the adversary is not aware that a message has been sent between two honest parties). We do not assume the existence of a *broadcast channel*, however, we will occasionally assume the availability of a trusted preprocessing phase, that is required for executing a broadcast protocol.

Throughout the execution of the protocol, all the honest parties follow the instructions of the prescribed protocol, whereas the corrupted parties receive their instructions from the adversary. The adversary is considered to be *malicious*, meaning that it can instruct the corrupted parties to deviate from the protocol in any arbitrary way. At the conclusion of the execution, the honest parties output their prescribed output from the protocol, the corrupted parties output nothing, and the adversary outputs an (arbitrary) function of its view of the computation (containing the views of the corrupted parties). The view of a party in a given execution of the protocol consists of its input, its random coins, and the messages it sees throughout this execution.

Definition A.9 (real-model execution)**.** *Let $\pi = (P_1, \ldots, P_n)$ be an n-party protocol and let $\mathcal{I} \subseteq [n]$ denote the set of indices of the parties corrupted by $\mathcal{A}$. The* joint execution of π under $(\mathcal{A}, \mathcal{I})$ in the real model*, on input vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, auxiliary input z, and security parameter κ, denoted $\mathrm{REAL}_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x}, \kappa)$, is defined as the output vector of $P_1, \ldots, P_n$ and $\mathcal{A}(z)$ resulting from the protocol interaction, where for every $i \in \mathcal{I}$, party P_i computes its messages according to $\mathcal{A}$, and for every $j \notin \mathcal{I}$, party P_j computes its messages according to π.*

Ideal-model execution. An ideal computation of an n-party functionality f on input $\boldsymbol{x} = (x_1, \ldots, x_n)$ for parties $(P_1, \ldots, P_n)$ in the presence of an ideal-model adversary $\mathcal{A}$ controlling the parties indexed by $\mathcal{I} \subseteq [n]$, proceeds via the following steps.

Sending inputs to trusted party: An honest party P_i sends its input x_i to the trusted party. The adversary may send to the trusted party arbitrary inputs for the corrupted parties. Let x'_i be the value actually sent as the input of party P_i.

Trusted party answers the parties: If x'_i is outside of the domain for P_i, for some index i, or if no input was sent for P_i, then the trusted party sets x'_i to be some predetermined default value. Next, the trusted party computes $(y_1, \ldots, y_n) = f(x'_1, \ldots, x'_n)$ and sends y_i to party P_i for every i.

Outputs: Honest parties always output the message received from the trusted party and the corrupted parties output nothing. The adversary $\mathcal{A}$ outputs an arbitrary function of the initial inputs $\{x_i\}_{i\in\mathcal{I}}$, the messages received by the corrupted parties from the trusted party $\{y_i\}_{i\in\mathcal{I}}$, and its auxiliary input.

Definition A.10 (ideal-model execution)**.** *Let $f\colon (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party functionality and let $\mathcal{I} \subseteq [n]$. The* joint execution of f under $(\mathcal{A}, I)$ in the ideal model*, on input vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, auxiliary input z to $\mathcal{A}$, and security parameter κ, denoted $\text{IDEAL}_{f,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x}, \kappa)$, is defined as the output vector of $P_1, \ldots, P_n$ and $\mathcal{A}(z)$ resulting from the above described ideal process.*

Security definition. Having defined the real and ideal models, we can now define security of protocols according to the real/ideal paradigm.

Definition A.11. *Let $f\colon (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party functionality, and let π be a probabilistic polynomial-time protocol computing f. The protocol π is a* protocol that t-securely computes f with computational security*, if for every probabilistic polynomial-time real-model adversary $\mathcal{A}$, there exists a probabilistic polynomial-time adversary $\mathcal{S}$ for the ideal model, such that for every $\mathcal{I} \subseteq [n]$ of size at most t, it holds that*

$$\left\{\text{REAL}_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x}, \kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\kappa\in\mathbb{N}} \stackrel{\text{c}}{\equiv} \left\{\text{IDEAL}_{f,\mathcal{I},\mathcal{S}(z)}(\boldsymbol{x}, \kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\kappa\in\mathbb{N}}.$$

The protocol π is a protocol that t-securely computes f with information-theoretic security*, if for every real-model adversary $\mathcal{A}$, there exists an adversary $\mathcal{S}$ for the ideal model, whose running time is polynomial in the running time of $\mathcal{A}$, such that for every $\mathcal{I} \subseteq [n]$ of size at most t, it holds that*

$$\left\{\text{REAL}_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x}, \kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\kappa\in\mathbb{N}} \stackrel{\text{s}}{\equiv} \left\{\text{IDEAL}_{f,\mathcal{I},\mathcal{S}(z)}(\boldsymbol{x}, \kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\kappa\in\mathbb{N}}.$$

The Hybrid Model. The hybrid model is a model that extends the real model with a trusted party that provides ideal computation for specific functionalities. The parties communicate with this trusted party in exactly the same way as in the ideal model described above.

Let f and g be n-party functionalities. Then, an execution of a protocol π computing g in the f-hybrid model, involves the parties sending normal messages to each other (as in the real model) and in addition, having access to a trusted party computing f. It is essential that the invocations of f are done sequentially, meaning that before an invocation of f begins, the preceding invocation of f must finish. In particular, there is at most one call to f per round, and no other messages are sent during any round in which f is called.

Let $\mathcal{A}$ be an adversary with auxiliary input z and let $\mathcal{I} \subseteq [n]$ be the set of corrupted parties. We denote by $\text{HYBRID}^{f}_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x}, \kappa)$ the random variable consisting of the view of the adversary and the output of the honest parties, following an execution of π with ideal calls to a trusted party computing f on input vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, auxiliary input z to $\mathcal{A}$, and security parameter κ.

In this work, we will employ the sequential composition theorem of Canetti [Can00].

Theorem A.12 ([Can00])**.** *Let f and g be n-party functionalities. Let ρ be a protocol that t-securely computes f, and let π be a protocol that t-securely computes g in the f-hybrid model. Then, protocol $\pi^{f\to\rho}$, that is obtained from π by replacing all ideal calls to the trusted party computing f with the protocol ρ, is a protocol that t-securely computes g in the real model.*

Extended functionalities and extended protocols. As mentioned above, in the sequential composition theorem (Theorem A.12) an n-party protocol π is considered, in which all n parties invoke the trusted party for computing an n-party functionality f. Next, the adjusted protocol $\pi^{f\to\rho}$, where all hybrid calls to f are replaced by an n-party protocol ρ for computing f, is proven secure. It is essential that the same set of n parties, defined by π, will run all executions of the sub-protocol ρ in order to claim security of $\pi^{f\to\rho}$. Looking ahead, in some of our constructions (in Section 4) we will use sub-protocols that are executed only by a subset of the parties. During the rounds in which the sub-protocol takes place, the remaining parties (that do not patriciate in the sub-protocol) should remain idle, i.e., not send any message and not receive any message. Toward this goal, we show how to extend functionality and protocols, that are defined for a subset of the n parties, into functionalities and protocols that are defined for the entire party-set, such that the parties outside of the original subset remain idle.

Let $m < n$. Given an m-party functionality f, an m-party protocol ρ that t-securely computes f, and a subset $\mathcal{P} = \{i_1, \ldots, i_m\} \subseteq [n]$ of size m, we define the "extended functionality" $\mathsf{extend}^{\mathcal{P}\hookrightarrow[n]}(f)$ as the n-party functionality in which the output vector $(y_1, \ldots, y_n)$ is defined as follows: for every $i \notin \mathcal{P}$ the output $y_i = \epsilon$ is defined to be the empty string, and for every $i \in \mathcal{P}$, the output is computed via $(y_{i_1}, \ldots, y_{i_m}) = f(x_{i_1}, \ldots, x_{i_m})$. In addition, we define the protocol $\mathsf{extend}^{\mathcal{P}\hookrightarrow[n]}(\rho)$ as the n-party protocol where every party P_i, with $i \in \mathcal{P}$, follows the code of ρ, and ignores messages from parties outside of $\mathcal{P}$, and every party P_i, with $i \notin \mathcal{P}$, doesn't send any message, ignores all incoming messages, and outputs ϵ. The following lemma is straightforward.

