Is Information-Theoretic Topology-Hiding Computation Possible?

Marshall Ball[*] Elette Boyle[†] Ran Cohen[‡] Tal Malkin[§] Tal Moran[¶]

September 24, 2019

Abstract

Topology-hiding computation (THC) is a form of multi-party computation over an incomplete communication graph that maintains the privacy of the underlying graph topology. Existing THC protocols consider an adversary that may corrupt an arbitrary number of parties, and rely on cryptographic assumptions such as DDH.

In this paper we address the question of whether *information-theoretic* THC can be achieved by taking advantage of an *honest majority*. In contrast to the standard MPC setting, this problem has remained open in the topology-hiding realm, even for simple "privacy-free" functions like broadcast, and even when considering only semi-honest corruptions.

We uncover a rich landscape of both positive and negative answers to the above question, showing that what types of graphs are used and how they are selected is an important factor in determining the feasibility of hiding topology information-theoretically. In particular, our results include the following.

- We show that topology-hiding broadcast (THB) on a *line* with four nodes, secure against a single semi-honest corruption, implies *key agreement*. This result extends to broader classes of graphs, e.g., THB on a *cycle* with two semi-honest corruptions.
- On the other hand, we provide the first feasibility result for information-theoretic THC: for the class of *cycle* graphs, with a single semi-honest corruption.

Given the strong impossibilities, we put forth a weaker definition of *distributional*-THC, where the graph is selected from some distribution (as opposed to worst-case).

- We present a formal separation between the definitions, by showing a distribution for which information theoretic distributional-THC is possible, but even topology-hiding broadcast is not possible information-theoretically with the standard definition.
- We demonstrate the power of our new definition via a new connection to adaptively secure low-locality MPC, where distributional-THC enables parties to "reuse" a secret low-degree communication graph even in the face of adaptive corruptions.

[*]Columbia University. E-mail: `marshall@cs.columbia.edu`.
[†]IDC Herzliya. E-mail: `elette.boyle@idc.ac.il`.
[‡]Boston University and Northeastern University. E-mail: `rancohen@ccs.neu.edu`.
[§]Columbia University. E-mail: `tal@cs.columbia.edu`.
[¶]IDC Herzliya. E-mail: `talm@idc.ac.il`.

Contents

1 Introduction **1**
- 1.1 Our Results: Standard (Strong) Topology Hiding . . . 2
- 1.2 Our Results: Distributional-Topology Hiding . . . 5
- 1.3 Open Problems . . . 7

2 Preliminaries **8**

3 TH-Broadcast on a Line Implies Key Agreement **9**

4 Perfect THC on a Cycle **13**

5 Distributional-Topology-Hiding Computation **18**

6 Distributional-THC with Hidden Sublinear Cuts **20**
- 6.1 Feasibility in the Distributional-THC Model . . . 21
  - 6.1.1 Ideal Functionalities used in the Construction . . . 22
  - 6.1.2 The protocol . . . 23
- 6.2 Impossibility in the Classical THC Model . . . 24

7 Sequential Composition of Low-Locality MPC **25**

Bibliography **28**

A Preliminaries (Cont'd) **31**
- A.1 UC Framework . . . 31

B Adaptively Secure Distributional THC with Hidden Sublinear Cuts (Cont'd) **32**

1 Introduction

In the setting of secure multiparty computation (MPC) [44, 30, 9, 19], a set of mutually distrusting parties wish to jointly perform a computation, such that no coalition of cheating parties can learn more information than their outputs (privacy) or affect the outputs of the computation any more than by choosing their own inputs (correctness). Seminal results initiated in the 1980s [44, 30, 9, 19], showed feasibility of MPC for general functions in many settings. The original definitions—and most works in the rich field of research they gave rise to—assume the participants are connected via a complete graph: i.e., any pair of parties can communicate directly with each other. However, in many settings the communication graph is in fact partial (either by design or by necessity). Moreover, as we discuss below, the network topology itself may be sensitive information to be hidden.

Several lines of work have studied secure computation over incomplete networks, in different contexts, but without attempting to hide the communication graph. For example, beginning with classical results in Byzantine agreement [26, 29], a line of work studied the feasibility of reliable communication over (known) incomplete networks (cf. [28, 27, 6, 39, 11, 7, 5, 17]). More recent lines of work study secure computation with restricted interaction patterns, motivated by improving efficiency, latency, scalability, usability, or security, including [32, 31, 12, 18, 8, 33, 13]. Some of these works utilize a secret communication subgraph of the complete graph that is available to the parties as a tool to achieve their goal; e.g., [12, 18, 13] use this idea in order to achieve communication locality.

Topology-Hiding Computation. Moran et al. [42] initiated the study of *Topology-Hiding Computation (THC)*, addressing the setting where the communication graph is incomplete and *sensitive*. Here, the goal is to allow parties who see only their immediate neighborhood (and possibly know that the graph belongs to some class), to securely compute arbitrary functions without revealing any other information about the graph topology. THC is of theoretical interest, but is also motivated by real-world settings where it is desired to keep the underlying communication graph private. These include social networks, ISP networks, vehicle-to-vehicle communications, wireless and ad-hoc sensor networks, and other Internet of Things networks.