Lemma A.13. *If ρ is an m-party protocol that t-securely computes the m-party functionality f, then $\mathsf{extend}^{\mathcal{P}\hookrightarrow[n]}(\rho)$ is an n-party protocol that t-securely computes the n-party functionality $\mathsf{extend}^{\mathcal{P}\hookrightarrow[n]}(f)$.*

A.3 Correlated Randomness Functionalities

The positive results presented in Section 4 are defined in the a model where the parties receive correlated randomness generated by a trusted setup phase (formally, in the correlated-randomness hybrid model). The correlated randomness that we consider is *public-key infrastructure (PKI)*, which is "minimal" in a sense, and commonly used in MPC protocols. In the computational setting, we consider PKI that is based on digital signatures (which exist assuming one-way functions exist), and in the information-theoretic setting a PKI that relies on information-theoretic signatures (see Appendix A.1.3).

Public-key infrastructure. The PKI functionality f_{pki} (Figure 15) generates a pair of signing and verification keys for every party, and hands each party its signing key along with all verification keys. For simplicity, when we say that a protocol π is in the PKI model, we mean that π is defined in the f_{pki}-hybrid model.

The functionality f_{pki}

The n-party functionality f_{pki} is parametrized by a signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ and proceeds with parties $\mathcal{P}_1 = \{P_1, \ldots, P_n\}$ as follows.

1. For every $i \in [n]$ generate $(\mathtt{sk}_i, \mathtt{vk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$.
2. The output for party P_i is the signing key $\mathtt{sk}_i$ and the vector of verification keys $(\mathtt{vk}_1, \ldots, \mathtt{vk}_n)$.

Figure 15: The PKI functionality

Information-theoretic PKI. The (ℓ_S, ℓ_V)-IT-PKI functionality $f_{\mathsf{it\text{-}pki}}^{(\ell_S,\ell_V)}$ (Figure 16) generates n tuples of signing and verification keys, and hands each party its signing key along with all corresponding verification keys. For simplicity, when we say that a protocol π is in the (ℓ_S, ℓ_V)-IT-PKI model, we mean that π is defined in the $f_{\mathsf{it\text{-}pki}}^{(\ell_S,\ell_V)}$-hybrid model. By the IT-PKI model we mean (ℓ_S, ℓ_V)-IT-PKI model where ℓ_S and ℓ_V are polynomial in κ.

The functionality $f_{\mathsf{it\text{-}pki}}$

The n-party functionality $f_{\mathsf{it\text{-}pki}}^{(\ell_S,\ell_V)}$ is parametrized by an information-theoretically (ℓ_S, ℓ_V)-secure signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ and proceeds with parties $\mathcal{P}_1 = \{P_1, \ldots, P_n\}$ as follows.

1. For every $i \in [n]$ generate $(\mathtt{sk}_i, \vec{\mathtt{vk}}_i) \leftarrow \mathsf{Gen}(1^\kappa, n, \ell_S, \ell_V)$, where $\vec{\mathtt{vk}}_i = (\mathtt{vk}_1^i, \ldots, \mathtt{vk}_n^i)$.
2. The output for party P_i is the signing key $\mathtt{sk}_i$ and the vector of verification keys $(\mathtt{vk}_i^1, \ldots, \mathtt{vk}_i^n)$.

Figure 16: The information-theoretic PKI functionality

B MPC with Non-Expanding Communication Graph (Cont'd)

We now provide complementary material for Section 4.

B.1 Proof of Proposition 4.3

Proof. Let $\mathcal{A}$ be an adversary attacking the execution of π_n^{ne}, and let $\mathcal{I} \subseteq [n]$ be a subset of size at most $t = \beta n$. We construct the following adversary $\mathcal{S}$ for the ideal model computing f. On inputs $\{x_i\}_{i\in\mathcal{I}}$ and auxiliary input z, the simulator $\mathcal{S}$ starts by emulating $\mathcal{A}$ on these inputs. Initially, $\mathcal{S}$ generates a pair of signature keys $(\mathtt{sk}_i, \mathtt{vk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$ for every $i \in [n]$, and hands $\{\mathtt{sk}_i\}_{i\in\mathcal{I}}$ and $(\mathtt{vk}_1, \ldots, \mathtt{vk}_n)$ to $\mathcal{A}$. Next, $\mathcal{S}$ plays towards $\mathcal{A}$ the roles of the honest parties and the ideal functionalities $f_{\mathsf{elect\text{-}share}}$, $f_{\mathsf{recon\text{-}compute}}$, and $f_{\mathsf{out\text{-}dist}}$. For simplicity, assume that all input values are elements in $\{0,1\}^\kappa$.

To simulate Step 1, the simulator emulates $f_{\mathsf{elect\text{-}share}}^{(t',n')}$ towards $\mathcal{A}$ as follows. $\mathcal{S}$ receives from $\mathcal{A}$ input values $\{(x_i', \mathtt{sk}_i')\}_{i\in\mathcal{I}\cap[m]}$ (replace invalid inputs with default). Next, $\mathcal{S}$ samples uniformly at random two subsets $\mathcal{C}_1 = \{i_{(1,1)}, \ldots, i_{(1,n')}\} \subseteq [m]$ and $\mathcal{C}_2 = \{i_{(2,1)}, \ldots, i_{(2,n')}\} \subseteq [m]$, sets $\mathtt{sk}_i' = \mathtt{sk}_i$ for every $i \in [m] \setminus \mathcal{I}$, and signs the subsets as $\sigma_1 = \mathsf{Sign}_{\mathtt{sk}_1', \ldots, \mathtt{sk}_m'}(\mathcal{C}_1)$ and $\sigma_2 = \mathsf{Sign}_{\mathtt{sk}_1', \ldots, \mathtt{sk}_m'}(\mathcal{C}_2)$. Finally, $\mathcal{S}$ generates secret shares of zero for every $i \in [m]$ as $(s_i^1, \ldots, s_i^{n'}) \leftarrow \mathsf{Share}(0^\kappa)$, and hands

$\mathcal{A}$ the output $\mathcal{C}_1$ for every P_i with $i \in ([m] \cap \mathcal{I}) \setminus \mathcal{C}_1$, and $(\mathcal{C}_1, \sigma_1, \mathcal{C}_2, \sigma_2, \boldsymbol{s}_j)$, with $\boldsymbol{s}_j = (s_1^j, \ldots, s_m^j)$, for every $i = i_{(1,j)} \in \mathcal{C}_1 \cap \mathcal{I}$.

To simulate Step 2, the simulator sends $(\mathcal{C}_1, \sigma_1, \mathcal{C}_2, \sigma_2)$ on behalf of every honest party P_i with $i \in \mathcal{C}_1 \setminus \mathcal{I}$ to every corrupted party $P_{m+i_{(2,j)}}$ with $i_{(2,j)} \in \mathcal{C}_2$ and $m + i_{(2,j)} \in \mathcal{I}$, and receives such values from the adversary ($\mathcal{S}$ ignores any invalid messages from $\mathcal{A}$). In addition, for every $j \in [n']$ such that $i_{(1,j)} \notin \mathcal{I}$ and $m + i_{(2,j)} \in \mathcal{I}$, the simulator $\mathcal{S}$ sends to $\mathcal{A}$ on behalf of the honest party $P_{i_{(1,j)}}$ the vector $\boldsymbol{s}_j$, intended to $P_{m+i_{(2,j)}}$.