THC protocols have been studied within two adversarial settings. In the *semi-honest* setting, the adversary follows the prescribed protocol but attempts to extrapolate disallowed information. In the *fail-stop* setting, the adversary may additionally abort the computation of parties at any point. Most existing THC protocols focus on the former, semi-honest setting, and this will also be our focus in this paper. We mention that in the fail-stop setting, Moran et al. [42] showed that THC is not possible except for extremely limited graphs/adversarial corruption patterns, and Ball et al. [3] and LaVigne et al. [40] showed how to achieve it with small leakage, assuming a secure hardware setup assumption, and assuming the hardness of decisional Diffie-Hellman (DDH), quadratic residuosity (QR), or learning with errors (LWE).

For the rest of this paper we assume the semi-honest setting (although some of our results could potentially be extended to fail-stop or malicious settings). In this regime, after several protocols achieving THC for various subclasses of graphs (log-diameter, cycles, trees, etc.) [42, 35, 1] from different cryptographic assumptions, Akavia et al. [2] showed how to achieve THC for all graphs from the DDH or QR assumptions, and LaVigne et al. [40] from LWE.

Our question: information-theoretic THC. Existing topology-hiding computation protocols provide a strong notion of hiding *all* information about the graph against an adversary who can corrupt an arbitrary number of parties. On the other hand, these existing protocols use structured cryptographic assumptions such as DDH, oblivious transfer (OT), or public-key encryption (PKE) with special properties, or even stronger assumptions such as a secure hardware box [3] to achieve more practical efficiency.

In this paper, we ask whether we can hide topology *information theoretically*, against a computationally unbounded adversary (in the plain model, with no correlated randomness or other trusted setup). A similar question, albeit only for (non-private) communication, was considered by Hinkelmann and Jakoby [34]. They claim an impossibility result for the class of all graphs, as well as a positive result showing an information-theoretic all-to-all communication protocol that leaks specific information about the graph (routing tables) but no other information. In contrast, here we are interested in (positive and negative) results for subclasses of graphs, as it is typically the case in applications of THC that the graph belongs to a certain known class. Looking ahead, we will see that what graphs are allowed and how they are chosen plays a crucial role for the feasibility of information-theoretic THC.

Ball et al. [3] have also considered this question, and showed that in their setting—semi-honest, arbitrary number of corruptions—the answer is negative. Specifically, they prove that even semi-honest secure topology-hiding *broadcast* for four parties or more, implies OT. Note that standard information-theoretic MPC for broadcast (where topology can be revealed) is trivial in the semi-honest setting, since there is nothing to hide: simply "flooding"—i.e., forwarding received messages to all neighbors—for sufficiently many rounds, works. Their proof crucially depends on the adversary corrupting at least half of the parties, namely no honest majority. This brings up a natural question, which we study in this paper:

> Can we take advantage of a low corruption threshold to achieve information-theoretic topology-hiding computation?

This question is particularly natural when we consider how fruitful this approach had been in the realm of standard (topology-revealing) secure computation. Indeed, classical results [9, 19, 43] show information-theoretic protocols for secure computation of general functions with an honest majority. However, in the topology-hiding realm, this question remained open (and explicitly mentioned in previous works such as [3]). In fact, the question was open even for the special case of topology-hiding broadcast (THB), where no privacy of inputs is required.

In this paper, we prove several results answering the above question, both negatively and positively, in different settings. All our positive results hold for general THC and all our negative results hold even for THB. Below we first describe our results for the standard definition of THC. We then discuss a new weaker definition of *distributional*-topology-hiding computation that we put forward, together with our results for this definition (as well as motivation and applications of this relaxation). Our results deepen our understanding of the nature of topology hiding, and point to a rich terrain of possibilities and applications of THC.

1.1 Our Results: Standard (Strong) Topology Hiding

We start by presenting both feasibility and infeasibility results of information-theoretic THC according to the standard definition from [41].

Broadcast on a line implies key agreement. We identify a large class of graphs for which information-theoretic THC is not possible, even when the semi-honest adversary can corrupt just a single party, and even without relying on input privacy.

> Theorem (informal): Topology-hiding broadcast for a graph with four parties on a line, resilient to one semi-honest corruption, implies key agreement.

Note that this theorem is for THB. Information-theoretic THC is trivially not possible here because the graph is only 1-connected, hence no privacy is possible with one corruption [27] (recall that we do not have any setup or correlated randomness).

At a high level, our key-agreement protocol considers two permutations of the four nodes: $G_0 = (1-2-3-4)$ and $G_1 = (2-3-4-1)$ (see Figure 1), with party 1 acting as the broadcaster. In this setting, a corrupted party 3 cannot distinguish which topology is being used: namely, whether 1 is a neighbor of 2 or of 4. This gap can be used to achieve a two-party key-agreement protocol. Consider an execution of the THB where Alice emulates parties 1, 2, and 3 while Bob emulates party 4, and another execution where Alice emulates parties 2 and 3 while Bob emulates parties 4 and 1. In both cases the messages that are exchanged by Alice and Bob—and so can be heard by an eavesdropper—consist of a partial view of party 3 in the THB protocol.