To simulate Step 3, the simulator emulates $f_{\mathsf{recon\text{-}compute}}^{\mathtt{vk}_1,\ldots,\mathtt{vk}_m}$ towards $\mathcal{A}$ as follows. $\mathcal{S}$ receives from $\mathcal{A}$ input values $\{(x'_{m+i}, z_{m+i})\}_{m+i \in \mathcal{I} \cap [m+1,2m]}$, where either $z_{m+i} = \epsilon$ or $z_{m+i} = (\mathcal{C}_{m+i}, \sigma_{m+i}, \boldsymbol{s}_{m+i})$ (replace invalid inputs with default). Next, $\mathcal{S}$ sends the values $\{x'_i\}_{i \in \mathcal{I}}$ to the trusted party (for $i \in \mathcal{I} \cap [m]$ the values x'_i were obtained in the simulation of Step 1, and for $i \in \mathcal{I} \cap [m+1, 2m]$ in the simulation of Step 3) and receives back the output value y. Finally, $\mathcal{S}$ hands y to $\mathcal{A}$ for every corrupted party P_{m+i}, with $m + i \in \mathcal{I} \cap [m+1, 2m]$.

To simulate Step 4, the simulator sends y on behalf of every honest party $P_{m+i_{(2,j)}}$ with $i_{(2,j)} \in \mathcal{C}_2$ and $m + i_{(2,j)} \notin \mathcal{I}$ to every corrupted party $P_{i_{(1,j)}}$ with $i_{(1,j)} \in \mathcal{C}_1 \cap \mathcal{I}$, and receives such values from the adversary.

To simulate Step 5, the simulator emulates the functionality $f_{\mathsf{out\text{-}dist}}^{\mathcal{C}_1}$ by receiving a value y_i from $\mathcal{A}$ for every $i \in \mathcal{I} \cap \mathcal{C}_1$, and sending y to every corrupted party P_i with $i \in \mathcal{I} \cap [m]$. Finally, $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs and halts.

We prove computational indistinguishability between the execution of the protocol π_n^{ne} running with adversary $\mathcal{A}$ and the ideal computation of f running with $\mathcal{S}$ via a series of hybrids experiments. The output of each experiment is the output of the honest parties and of the adversary.

The game $\mathrm{HYB}^1_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x}, \kappa)$**.** This game is defined to be the execution of the protocol π_n^{ne} in the $(f_{\mathsf{elect\text{-}share}}, f_{\mathsf{recon\text{-}compute}}, f_{\mathsf{out\text{-}dist}})$-hybrid model on inputs $\boldsymbol{x} \in (\{0,1\}^*)^n$ and security parameter κ with adversary $\mathcal{A}$ running on auxiliary information z and controlling parties in $\mathcal{I}$.

The game $\mathrm{HYB}^2_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x}, \kappa)$**.** In this game, we modify $\mathrm{HYB}^1_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x}, \kappa)$ as follows. Instead of verifying the signatures of the input values $\mathcal{C}_2$ and σ_2, the functionality $f_{\mathsf{recon\text{-}compute}}$ considers the subset $\mathcal{C}_2$ that was sampled by $f_{\mathsf{elect\text{-}share}}$.

Claim B.1. $\{\mathrm{HYB}^1_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x}, \kappa)\}_{\boldsymbol{x},z,\kappa} \stackrel{\mathrm{c}}{\equiv} \{\mathrm{HYB}^2_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x}, \kappa)\}_{\boldsymbol{x},z,\kappa}$.

Proof. The claim follows from two observations. First, the signed subset $\mathcal{C}_2$ that was computed by $f_{\mathsf{elect\text{-}share}}$ will be given as input to $f_{\mathsf{recon\text{-}compute}}$ by at least one party, as long as there exist honest parties in $\mathcal{C}_1$ and $\mathcal{C}_2$. This is guaranteed to occur except for a negligible probability following Corollary A.3.

Second, by the security of the signature scheme, the functionality $f_{\mathsf{recon\text{-}compute}}$ will not receive a second signed subset. Indeed, if two subsets $\mathcal{C}_2$ and $\mathcal{C}'_2$ with accepting signatures σ_2 and σ'_2 (resp.) are given to $f_{\mathsf{recon\text{-}compute}}$ with a noticeable probability, one can construct a forger to the signature scheme. □

The game $\mathrm{HYB}^3_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x}, \kappa)$**.** In this game, we modify $\mathrm{HYB}^2_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x}, \kappa)$ as follows. instead of computing the function f based on the inputs of $\mathcal{P}_2$ provided to $f_{\mathsf{recon\text{-}compute}}$ and the reconstructed input values of $\mathcal{P}_1$ based on the shares provided by the parties in $\mathcal{C}_2$, the computation of f is performed as follows. The input values $\{x'_i\}_{i \in \mathcal{I} \cap [m]}$ provided by the adversary to $f_{\mathsf{elect\text{-}share}}$ and the

input values $\{x'_i\}_{i\in\mathcal{I}\cap[m+1,2m]}$ provided by the adversary to $f_{\text{recon-compute}}$ are sent to an external party that computes $y = f(x'_1,\ldots,x'_n)$, where $x'_i = x_i$ for every $i \in [n]\setminus\mathcal{I}$.

Claim B.2. $\{\text{HYB}^2_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x},\kappa)\}_{\boldsymbol{x},z,\kappa} \stackrel{s}{\equiv} \{\text{HYB}^3_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x},\kappa)\}_{\boldsymbol{x},z,\kappa}$.

Proof. The claim will follow as long as $|(\mathcal{C}_1 \cup \{m+i \mid i\in\mathcal{C}_2\})\cap\mathcal{I}| \le t'$. In order to prove this, we will show separately that $|\mathcal{C}_1\cap\mathcal{I}| < t'/2$ and that $|\{m+i \mid i\in\mathcal{C}_2\}\cap\mathcal{I}| < t'/2$, except for a negligible probability.

Since $\beta < 1/4-\delta$, for some fixed $\delta>0$, and there are at most $\beta\cdot n$ corruptions, and $n=2m$ parties, it holds that $|\mathcal{I}| < \beta\cdot 2m < (1/2-2\delta)\cdot m$, therefore, $|[m]\cap\mathcal{I}| < (1/2-2\delta)\cdot m$, and similarly, $|[m+1,2m]\cap\mathcal{I}| < (1/2-2\delta)\cdot m$. Now, since $n' = \omega(\log(n)) = \omega(\log(m))$, it follows from Corollary A.3 that $|\mathcal{C}_1\cap\mathcal{I}| < (1/2-\delta)\cdot n'$ and $|\{m+i \mid i\in\mathcal{C}_2\}\cap\mathcal{I}| < (1/2-\delta)\cdot n'$, except for a negligible probability. □

The game $\text{HYB}^4_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x},\kappa)$**.** In this game, we modify $\text{HYB}^3_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x},\kappa)$ as follows. instead of computing shares of the input values $\{x_i\}_{i\in[m]}$, the functionality $f_{\text{elect-share}}$ computes shares of 0^κ.

Claim B.3. $\{\text{HYB}^4_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x},\kappa)\}_{\boldsymbol{x},z,\kappa} \stackrel{s}{\equiv} \{\text{HYB}^3_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x},\kappa)\}_{\boldsymbol{x},z,\kappa}$.

Proof. The claim follows from the privacy of the ECSS scheme and since $|\mathcal{C}_1\cap\mathcal{I}| \le t'$ and $|\{m+i \mid i\in\mathcal{C}_2\}\cap\mathcal{I}| \le t'$, except for a negligible probability. □

The proof now follows since $\text{HYB}^4_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x},\kappa)$ exactly describes the simulation done by $\mathcal{S}$, and in particular, does not depend on the input values of honest parties. This concludes the proof of Proposition 4.3. □

B.2 Proof of Lemma 4.10

Proof. For $m\in\mathbb{N}$ and $n=2m$, we construct the n-party protocol $\pi^{\text{a-ne}}_n$ (see Figure 17) in the $(f_{\text{a-elect-share}}, f_{\text{a-recon-compute}}, f_{\text{a-out-dist}})$-hybrid model. As in the proof of Lemma 4.2, the parameters for the protocol are $n' = \log^2(n)$ and $t' = (1/2-\delta)\cdot n'$. We start by proving in Proposition B.4 that the protocol $\pi^{\text{a-ne}}_n$ securely computes f_n. Next, in Proposition B.8 we prove that the communication graph of $\pi^{\text{a-ne}}$ is strongly not an expander. Finally, in Proposition B.9 we prove that by instantiating the functionalities $(f_{\text{a-elect-share}}, f_{\text{a-recon-compute}}, f_{\text{a-out-dist}})$ using low-locality protocols, the resulting protocol has low locality.