The key-agreement protocol now comprises of repeated phases, where in each phase Alice and Bob run two executions of the THB protocol. Each party tosses a private coin to decide whether to emulate the broadcaster party 1 in the first execution or the second. If Alice and Bob toss *different* coins, then either both emulate party 1 or nobody does. In this case they simply discard this phase and continue to the next one. However, if they toss the *same* coin, an eavesdropper will not be able to guess with more than negligible probability whether Alice emulated 1 in the first run and Bob in the second, or vice versa; hence, Alice and Bob can agree on this bit.

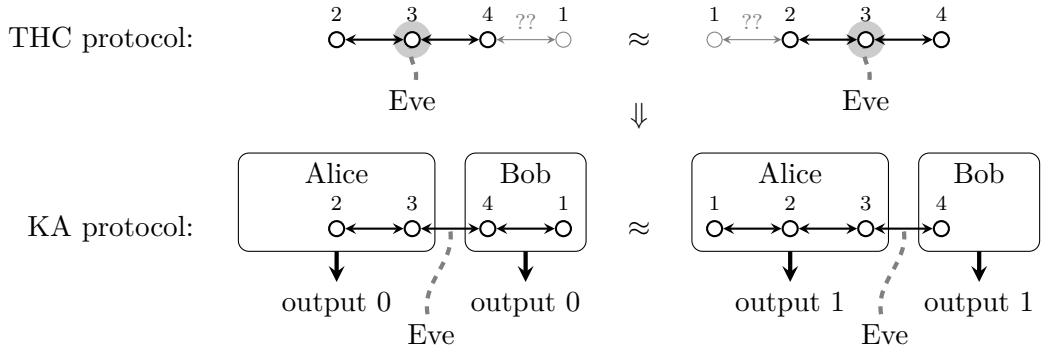

Figure 1: Four-party THB implies two-party key agreement. At the top are two configurations of the line, where party 3 is connected to party 2 on the left and to party 4 on the right. Party 3 does not know the location of party 1. At the bottom is the induced KA protocol, where Alice and Bob simulate executions of the THB protocol. The transcript visible to Eve forms a partial view of party 3's view in the THB; hence, Eve cannot distinguish between both scenarios.

Extension to broader classes of graphs. Clearly, this theorem holds for any class of graphs that includes all lines over $n \geq 4$ parties (topology-hiding here means that the order of the parties on the line, other than the two neighbors of the corrupted party, is not known, and in particular, the location of the broadcaster is hidden).

Our theorem further extends by a standard player-partitioning argument to more general classes of graphs, namely, any graph that can be partitioned into 4 "subsets" on a line. An example for such a class, most relevant to our positive result (below), are cycles of seven parties or more and with two corruptions (see Figure 2).

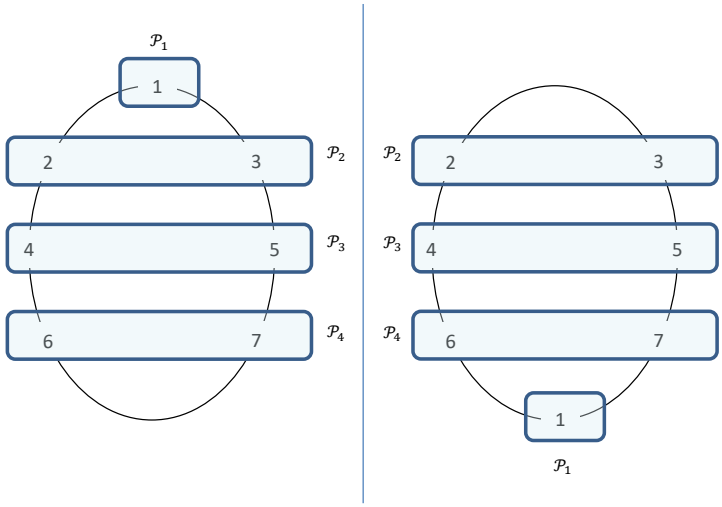

Figure 2: Reducing a seven-node cycle to a four-node line. Consider the partition of the seven nodes into $\mathcal{P}_1 = \{1\}$, $\mathcal{P}_2 = \{2, 3\}$, $\mathcal{P}_3 = \{4, 5\}$, and $\mathcal{P}_4 = \{6, 7\}$. The cycle on the left yields $(\mathcal{P}_1 - \mathcal{P}_2 - \mathcal{P}_3 - \mathcal{P}_4)$ and the cycle on the right yields $(\mathcal{P}_2 - \mathcal{P}_3 - \mathcal{P}_4 - \mathcal{P}_1)$.

Information-theoretic THC on a cycle. Our negative result rules out information-theoretic THC on cycles with two corruptions. Does a similar result hold even when we have a single corruption? Our next result shows that the answer is no. We construct a perfectly secure THC protocol on cycles, resilient to a single corruption.

> Theorem (informal): THC on a cycle with one corruption can be achieved information theoretically, with perfect correctness.

Note that this does not contradict the negative result claimed by Hinkelmann and Jakoby [34]. While that result precludes information-theoretic THC for the class of all graphs, here the parties know they are on a cycle (but do not know in which order the parties are arranged on the cycle).

The proof consists of two parts. Initially, we show how to realize anonymous and private pairwise communication. That is, each party can send a message to any other party on the cycle, but without knowing to whom he is sending, and from whom he is receiving messages. Instead, the sender can send the messages to the *relative location* on the cycle, i.e., he can send one message to a party that is 2 hops to his right, another message to a party that is 3 hops to his left, and so on. To send a message to a party that is j hops to his right (i.e., $n - j$ hops to his left), the sender secret shares the message and sends one share to his right neighbor and the second to his left neighbor. A party that receives a message from one of his neighbors forwards the message to his other neighbor. As there are $n - 1$ hops in the cycle, sending a message takes $n - 1$ rounds, and the sender (that is sending to the party that is j hops to his right) starts sending the right share after $n - j$ rounds and the left share after j rounds. This way, after $n - 1$ rounds, the receiver obtains both shares and can reconstruct the message.

Once establishing private pairwise channels, the parties can compute any function using the BGW protocol [9]. However, BGW cannot be executed immediately over an anonymous network, since to process input wires the real identities should be known, rather than the alias IDs (e.g., for computing $(x_1 + x_2) \cdot x_3$). To overcome this obstacle, we first observe that *symmetric* functions f can be implemented immediately via BGW over an anonymous communication network. Then, we generically reduce arbitrary f to the symmetric case, by having parties submit their real ID as part of their input (i, x_i), and computing the modified *symmetric* function f' which acts equivalently on all input pairs via multiplexing.

1.2 Our Results: Distributional-Topology Hiding

Having shown that information-theoretic THC is impossible for a large class of graphs even in the honest-majority setting, a natural question is whether we can construct weaker—but still useful—variants of THC for such settings. In particular, suppose we do not aim to hide everything about the graph, but rather just hide *something* about the graph, which will allow us to use the protocol as a building block in other applications.

As a motivating example, consider the work of Boyle et al. [13], who showed a protocol achieving adaptively secure MPC, where the actual communication graph has a sublinear cut, and thus is not an expander. Their protocol is in the so-called *hidden-channel model*, introduced in [18], where the adversary is unaware of the communication between honest parties (otherwise a trivial attack would separate the graph).[1] Intuitively, the adaptive security of their protocol hinges on the fact that the adversary cannot find which parties are on the small cut; if it could corrupt those parties, the security would be compromised. Thus, although hiding information about the topology was not their goal, it seems that the main tool used by [13] to prove their result is that something about the topology (where the sublinear cut is) is hidden. Intuitively, their protocol captures some notion of topology hiding.

Trying to formalize this claim and prove it within the existing framework of THC quickly fails. Indeed, the standard definition of THC (considered in Section 1.1 and in all prior work) captures security in "worst-case" graphs; hence, the communication graph is chosen by the environment. Since the environment can choose which parties to corrupt in a correlated way, it can simply corrupt the parties on the cut and break security of the protocol (even with static corruptions). This motivates us to define a weaker notion:

> We define *distributional* topology-hiding computation, where, informally, the environment only knows the distribution from which the graph is chosen, not the specific graph.

Defining distributional-topology hiding. Formalizing this definition poses some subtleties. In its most intuitive form, this definition resembles the *hidden-graph model* from [18]. In this model, the graph is sampled according to some predefined distribution, and each party learns its local neighborhood. Chandran et al. [18] used this model to construct adaptively secure MPC with sublinear communication locality; however, their protocol was *not* meant to hide topology, and indeed each graph was only valid for a one-time use. In the distributional-topology-hiding case, we wish to construct protocols that *do* hide the topology, and so can reuse the same graph.

[1] This is in fact the communication model that is considered in the topology-hiding setting, since if the communication is over standard private channels, the adversary would learn information about the graph just by observing with whom honest parties communicate.

To support hidden topology during the computation along with strong composition capabilities, we allow the environment to receive the communication graph from the ideal functionality (either the communication-graph functionality in the real world, or the graph-information functionality in the ideal world), before announcing its decision-bit: real or ideal. Once the environment has learned the graph, we fall back to a similar state as in the classical THC setting, and we cannot base the security of the protocol on the graph's entropy. For this reason, after the environment receives the graph, the ideal functionality will stop processing any further messages, and in a sense, the communication network enters an "out of order" state.

However, the environment might still attempt to misuse this additional power, and after receiving the communication graph, corrupt a set of parties in a way that will break security (e.g., corrupt the entire sublinear cut in the example above). This attack is quite subtle, since essentially, after learning the graph the environment has the capability to learn all of the inputs that were used in the protocol just from the messages received by a small set of parties (recall that we consider information-theoretic protocols in the plain model). Clearly, the simulator will not be able to simulate such an attack. One way to protect against this attack is to rely on *secure data erasures* and instruct every party to erase all of the received and sent messages as soon as the network goes out of order. However, since secure erasures form a strong assumption that cannot always be realized, and thus limit the model, we resort to an alternative, more general, solution. To overcome this subtlety, once the environment receives the graph the ideal functionality will provide the simulator with all of the input messages it received from honest parties. This new information will allow the simulator to simulate additional corruption requests that are issued as a function of the graph, and will balance the additional advantage the environment gained.