Proposition B.4. *For sufficiently large n, the protocol $\pi^{\text{a-ne}}_n$ securely computes the function f_n, tolerating adaptive, PPT βn corruptions, in the $(f_{\text{a-elect-share}}, f_{\text{a-recon-compute}}, f_{\text{a-out-dist}})$-hybrid model.*

Proof. Let $\mathcal{A}$ be an adversary attacking the execution of $\pi^{\text{a-ne}}_n$ and let $\mathcal{Z}$ be an environment. We construct an ideal-process adversary $\mathcal{S}$, interacting with the environment $\mathcal{Z}$, the ideal functionality f_n, and with ideal (dummy) parties $\tilde{P}_1,\ldots,\tilde{P}_n$. The simulator $\mathcal{S}$ constructs virtual parties $P_1,\ldots,P_n$, and runs the adversary $\mathcal{A}$. Note that the protocol $\pi^{\text{a-ne}}$ is deterministic and the only randomness arrives from the ideal functionalities. Therefore, upon a corruption request, the simulator is only required to provide the party's input, interface with the ideal functionalities, and possibly the output. Denote by $\mathcal{I}$ the set of corrupted parties (note that this set is dynamic: initially it is set to $\emptyset$ and it grows whenever an honest party gets corrupted).

Protocol $\pi_n^{\mathsf{a\text{-}ne}}$

- **Hybrid Model:** The protocol is defined in $(f_{\mathsf{a\text{-}elect\text{-}share}}, f_{\mathsf{a\text{-}recon\text{-}compute}}, f_{\mathsf{a\text{-}out\text{-}dist}})$-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 $f_{\mathsf{a\text{-}elect\text{-}share}}^{(t',n')}$, where every $P_i \in \mathcal{P}_1$ sends input $(x_i, \mathtt{sk}_i)$, and receives back either the empty output, or an output consisting of an index $i_{(2,j)} \in [m]$, a signature $\sigma_{1,j}$ of its own index (denoted $i_{1,j}$), a signature $\sigma_{2,j}$ of the index $m + i_{(2,j)}$, and a vector $\boldsymbol{s}_j = (s_1^j, \ldots, s_m^j)$.
2. Denote $\mathcal{C}_1 = \{i_{1,1}, \ldots, i_{1,n'}\}$. Every P_i with $i = i_{(1,j)} \in \mathcal{C}_1$ sends $(\sigma_{1,j}, \sigma_{2,j}, \boldsymbol{s}_j)$ to $P_{m+i_{(2,j)}}$.
   A party $P_{i_2} \in \mathcal{P}_2$ that receives a message $(\sigma_{1,j}, \sigma_{2,j}, \boldsymbol{s}_j)$ from $P_{i_1} \in \mathcal{P}_1$ will discard the message in the following cases:
   (a) If $\mathsf{Verify}_{\mathtt{vk}_1,\ldots,\mathtt{vk}_m}(i_1, \sigma_{1,j}) = 0$.
   (b) If $\mathsf{Verify}_{\mathtt{vk}_1,\ldots,\mathtt{vk}_m}(i_2, \sigma_{2,j}) = 0$.
3. Denote the set of parties that received a valid message in Step 2 by $\mathcal{C}_2$. The parties in $\mathcal{P}_2$ invoke $f_{\mathsf{a\text{-}recon\text{-}compute}}^{(\mathtt{vk}_1,\ldots,\mathtt{vk}_m)}$, where $P_{m+i} \in \mathcal{P}_2$ sends input (x_{m+i}, z_{m+i}) such that for $m+i \notin \mathcal{C}_2$, set $z_{m+i} = \epsilon$, and for $m + i = m + i_{(2,j)} \in \mathcal{C}_2$, set $z_{m+i} = (\sigma_{(2,j)}, \boldsymbol{s}_j)$. Every party in $\mathcal{P}_2$ receives the 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 $f_{\mathsf{a\text{-}out\text{-}dist}}^{(\mathtt{vk}_1,\ldots,\mathtt{vk}_m)}$, where party P_i, with $i \in \mathcal{C}_1$, has input $(\sigma_{(1,j)}, y)$, 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 17: Non-expanding MPC in the $(f_{\mathsf{a\text{-}elect\text{-}share}}, f_{\mathsf{a\text{-}recon\text{-}compute}}, f_{\mathsf{a\text{-}out\text{-}dist}})$-hybrid model

Simulating communication with the environment: In order to simulate the communication with $\mathcal{Z}$, every input value that $\mathcal{S}$ receives from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape. Likewise, every output value written by $\mathcal{A}$ on its output tape is copied to $\mathcal{S}$'s own output tape.

Simulating the PKI: The simulator $\mathcal{S}$ generates $(\mathtt{sk}_i, \mathtt{vk}_i) \leftarrow \mathsf{Gen}(1^\kappa)$ for every $i \in [n]$.

Simulating the protocol: To simulate Step 1, the simulator emulates $f_{\mathsf{a\text{-}elect\text{-}share}}^{(t',n')}$ towards $\mathcal{A}$ as follows. $\mathcal{S}$ receives from $\mathcal{A}$ input values $\{(x_i', \mathtt{sk}_i')\}_{i \in \mathcal{I} \cap [m]}$ (replace invalid inputs with default). Next, $\mathcal{S}$ samples uniformly at random two subsets $\mathcal{C}_1 = \{i_{(1,1)}, \ldots, i_{(1,n')}\} \subseteq [m]$ and $\mathcal{C}_2 = \{i_{(2,1)}, \ldots, i_{(2,n')}\} \subseteq [m]$, and sets $\mathtt{sk}_i' = \mathtt{sk}_i$ for every $i \in [m] \setminus \mathcal{I}$. For every $j \in [n']$, sign $\sigma_{(1,j)} = \mathsf{Sign}_{\mathtt{sk}_1,\ldots,\mathtt{sk}_m}(i_{(1,j)})$ and $\sigma_{(2,j)} = \mathsf{Sign}_{\mathtt{sk}_1,\ldots,\mathtt{sk}_m}(m + i_{(2,j)})$. Finally, generate secret shares of zero for every $i \in [m]$ as $(s_i^1, \ldots, s_i^{n'}) \leftarrow \mathsf{Share}(0^\kappa)$, and hand $\mathcal{A}$ the output $(i_{(2,j)}, \sigma_{i_{(1,j)}}, \sigma_{i_{(2,j)}}, \boldsymbol{s}_j)$, with $\boldsymbol{s}_j = (s_1^j \ldots, s_m^j)$, for every $i = i_{(1,j)} \in \mathcal{C}_1 \cap \mathcal{I}$.

To simulate Step 2, for every $j \in [n']$ such that $i_{(1,j)} \notin \mathcal{I}$ and $m + i_{(2,j)} \in \mathcal{I}$, the simulator $\mathcal{S}$ sends to $\mathcal{A}$ on behalf of the honest party $P_{i_{(1,j)}}$ the value $(\sigma_{i_{(1,j)}}, \sigma_{i_{(2,j)}}, \boldsymbol{s}_j)$, intended to $P_{m+i_{(2,j)}}$. In addition, $\mathcal{S}$ receives such values from the adversary ($\mathcal{S}$ ignores any invalid messages from $\mathcal{A}$).