In a sense, the new definition guarantees privacy of the communication network *as long as it is active*; however, if the network enters an "out of order" state, it does not retain the privacy of the protocols that used it, unless secure data erasure are employed.

We note that since the new definition hides the communication graph from the environment while it is active, computation that depend on the communication graph itself (e.g., finding shortest paths) cannot be supported - this is another weakening of the original definition.

Relation to classical THC. Having formalized distributional-THC, one may ask whether this definition can be used to achieve meaningful computations, and whether it implies standard THC. We show that this definition can capture the intuitive topology-hiding property of the protocol in [13], discussed above. In fact, we modify their protocol to show a strong separation between the definitions. We construct a distribution $\mathcal{D}$ which, on the one hand, can be used for computing any function while hiding a sublinear cut between two cliques (tolerating a linear number of adaptive corruptions), and on the other hand, even broadcast cannot be computed in a topology-hiding manner (in the classical sense) using any graph in the support of $\mathcal{D}$ (tolerating merely a sublinear number of static corruptions).

Theorem (informal): We show a distribution $\mathcal{D}$ over graphs with n nodes such that:

- Distributional-THC of every function can be achieved with respect to $\mathcal{D}$, with information-theoretic security, against an adaptive semi-honest adversary.
- For any class of graphs $\mathcal{C}$ with $\mathcal{C} \cap \mathsf{supp}(\mathcal{D}) \neq \emptyset$, even broadcast cannot be computed information theoretically in the strong THC setting, even with static semi-honest corruptions (as it implies key agreement).

Connection to adaptively secure low-locality MPC. Finally, we demonstrate the power of our new definition via a new connection to adaptively secure low-locality MPC, where distributional-THC enables parties to "reuse" a secret low-degree communication graph even in the face adaptive corruptions. Concretely, this will enable sequential composition of the adaptively secure MPC protocol from [18] while maintaining sublinear locality. The starting point of [18] was any adaptively secure MPC protocol over pairwise private channels. They used the hidden-graph model to sample an Erdős-Rényi graph G (with sublinear degree and polylog diameter) and showed how to emulate pairwise private communication over the graph G. In addition, an elegant distributed sampling algorithm for a Erdős-Rényi graph was given in [18] (based on [38, 17]).

However, as discussed above, their protocol does not hide the topology of G, and so a fresh graph is used for every communication round. For this reason, their protocol can be used for executing MPC protocols with sublinear many communication rounds, and maintains sequential composition of sublinear many computations (otherwise the locality will blow up).

We show that if the private pairwise communication can be instantiated in a distributional-THC manner, the adaptively secure MPC protocol from [18] will be able to reuse the *same* secret Erdős-Rényi communication graph for polynomially many rounds, and so will remain secure under arbitrary sequential composition.

> Theorem (informal): If there exists an adaptively secure distributional-THC protocol for private pairwise communication with respect to the Erdős-Rényi distribution from [18] (tolerating a linear number of semi-honest corruptions), then there exists an honest-majority adaptively secure MPC protocol with sublinear locality (tolerating the same corruptions) that remains secure under polynomially many sequential executions.

We note that this theorem does not present a new feasibility result, as we do not yet know how to implement the required underlying adaptively secure distributional-THC protocol. We leave this as an interesting open problem. Instead, the theorem demonstrates the power and usefulness of our definition (despite its weakness compared to the original).

1.3 Open Problems

Our results from Section 1.1 characterize the feasibility of information-theoretic THC over *lines* and *cycles*. Ultimately, the desire is to provide a similar characterization for all graphs. An interesting starting point is to extend our understanding in broader classes of graph, e.g., wheel graphs or 3-regular graphs.

Another intriguing question to come up with more distributions over graphs that can be computed in a distributional-THC manner.

Finally, as mentioned above, it is not clear whether private pairwise communication can be realized with distributional-THC security with respect to the Erdős-Rényi distribution. Answering this question will have implications on low-locality adaptively secure MPC.

Additional related work. In an independent and concurrent work, Damgård et al. [25] investigate the feasibility of information-theoretic THC. Their setting is different from ours, as they consider a trusted setup phase to generate correlated randomness for the parties.

Organization of the paper. The preliminaries can be found in Section 2. Initially, we consider the standard THC definition and present our lower bound in Section 3, followed by the positive results in Section 4. We proceed to define distributional-THC in Section 5, show a separation between the definitions in Section 6, and conclude with the connection to low-locality MPC in Section 7.

2 Preliminaries

Notations. For $n \in \mathbb{N}$ let $[n] = \{1, \cdots, n\}$. We denote by κ the security parameter, by n the number of parties, and by t an upper bound on the number of corrupted parties. The empty string is denoted by ϵ.

UC framework. In Appendix A.1, we present an informal overview of the UC framework of Canetti [14]. Unless stated otherwise, we will consider computationally unbounded and semi-honest adversaries and environments. We will consider both *static* corruptions (where the corrupted parties are chosen before the protocol begins) and *adaptive* corruptions (where parties can get corrupted dynamically during the course of the computation), and explicitly mention which type of corruption ir considered in every section.