To simulate Step 3, the simulator emulates $f_{\mathsf{a\text{-}recon\text{-}compute}}^{\mathtt{vk}_1,\ldots,\mathtt{vk}_m}$ towards $\mathcal{A}$ as follows. $\mathcal{S}$ receives from $\mathcal{A}$ input values $\{(x'_{m+i}, z_{m+i})\}_{m+i \in \mathcal{I} \cap [m+1,2m]}$, where either $z_{m+i} = \epsilon$ or $z_{m+i} = (\sigma_{i_{(2,j)}}, \boldsymbol{s}_j)$ (replace invalid inputs with default). Next, $\mathcal{S}$ sends the values $\{x'_i\}_{i \in \mathcal{I}}$ to the trusted party (for $i \in \mathcal{I} \cap [m]$ the values x'_i were obtained in the simulation of Step 1, and for $i \in \mathcal{I} \cap [m+1, 2m]$ in the simulation of Step 3) and receives back the output value y. Finally, $\mathcal{S}$ hands y to $\mathcal{A}$ for every corrupted party P_{m+i}, with $m + i \in \mathcal{I} \cap [m + 1, 2m]$.

To simulate Step 4, the simulator sends y on behalf of every honest party $P_{m+i_{(2,j)}}$ with $i_{(2,j)} \in \mathcal{C}_2$ and $m + i_{(2,j)} \notin \mathcal{I}$ to every corrupted party $P_{i_{(1,j)}}$ with $i_{(1,j)} \in \mathcal{C}_1 \cap \mathcal{I}$, and receives such values from the adversary.

To simulate Step 5, the simulator emulates the functionality $f_{\mathsf{a\text{-}out\text{-}dist}}^{(\mathtt{vk}_1,\ldots,\mathtt{vk}_m)}$ by receiving either a value (σ_i, y_i) or an empty string ϵ from $\mathcal{A}$ for every $i \in \mathcal{I} \cap [m]$, and sending y to every corrupted party P_i with $i \in \mathcal{I} \cap [m]$.

Simulating corruption requests by $\mathcal{A}$: Whenever the adversary $\mathcal{A}$ requests to corrupt an honest party P_i, the simulator $\mathcal{S}$ corrupts $\tilde{P}_i$, learn its input x_i and continues as follows to compute the internal state of P_i, based on the timing of the corruption request:

- **In Step 1, before calling $f_{\mathsf{a\text{-}elect\text{-}share}}$:** The simulator $\mathcal{S}$ sets the contents of P_i's input tape to be x_i. The secret signing key is set to be $\mathtt{sk}_i$ and the verification keys are set to be $\mathtt{vk}_1, \ldots, \mathtt{vk}_n$.
- **In Step 1, after calling $f_{\mathsf{a\text{-}elect\text{-}share}}$:** In addition to the above, if $P_i \in \mathcal{P}_1$, the simulator $\mathcal{S}$ sets the input of P_i to $f_{\mathsf{a\text{-}elect\text{-}share}}$ to be $(x_i, \mathtt{sk}_i)$. If $i = i_{(1,j)} \in \mathcal{C}_1$, set the output from $f_{\mathsf{a\text{-}elect\text{-}share}}$ to be $(i_{(2,j)}, \sigma_{i_{(1,j)}}, \sigma_{i_{(2,j)}}, \boldsymbol{s}_j)$ as computed in the simulation.
- **In Step 2:** In addition to the above, if $i = i_{(1,j)} \in \mathcal{C}_1$, the simulator sets the outgoing message of P_i to $P_{m+i_{(2,j)}}$ to be $(\sigma_{i_{(1,j)}}, \sigma_{i_{(2,j)}}, \boldsymbol{s}_j)$. If $i = m + i_{(2,j)}$ with $i_{(2,j)} \in \mathcal{C}_1$, the simulator set the incoming message of P_i from $P_{i_{(1,j)}}$ as follows: if $P_{i_{(1,j)}}$ is honest set it to be $(\sigma_{i_{(1,j)}}, \sigma_{i_{(2,j)}}, \boldsymbol{s}_j)$; otherwise, set it according to the values sent by $\mathcal{A}$.
- **In Step 3, before calling $f_{\mathsf{a\text{-}recon\text{-}compute}}$:** In addition to the above, if $P_i \in \mathcal{P}_2$, the simulator $\mathcal{S}$ sets the input of P_i to $f_{\mathsf{a\text{-}recon\text{-}compute}}$ to be (x_i, z_i), where if $i = m + i_{(2,j)}$ with $i_{(2,j)} \in \mathcal{C}_2$, set $z_i = (\sigma_{i_{(2,j)}}, \tilde{\boldsymbol{s}}_j)$ where $\tilde{\boldsymbol{s}}_j = (\tilde{s}_j^1, \ldots, \tilde{s}_j^{n'})$, such that if party $P_{i_{(1,k)}}$ was corrupted during Step 2 then $\tilde{s}_j^k$ is set according to the values sent by $\mathcal{A}$, and if $P_{i_{(1,k)}}$ was honest then set $\tilde{s}_j^k = s_j^k$. Otherwise, set $z_i = \epsilon$.
- **In Step 3, after calling $f_{\mathsf{a\text{-}recon\text{-}compute}}$:** In addition to the above, if $P_i \in \mathcal{P}_2$, the simulator $\mathcal{S}$ sets the output of P_i from $f_{\mathsf{a\text{-}recon\text{-}compute}}$ to be y.
- **In Step 4:** In addition to the above, if $i = m + i_{(2,j)}$ with $i_{(2,j)} \in \mathcal{C}_2$, the simulator sets the outgoing message of P_i to $P_{i_{(1,j)}}$ to be y. If $i = i_{(1,j)} \in \mathcal{C}_1$, the simulator sets the incoming message of P_i from $P_{m+i_{(2,j)}}$ as follows: if $P_{m+i_{(2,j)}}$ is honest set it to be y; otherwise, set it according to the values sent by $\mathcal{A}$.

- **In Step 5, before calling $f_{\text{a-out-dist}}$:** In addition to the above, if $i = i_{(1,j)} \in \mathcal{C}_1$, the simulator $\mathcal{S}$ sets the input of P_i to $f_{\text{a-out-dist}}$ to be $(\sigma_{i_{(1,j)}}, y)$.
- **In Step 5, after calling $f_{\text{a-out-dist}}$:** In addition to the above, if $i = [m]$, the simulator $\mathcal{S}$ sets the output of P_i from $f_{\text{a-out-dist}}$ to be y.

Next, $\mathcal{S}$ sends the internal state of P_i to $\mathcal{A}$.

Simulating post-execution corruption requests by $\mathcal{Z}$: Whenever the environment $\mathcal{Z}$ requests to corrupt an honest party P_i in the post-execution corruption phase, the simulator $\mathcal{S}$ proceeds to compute the internal state of P_i as in a corruption request from $\mathcal{A}$ *in Step 5, after calling* $f_{\text{a-out-dist}}$, and sends it to $\mathcal{Z}$.

Proving security. We prove computational indistinguishability between the execution of the protocol $\pi_n^{\text{a-ne}}$ running with adversary $\mathcal{A}$ and the ideal computation of f_n running with $\mathcal{S}$ via a series of hybrids experiments. The output of each experiment is the output of the honest parties and of the adversary.

The game $\text{HYB}^1_{\pi,\mathcal{A},\mathcal{Z}}(\boldsymbol{x}, z, \kappa)$. This game is defined to be the execution of the protocol $\pi_n^{\text{a-ne}}$ in the $(f_{\text{a-elect-share}}, f_{\text{a-recon-compute}}, f_{\text{a-out-dist}})$-hybrid model on inputs $\boldsymbol{x} \in (\{0,1\}^*)^n$ and security parameter κ with adversary $\mathcal{A}$ and environment $\mathcal{Z}$ running on auxiliary information z.