We will consider the standard *secure function evaluation* (SFE) functionality, denoted $\mathcal{F}^f_{\mathsf{sfe}}$. Informally, the functionality is parametrized by an efficiently computable function $f : (\{0,1\}^*)^n \to \{0,1\}^*$. Every honest party forwards its input received from the environment to the ideal functionality, and the simulator sends the corrupted parties' inputs. The functionality computes $y = f(x_1, \ldots, x_n)$ and returns y to every party. *Broadcast* is a special case of SFE for the function that receives an input from a single party, named *the broadcaster*, (formally, every other party gives the empty string ϵ as input) and delivers this value to every party as the output. We denote the broadcast functionality by $\mathcal{F}_{\mathsf{bc}}$.

Topology-hiding computation (THC). We recall the definition of topology-hiding computation from [42]. The real-world protocol is defined in a model where all communication is transmitted via the $\mathcal{F}^{\mathcal{G}}_{\mathsf{graph}}$ functionality (described in Figure 3). The functionality $\mathcal{F}^{\mathcal{G}}_{\mathsf{graph}}$ is parametrized by a family of graphs $\mathcal{G}$. Initially, before the protocol begins, $\mathcal{F}^{\mathcal{G}}_{\mathsf{graph}}$ receives the network communication graph G from a special graph party P_{graph}, makes sure that $G \in \mathcal{G}$, and provides to each party his local neighbor-set. Next, during the protocol's execution, the functionality receives a message to be delivered from a sender P_v to a receiver P_w and delivers the message if the edge (v, w) appears in the graph.

An ideal-model computation of a functionality $\mathcal{F}$ is augmented to provide the corrupted parties with the information that is leaked about the graph; namely, every ideal (dummy) party should learn his neighbor-set. To capture this, we define the wrapper-functionality $\mathcal{W}^{\mathcal{G}}_{\mathsf{graph\text{-}info}}(\mathcal{F})$, that runs internally a copy of the functionality $\mathcal{F}$. The wrapper receives the graph $G = (V, E)$ from P_{graph}, makes sure that $G \in \mathcal{G}$, and upon receiving an initialization message from a party P_i responds with its neighbor set $\mathcal{N}_G[i]$ (just like $\mathcal{F}^{\mathcal{G}}_{\mathsf{graph}}$). All other input messages are forwarded to $\mathcal{F}$ and every message from $\mathcal{F}$ is delivered to its recipient.

The functionality $\mathcal{F}^{\mathcal{G}}_{\mathsf{graph}}$

The n-party functionality $\mathcal{F}^{\mathcal{G}}_{\mathsf{graph}}$ is parametrized by a family of graphs $\mathcal{G}$ and proceeds with parties $P_1, \ldots, P_n$ and a special graph party P_{graph} as follows.

Initialization Phase:

Input: $\mathcal{F}^{\mathcal{G}}_{\mathsf{graph}}$ waits to receive the graph $G = (V, E)$ from P_{graph}. If $G \notin \mathcal{G}$, abort.

Output: $\mathcal{F}^{\mathcal{G}}_{\mathsf{graph}}$ outputs $\mathcal{N}_G[v]$ to each P_v.

Communication Phase:

Input: $\mathcal{F}^{\mathcal{G}}_{\mathsf{graph}}$ receives from a party P_v a destination/data pair (w, m) where $w \in \mathcal{N}_G[v]$ and m is the message P_v wants to send to P_w. (If w is not a neighbor of v, $\mathcal{F}^{\mathcal{G}}_{\mathsf{graph}}$ ignores this input.)

Output: $\mathcal{F}^{\mathcal{G}}_{\mathsf{graph}}$ gives output (v, m) to P_w indicating that P_v sent the message m to P_w.

Figure 3: The communication graph functionality

Definition 2.1 (Topology-hiding computation)**.** *We say that a protocol π securely realizes a functionality $\mathcal{F}$ in a topology-hiding manner with respect to $\mathcal{G}$ tolerating semi-honest t-adversaries if π securely realizes $\mathcal{W}^{\mathcal{G}}_{\mathsf{graph\text{-}info}}(\mathcal{F})$ in the $\mathcal{F}^{\mathcal{G}}_{\mathsf{graph}}$-hybrid model tolerating semi-honest t-adversaries.*

We note a few technical changes in the definition above compared to [42]. First, we let the graph functionality $\mathcal{F}_{\mathsf{graph}}$ and the wrapper $\mathcal{W}_{\mathsf{graph\text{-}info}}$ be parametrized by a family of graphs $\mathcal{G}$. This captures the fact that certain properties of the graphs might be inherently leaked e.g., the diameter of the graph [42] or that the graph is a cycle or a tree [1]. This technical adjustment has also been considered in [35]. A second difference is that we define the graph-information as a *wrapper functionality* around $\mathcal{F}$ rather than a separate functionality that is composed with $\mathcal{F}$. Although this difference is only syntactic with respect to the definition above, it will enable a cleaner definition of *distributional THC* in Section 5.

3 TH-Broadcast on a Line Implies Key Agreement

In this section, we show that a topology-hiding broadcast protocol of four parties (or more) connected in a line that tolerates one semi-honest corruption, implies the existence of two-party key-agreement protocols.

We define the following class of graphs $\mathcal{G}_{\mathsf{line}} = \{G_0, G_1\}$, where each graph has four nodes on a line: $G_0 = (1 - 2 - 3 - 4)$ and $G_1 = (2 - 3 - 4 - 1)$ (see Figure 1). Consider party 1 to be the broadcaster, then a corrupted party 3 will not know whether 1 is a neighbor of 2 or of 4. We next show how to utilize this property to achieve a two-party key-agreement protocol. The high-level idea is that either Alice will emulate parties $1, 2$, and 3 and Bob will emulate party 4, or that Alice will emulate parties 2 and 3, and Bob will emulate parties 4 and 1. An eavesdropper listening to their communication will in fact hear all the messages exchanged between party 3 and party 4 in the THB protocol, and therefore will not be able to guess with more than negligible probability who emulates party 1.

Theorem 3.1. *The existence of four-party topology-hiding broadcast with respect to class $\mathcal{G}_{\mathsf{line}}$ secure against semi-honest adversaries that may make a single corruption implies the existence of key agreement.*

High-level idea. Our key-agreement protocol proceeds in phases. In a given phase, Alice and Bob will jointly simulate the topology-hiding broadcast protocol on a line graph of nodes 1,2,3,4. Alice will always simulate nodes 2 and 3 and Bob will always simulate node 4. Alice and Bob will flip private coins to determine if they simulate 1. Note that it may be that neither or both of them simulate node 1. It will always be the case that node 2 has an edge to node 3 which is in turn has an edge to node 4. If Alice's coin is heads she will simulate node 1 with a unique edge to node 2. Similarly, if Bob's coin is heads, he will simulate node 1 with a unique edge to node 4. The node 1 will always be broadcaster, and will correspond to the bit agreed upon. The eavesdropper, Eve, will of course see the messages between 3 and 4 as Alice and Bob communicate to simulate the protocol execution. We will design our protocol so that Alice and Bob can identify when both or neither are controlling node 1 so they can throw them out, as the protocol will have no guarantees in this case. In the other cases, whether Alice or Bob controls 1 will indicate the bit agreed upon. This bit will be obvious to both Alice and Bob; however, it will be obscured from Eve. In particular, any advantage Eve has in guessing the bit can be used to break the topology hiding of the protocol. To increase the probability of successfully agreeing on a bit the protocol can simply be repeated. However, for simplicity we will specify and analyze the low-success version.

Proof. Let π be a topology-hiding broadcast protocol with respect to $\mathcal{G}_{\mathsf{line}}$, where node 1 is the broadcaster. Via sequential composition, we may assume π is a κ-bit broadcast protocol without. We use π to construct the following key-agreement protocol.

Protocol 3.2 (Two-party key agreement)**.**

1. *Alice sends two random κ-bit strings, r_1 and r_2, to Bob. These will be the strings broadcasted in the simulations of π.*
2. *Alice and Bob each flips a coin: $c_A, c_B \leftarrow \{0,1\}$, respectively. They will jointly simulate the protocol twice.*
   - *If $c_A = 1$, Alice will first simulate nodes 1 (broadcasting r_1), 2, and 3 in π. The second time, Alice will just simulate nodes 2 and 3. If $c_A = 0$, Alice will just simulate nodes 2 and 3 the first time and additionally simulate node 1 (broadcasting r_2), the second time.*
   - *If $c_B = 1$, Bob will first simulate nodes 1 (broadcasting r_1) and 4 in π. The second time, Bob will just simulate node 4. If $c_B = 0$, Bob will just simulate node 4 the first time, and additionally simulate node 1 (broadcasting r_2), the second time.*
3. *Alice and Bob jointly simulate π twice according to the roles designated above, communicating messages between 3 and 4 as needed.*
   - *If node 2 did not output either r_1 in the first simulation of π or r_2 in the second simulation of π, Alice outputs $\perp$. Otherwise, Alice outputs c_A.*
   - *If node 4 did not output either r_1 in the first simulation of π or r_2 in the second simulation of π, Bob outputs $\perp$. Otherwise, Bob outputs $1 - c_B$.*

There are 4 cases for how (c_A, c_B) is chosen (each occurring with probability 1/4. We will divide them into two sets (each occurring with probability 1/2): $c_A = c_B$ and $c_A \neq c_B$. We claim that in the first case both Alice and Bob output $\perp$ with probability $\geq 1 - 2^{1-\kappa}$. In the second case, we claim that both Alice and Bob output c_A with overwhelming probability and that Eve's output can be at most negligibly correlated with c_A. Thus, conditioned on Alice and Bob not outputting $\perp$ (which happens with probability negligibly close to 1/2), Alice and Bob will agree on a bit (with overwhelming probability) that is negligibly correlated with any bit outputted by an efficient eavesdropper. Therefore, it suffices to prove the claim for each case.