The game $\text{HYB}^2_{\pi,\mathcal{A},\mathcal{Z}}(\boldsymbol{x}, z, \kappa)$. In this game, we modify $\text{HYB}^1_{\pi,\mathcal{A},\mathcal{Z}}(\boldsymbol{x}, z, \kappa)$ as follows. Instead of verifying the signatures of the index of every party that provides an additional input, the functionality $f_{\text{a-recon-compute}}$ takes the shares $\boldsymbol{s}_j$ from parties $P_{m+i_{(2,j)}}$, with the $i_{(2,j)} \in \mathcal{C}_2$ that was sampled by $f_{\text{a-elect-share}}$.

Claim B.5. $\{\text{HYB}^1_{\pi,\mathcal{A},\mathcal{Z}}(\boldsymbol{x}, z, \kappa)\}_{\boldsymbol{x},z,\kappa} \stackrel{c}{\equiv} \{\text{HYB}^2_{\pi,\mathcal{A},\mathcal{Z}}(\boldsymbol{x}, z, \kappa)\}_{\boldsymbol{x},z,\kappa}$.

Proof. The claim follows from the following observations. First, For a fixed $j \in [n']$, if both parties $P_{i_{(1,j)}}$ and $P_{m+i_{(2,j)}}$ are honest, then $P_{m+i_{(2,j)}}$ will provide $f_{\text{a-recon-compute}}$ with the correct signature of its index.

Second, with overwhelming probability, for at least $\lceil n'/2 \rceil$ of the j's it holds that both $P_{i_{(1,j)}}$ and $P_{m+i_{(2,j)}}$ are honest. This follows from the strong honest-majority assumption. In more detail, since the communication between honest parties is hidden from the adversary, he can identify that an honest party P_i is in the committee $\mathcal{C}_1$, i.e., $i = i_{(1,j)}$, only if the corresponding party $P_{m+i_{(2,j)}}$ in $\mathcal{C}_2$ is corrupted and receives a message from $P_{i_{(1,j)}}$ (and vice versa). Assume towards a contradiction that for $\lceil n'/2 \rceil$ of the j's at least one of $P_{i_{(1,j)}}$ or $P_{m+i_{(2,j)}}$ is corrupted. This means that the fraction of corrupted parties in one of the committees $\mathcal{C}_1$ and $\mathcal{C}_2$ is $\beta' \geq 1/4$. However, since by assumption there are at most $(1/8 - \delta) \cdot n = (1/4 - 2\delta) \cdot m$ corrupted parties at any point during the protocol's execution (and after its completion), it holds that in $\mathcal{P}_1$ and in $\mathcal{P}_2$ the fraction of corrupted parties is at most $(1/4 - 2\delta)$. Now, since $n' = \omega(\log n)$ it follows from Corollary A.3 with overwhelming probability that if all the corruption took place at the onset of the protocol, then the fraction of corrupted parties in the committees is at most $(1/4 - \delta)$ (by setting $\epsilon = \delta$).

It is now remains to show that other than identifying "matching parties" in the committees (i.e., the pair of parties $P_{i_{(1,j)}}$ and $P_{m+i_{(2,j)}}$) in Steps 2 and 4, the adversary does not gain any advantage in increasing the fraction of corrupted parties in the committees by dynamically corrupting parties.

This follows since the communication is hidden from the adversary, and its view in the protocol (except for Steps 2 and 4) is independent of the committees. Therefore, we derive a contradiction.

Finally, by the security of the signature scheme, the functionality $f_{\text{a-recon-compute}}$ will not receive input with signed index from parties outside of $\mathcal{C}_2$. □

The game $\text{HYB}^3_{\pi,\mathcal{A},\mathcal{Z}}(\boldsymbol{x}, z, \kappa)$**.** In this game, we modify $\text{HYB}^2_{\pi,\mathcal{A},\mathcal{Z}}(\boldsymbol{x}, z, \kappa)$ as follows. Instead of computing the function f using the inputs of $\mathcal{P}_2$ as provided to $f_{\text{a-recon-compute}}$, and the input values of $\mathcal{P}_1$ as reconstructed from the shares provided by the parties in $\mathcal{C}_2$, the computation of f is performed as follows. Let $\mathcal{I}$ be the set of corrupted parties in Step 3 when calling $f_{\text{a-recon-compute}}$. The input values $\{x'_i\}_{i\in\mathcal{I}\cap[m]}$ provided by the adversary to $f_{\text{a-elect-share}}$ and the input values $\{x'_i\}_{i\in\mathcal{I}\cap[m+1,2m]}$ provided by the adversary to $f_{\text{a-recon-compute}}$ are sent to an external party that computes $y = f(x'_1, \ldots, x'_n)$, where $x'_i = x_i$ for every $i \in [n] \setminus \mathcal{I}$.

Claim B.6. $\{\text{HYB}^2_{\pi,\mathcal{A},\mathcal{Z}}(\boldsymbol{x}, z, \kappa)\}_{\boldsymbol{x},z,\kappa} \overset{\text{s}}{\equiv} \{\text{HYB}^3_{\pi,\mathcal{A},\mathcal{Z}}(\boldsymbol{x}, z, \kappa)\}_{\boldsymbol{x},z,\kappa}$.

Proof. The claim will follow as long as $|(\mathcal{C}_1 \cup \{m+i \mid i \in \mathcal{C}_2\}) \cap \mathcal{I}| \leq t'$. This follows from the proof of Claim B.5. □

The game $\text{HYB}^4_{\pi,\mathcal{A},\mathcal{Z}}(\boldsymbol{x}, z, \kappa)$**.** In this game, we modify $\text{HYB}^3_{\pi,\mathcal{A},\mathcal{Z}}(\boldsymbol{x}, z, \kappa)$ as follows. instead of computing shares of the input values $\{x_i\}_{i\in[m]}$, the functionality $f_{\text{a-elect-share}}$ computes shares of 0^κ.

Claim B.7. $\{\text{HYB}^4_{\pi,\mathcal{A},\mathcal{Z}}(\boldsymbol{x}, z, \kappa)\}_{\boldsymbol{x},z,\kappa} \overset{\text{s}}{\equiv} \{\text{HYB}^3_{\pi,\mathcal{A},\mathcal{Z}}(\boldsymbol{x}, z, \kappa)\}_{\boldsymbol{x},z,\kappa}$.

Proof. The claim follows from the privacy of the ECSS scheme and since $|\mathcal{C}_1 \cap \mathcal{I}| \leq t'$ and $|\{m+i \mid i \in \mathcal{C}_2\} \cap \mathcal{I}| \leq t'$ at any point during the protocol, except for a negligible probability. □

The proof now follows since $\text{HYB}^4_{\pi,\mathcal{I},\mathcal{A}(z)}(\boldsymbol{x}, \kappa)$ exactly describes the simulation done by $\mathcal{S}$, and in particular, does not depend on the input values of honest parties. This concludes the proof of Proposition B.4. □

Proposition B.8. *The communication graph of* π^{ne} *is strongly not an expander.*

Proof. The proof follows in a similar way as the proof of Proposition 4.4. □

Proposition B.9. *Let* ρ_1, ρ_2, ρ_3*, and* $\pi^{f_i \to \rho_i}$ *be the protocols defined in Lemma 4.10, 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* $\ell = 2 \cdot \ell_\rho + 1$*.*

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 exactly one party 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 exactly one party in $\mathcal{C}_1$. It follows that maximal number of parties that a party communicates with during the protocol is $2 \cdot \ell_\rho + 1$. □

This concludes the proof of Lemma 4.10. □

C Expansion is Necessary for Correct Computation (Cont'd)

We now provide complementary material for Section 5.

C.1 Proof of the Graph-Theoretic Theorem (Theorem 5.6)

In this section, we prove Theorem 5.6, stating that an (α, d)-partition exists and can be efficiently computed when the minimal degree of a vertex in the graph is sufficiently large.

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 an $(\alpha(n), n/c)$-partition of G_n, denoted Γ; it holds that $|\Gamma| \leq c$.*
2. *An $(\alpha(n), n/c)$-partition Γ of G_n can be found in (deterministic) polynomial time.*

In Lemma C.1, we prove the existence of such a partition; and in Lemma C.5, we present a deterministic polynomial-time algorithm for computing it. Recall that given a graph G with n vertices and a subset $S \subseteq [n]$, we denote by $\mathsf{edges}(S) = \mathsf{edges}(S, \bar{S})$ the set of edges from S to its complement.

Lemma C.1. *Consider the setting of Theorem 5.6. Then, there exists an $(\alpha(n), n/c)$-partition Γ of G_n, and it holds that $|\Gamma| \leq c$. Furthermore, the number of $\alpha(n)$-cuts in G_n is at most 2^{c-1}.*

Proof. Let $n \in \mathbb{N}$ and let

$$\left\{\{S_1, \bar{S}_1\}, \ldots, \{S_\ell, \bar{S}_\ell\}\right\}$$

be the set of all $\alpha(n)$-cuts in G_n, i.e., for every $i \in [\ell]$ it holds that

$$\left|\mathsf{edges}_{G_n}(S_i)\right| \leq \alpha(n). \tag{8}$$

We proceed by defining the following family of subsets

$$\Gamma = \left\{ \bigcap_{i=1}^{\ell} S_i^{b_i} : (b_1, b_2, \ldots, b_\ell) \in \{0,1\}^\ell \right\} \setminus \{\emptyset\},$$

where for every $i \in [\ell]$ and $b \in \{0,1\}$, the set S_i^b is defined as

$$S_i^b := \begin{cases} S_i & \text{if } b = 0, \\ \bar{S}_i & \text{if } b = 1. \end{cases}$$

We will show that for sufficiently large n, the set Γ is an $(\alpha(n), n/c)$-partition of G_n and that $|\Gamma| \leq c$. We start by proving two useful claims.

Claim C.2. *For every $S \subseteq [n]$, if $1 \leq |S| \leq \frac{n}{c} - 1$ then $|\mathsf{edges}_{G_n}(S)| \geq \frac{n}{c} - 1$.*

Proof. Consider an arbitrary set $S \subseteq [n]$. The claim follows from the following set of inequalities:

$$\begin{aligned}
|\mathsf{edges}_{G_n}(S)| &= (\text{total degree of the vertices in } S) \\
&\qquad - 2 \cdot (\text{total number of edges whose both vertices are inside } S) \\
&\geq |S| \cdot \left(\frac{n}{c} - 1\right) - 2 \cdot \binom{|S|}{2} \\
&= |S| \cdot \left(\frac{n}{c} - 1\right) - |S| \cdot (|S| - 1) \\
&= |S| \cdot \left(\frac{n}{c} - |S|\right) \\
&\overset{(*)}{\geq} \frac{n}{c} - 1.
\end{aligned}$$

The last inequality $(*)$ follows since for a constant a and $1 \leq x \leq a-1$, if $f(x) = x(a-x)$ then $a - 1 \leq f(x) \leq \frac{a^2}{4}$. In our case $1 \leq |S| \leq \frac{n}{c} - 1$; therefore

$$\frac{n}{c} - 1 \leq |S| \cdot \left(\frac{n}{c} - |S|\right) \leq \frac{n^2}{4c^2}.$$

This concludes the proof of Claim C.2. □

Claim C.3. *For every $(b_1, b_2, \ldots, b_\ell) \in \{0,1\}^\ell$, either $\bigcap_{i=1}^{\ell} S_i^{b_i} = \emptyset$ or $|\bigcap_{i=1}^{\ell} S_i^{b_i}| \geq n/c$.*

Proof. Suppose, to the contrary, that there exists an ℓ-bit vector $(b_1, b_2, \ldots, b_\ell) \in \{0,1\}^\ell$ such that $1 \leq |\bigcap_{i=1}^{\ell} S_i^{b_i}| \leq \frac{n}{c} - 1$. Let us consider the following nested sequence of sets

$$A_\ell \subseteq \ldots \subseteq A_2 \subseteq A_1,$$

where for every $j \in [\ell]$ the set A_j is defined as $A_j := \bigcap_{i=1}^{j} S_i^{b_i}$. Since $A_1 = S_1^{b_1}$ and S_1 satisfies Equation (8), it holds that

$$|\mathsf{edges}_{G_n}(A_1)| \leq \alpha(n). \tag{9}$$

Also, since $A_\ell = \bigcap_{i=1}^{\ell} S_i^{b_i}$ and $|\bigcap_{i=1}^{\ell} S_i^{b_i}| \leq \frac{n}{c} - 1$ (by assumption), it follows from Claim C.2 that

$$|\mathsf{edges}_{G_n}(A_\ell)| \geq \frac{n}{c} - 1. \tag{10}$$

The following two cases can occur:

Case 1. For every $j \in [\ell - 1]$, either $A_j \setminus A_{j+1} = \emptyset$ or $|A_j \setminus A_{j+1}| \geq n/c$: Consider the set $\mathcal{I} = \{i \in [\ell - 1] : A_i \setminus A_{i+1} \neq \emptyset\}$. It follows from the assumption that for every $i \in \mathcal{I}$, $|A_i \setminus A_{i+1}| \geq n/c$. Since these sets are disjoint and A_ℓ is non-empty, we have $|\mathcal{I}| \leq c - 1$. Note that there are at least $n/c - 1$ edges which are coming out of A_ℓ. Since $|\mathsf{edges}(A_1)| \leq \alpha(n)$, at least $n/c - 1 - \alpha(n)$ of these edges (whose one end-point is inside A_ℓ) must have their other end-point inside $A_1 \setminus A_\ell$. Observe that $A_1 \setminus A_\ell = \bigcup_{i=1}^{\mathcal{I}} A_i \setminus A_{i+1}$, which implies $|\mathsf{edges}(A_\ell, A_k \setminus A_{k+1})| \geq \frac{(n/c)-1-\alpha(n)}{c-1}$ for some $k \in \mathcal{I}$. Since $A_\ell \subseteq S_{k+1}^{b_{k+1}}$, which is disjoint from $A_k \setminus A_{k+1}$, we have $|\mathsf{edges}(S_{k+1}^{b_{k+1}})| \geq \frac{(n/c)-1-\alpha(n)}{c-1}$. Now, since $\alpha(n)$ is a sub-linear function in n, for sufficiently large n we have that $|\mathsf{edges}(S_{k+1}^{b+1})| > \alpha(n)$, which is a contradiction to Equation (8).

Case 2. $0 < |A_j \setminus A_{j+1}| \leq n/c - 1$ for some $j \in [\ell - 1]$: As in the previous case, consider the set $\mathcal{I} = \{i \in [\ell - 1] : A_i \setminus A_{i+1} \neq \emptyset\}$. Let $k \in \mathcal{I}$ be the first index such that $0 < |A_k \setminus A_{k+1}| \leq n/c - 1$. Define another set $\mathcal{I}' = \{i \in \mathcal{I} : i \leq k\}$. Since $k \in \mathcal{I}$ is the first index such that $0 < |A_k \setminus A_{k+1}| \leq n/c - 1$, the number of i's such that $i \in \mathcal{I}$ and $i \leq k$ must be less than c, which implies that $|\mathcal{I}'| \leq c - 1$. Note that A_k and $A_k \setminus A_{k+1}$ can be written as $A_k = \bigcap_{i \in \mathcal{I}'} S_i^{b_i}$ and $A_k \setminus A_{k+1} = \bigcap_{i \in \mathcal{I}'} S_i^{b_i} \bigcap S_{k+1}^{b_{k+1} \oplus 1}$, respectively. Let $s := |\mathcal{I}'|$. For simplicity of notation, let us renumber these cuts from 1 to $s+1$, i.e., let $\{S_1, S_2, \ldots, S_{s+1}\} := \{S_i : i \in \mathcal{I}'\} \bigcup S_{k+1}$. With this notation, there exists some $(\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_{s+1}) \in \{0,1\}^{s+1}$, such that $A_k \setminus A_{k+1} = \bigcap_{i=1}^{s+1} S_i^{\hat{b}_i}$ Since $0 < |A_k \setminus A_{k+1}| \leq n/c - 1$, we have from Claim C.2 that $|\mathsf{edges}(\bigcap_{i=1}^{s+1} S_i^{\hat{b}_i})| \geq n/c - 1$.

Note that for any edge $(u, v) \in \mathsf{edges}(\bigcap_{i=1}^{s+1} S_i^{\hat{b}_i})$, it holds that $v \in \bigcap_{i=1}^{s+1} S_i^{e_i}$ for some $(e_1, e_2, \ldots, e_{s+1}) \in \{0,1\}^{s+1}$. Since $|\mathsf{edges}(\bigcap_{i=1}^{s+1} S_i^{\hat{b}_i})| \geq n/c - 1$, there exists $(\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_{s+1}) \in \{0,1\}^{s+1}$ such that $|\mathsf{edges}(\bigcap_{i=1}^{s+1} S_i^{\hat{b}_i}, \bigcap_{i=1}^{s+1} S_i^{\hat{e}_i})| \geq \frac{(n/c)-1}{2^{s+1}}$. Since $s + 1 = |\mathcal{I}'| + 1 \leq c$, we have

$$\left| \mathsf{edges}\left(\bigcap_{i=1}^{s+1} S_i^{\hat{b}_i}, \bigcap_{i=1}^{s+1} S_i^{\hat{e}_i} \right) \right| \geq \frac{(n/c) - 1}{2^c}. \tag{11}$$

Since $(\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_{s+1}) \neq (\hat{e}_1, \hat{e}_2, \ldots, \hat{e}_{s+1})$, there exists $l \in [s+1]$ such that $\hat{b}_l \neq \hat{e}_l$. This, together with Equation (11), implies that $|\mathsf{edges}(S_l)| \geq \frac{(n/c)-1}{2^c}$. Since c is a constant, it holds $\frac{(n/c)-1}{2^c} > \alpha(n)$ for sufficiently large n, implying that $|\mathsf{edges}(S_l)| > \alpha(n)$, which is a contradiction to Equation (8). This concludes the proof of Claim C.3. □

Now we show that Γ is an $(\alpha(n), n/c)$-partition of G_n. First observe that the union of all the sets in Γ is indeed $[n]$. By Claim C.3 for every $S \in \Gamma$ it holds that $|S| \geq n/c$. Furthermore, for every $\alpha(n)$-cut $\{S_j, \bar{S}_j\}$ it holds that S_j and $\bar{S}_j$ can be represented as a union of some sets from Γ, more precisely

$$S_j = \bigcup_{(\vec{b} \in \{0,1\}^\ell \,:\, b_j = 0)} \bigcap_{i=1}^{\ell} S_i^{b_i} \quad \text{and} \quad \bar{S}_j = \bigcup_{(\vec{b} \in \{0,1\}^\ell \,:\, b_j = 1)} \bigcap_{i=1}^{\ell} S_i^{b_i}.$$

In addition, it is easy to see that the sets in Γ are pairwise disjoint, and since each is of size at least n/c it holds that $|\Gamma| \leq c$.

To show that distinct sets from Γ have at most $\alpha(n)$ crossing edges, consider two different sets $U_i, U_j \in \Gamma$. There exist distinct binary vectors $\vec{b}, \vec{e} \in \{0,1\}^\ell$ such that $U_i = \bigcap_{k=1}^{\ell} S_k^{b_k}$ and $U_j = \bigcap_{k=1}^{\ell} S_k^{e_k}$. Let $\hat{k} \in [\ell]$ be an index such that $b_{\hat{k}} \neq e_{\hat{k}}$. Since (1) $U_i \subseteq S_{\hat{k}}^{b_{\hat{k}}}$, (2) $U_j \subseteq S_{\hat{k}}^{e_{\hat{k}}}$, (3) $S_{\hat{k}}^{b_{\hat{k}}} \cap S_{\hat{k}}^{e_{\hat{k}}} = \emptyset$, and (4) $|\mathsf{edges}(S_{\hat{k}}^{b_{\hat{k}}}, S_{\hat{k}}^{e_{\hat{k}}})| \leq \alpha(n)$, it holds that $|\mathsf{edges}(U_i, U_j)| \leq \alpha(n)$.

Finally, since $|\Gamma| \leq c$ and for every $\alpha(n)$-cut $\{S_j, \bar{S}_j\}$ it holds that S_j and $\bar{S}_j$ can be represented as a union of some sets from Γ, we have that $\ell \leq 2^c - 1$ (which is the total number of nonempty subsets of Γ). However, a cut is defined by two such subsets; therefore it holds that the total number of $\alpha(n)$-cuts is at most 2^{c-1}.

This completes the proof of Lemma C.1. □

Lemma C.1 proved existence of a partition. In Lemma C.5, we show how to efficiently find such a partition. A core element of our algorithm is the algorithm for enumerating all cuts of a weighted graph due to Vazirani and Yannakakis [VY92] that runs in polynomial time with $\tilde{O}(n^2 m)$ delay

between two successive outputs (i.e., once the i'th cut was found, this is the running time needed to produce the $(i+1)$'th cut). Yeh et al. [YWS10] reduced the delays to $\tilde{O}(nm)$.

Theorem C.4 ([VY92, YWS10])**.** *Let $G = (V, E)$ be a weighted undirected graph on n vertices and m edges. There exists a deterministic polynomial-time algorithm to enumerate all the cuts of G by non-decreasing weights with $\tilde{O}(nm)$ delay between two successive outputs.*

Lemma C.5. *Consider the setting of Theorem 5.6. Then, for sufficiently large n, an $(\alpha(n), n/c)$-partition Γ of G_n can be found in polynomial time (precisely in $\tilde{O}(n^3)$ time).*

Proof. We start by assigning weight 1 to every edge of our graph G_n and run the algorithm from Theorem C.4 on this weighted graph. We enumerate all cuts by non-decreasing weights until we hit a cut whose weight is more than $\alpha(n)$. Since the total number of $\alpha(n)$-cuts is at most 2^{c-1}, we stop this algorithm after it has enumerated at most 2^{c-1} cuts.

Let $\{S_1, \bar{S}_1\}, \ldots, \{S_\ell, \bar{S}_\ell\}$ denote all the $\alpha(n)$-cuts of G_n. We define the partition Γ as above

$$\Gamma = \left\{ \bigcap_{i=1}^{\ell} S_i^{b_i} : (b_1, b_2, \ldots, b_\ell) \in \{0,1\}^\ell \right\} \setminus \{\emptyset\}.$$

Following the proof of Lemma C.1, the partition Γ is an $(\alpha(n), n/c)$-partition of G_n.

Running time analysis. Since c is a constant and total number of edges in our graph is $O(n^2)$, we can generate all the $\alpha(n)$-cuts in $\tilde{O}(n^3)$ time using the algorithm of Yeh et al. [YWS10]. Once we have found all the $\alpha(n)$-cuts, generating the $(\alpha(n), n/c)$-partition takes $O(n)$ time: since the total number of cuts ℓ is constant, computing $\bigcap_{i=1}^{\ell} S_i^{b_i}$ for any fixed vector $(b_1, b_2, \ldots, b_\ell) \in \{0,1\}^\ell$ takes $O(n)$ time. We are computing $2^\ell \leq 2^{2^{c-1}}$ such intersections corresponding to the 2^ℓ vectors. Because c is a constant, the total time is still $O(n)$. Hence, the total time required by our procedure for finding an $(\alpha(n), n/c)$-partition is $\tilde{O}(n^3)$. This completes the proof of Lemma C.5. □