The case of $c_A = c_B$. If both $c_A = 1$ and $c_B = 1$, then neither Alice nor Bob is simulating the broadcasting node 1 in the second simulation. In which case, all outputs of π in this simulation is independent of r_2. Thus, the probability that either node 2 or node 4 outputs r_2 in the simulation is at most $2/2^\kappa$. Conversely, if both $c_A = 0$ and $c_B = 0$, then neither party is simulating node 1 in the first simulation and all outputs are independent of r_1. And similarly the probability that either node 2 or node 4 outputs r_1, in this case, is at most $2/2^\kappa$.

In either case there is a simulation where both node 2 and node 4 fail to output the chosen string with probability at least $1 - 2^{1-\kappa}$. Thus, both Alice and Bob will output $\perp$ with probability at least $1 - 2^{1-\kappa}$.

The case of $c_A \neq c_B$. In this case, in each simulation exactly one of Alice and Bob is simulating node 1, the broadcaster. By correctness, all nodes (including nodes 2 and 4) will output the string r_1 in the first simulation and r_2 in the second simulation with overwhelming probability. Thus, both Alice and Bob will output c_A (note that $c_A = 1 - c_B$, in this case) with overwhelming probability.

On the other hand, suppose Eve outputs a bit b such that $\Pr[b = c_A] = 1/2 + \alpha$. Note that Eve only sees the correspondence between nodes 3 and 4. We can use such an Eve to distinguish between running π on G_0 or G_1 with advantage at least $\alpha/3$. Moreover, it will distinguish with respect to a *specific* distribution of broadcast messages and topology: the one where both message and topology are chosen uniformly and independently.

A semi-honest adversary that has corrupted node 3 will wait until the protocol has completed and the output r has been received before simulating Eve. The adversary will flip a bit b': effectively guessing the opposite topology of actual execution 1-2-3-4 (in the case that $b' = 0$) or 2-3-4-1 (in the case that $b' = 1$). After the protocol has completed, the adversary will sample a random string r' and run Eve on a transcript comprised of r, r', the actual communication between nodes 3 and 4, and a communication between nodes 3 and 4 in a simulated execution where r' is broadcasted over the guessed topology. The simulated Eve will output a bit b. If $b = 1$, the adversary will output the 1-2-3-4 topology, and the 2-3-4-1 topology otherwise.

In the case that the adversary guessed correctly (which happens with probability 1/2), the transcript Eve is given is identically distributed to the that of the key-agreement protocol. In this case, the simulated Eve's bit will be α-correlated with the actual topology. In the other case, when Eve is given two independent invocations of the protocol on the same graph, Eve's output must be negligibly close to 1/2 by Claim 3.3 (below) and the security of π. Therefore, the probability the adversary outputs the correct topology is at least

$$1/4 - \mathsf{negl}(\kappa) + 1/4 + \alpha/2 > 1/2 + \alpha/3.$$

So, by the topology-hiding property, α must be negligible.

To complete the proof, we formally analyze Eve's success probability. Let r be an arbitrary κ-bit string and let G_0, G_1 defined as above. Let $\pi_{3,4}(G_0)$ and $\pi_{3,4}(G_1)$ denote the distribution of the transcript of an interaction between parties 3 and 4 when protocol π is executed with broadcast string r on G_0 and G_1, respectively.

Claim 3.3. *If for all probabilistic polynomial time $\mathcal{A}$ it holds that*

$$\left|\Pr_{\tau_0 \sim \pi_{3,4}(G_0)}[\mathcal{A}(\tau_0) = 1] - \Pr_{\tau_1 \sim \pi_{3,4}(G_1)}[\mathcal{A}(\tau_1) = 1]\right| < \epsilon,$$

then for all probabilistic polynomial time $\mathcal{A}'$ it holds that

$$\left|\Pr_{\tau_0^1, \tau_0^2 \sim \pi_{3,4}(G_0)}[\mathcal{A}'(\tau_0^1, \tau_0^2) = 1] - \Pr_{\tau_1^1, \tau_1^2 \sim \pi_{3,4}(G_1)}[\mathcal{A}'(\tau_1^1, \tau_1^2) = 1]\right| < 2\epsilon.$$

Proof. Let $\mathcal{A}'$ be a PPT adversary that

$$\left|\Pr_{\tau_0^1, \tau_0^2 \sim \pi_{3,4}(G_0)}[\mathcal{A}'(\tau_0^1, \tau_0^2) = 1] - \Pr_{\tau_1^1, \tau_1^2 \sim \pi_{3,4}(G_1)}[\mathcal{A}'(\tau_1^1, \tau_1^2) = 1]\right| \geq 2\epsilon.$$

We will use $\mathcal{A}'$ to construct $\mathcal{A}$ that can distinguish a single execution of π with probability at least ϵ. On input τ_b, $\mathcal{A}$ does the following:

1. Flip a random coin, b', and sample $\tau_{b'} \sim \pi_{3,4}(G_{b'})$.
2. Flip another random coin c:
   (a) If $c = 0$, output $\mathcal{A}'(\tau_b, \tau_{b'})$;
   (b) Otherwise, if $c = 1$, output $\mathcal{A}'(\tau_{b'}, \tau_b)$.

Now, we will analyze $\mathcal{A}$. Note that for both $b \in \{0, 1\}$ and $\tau_b \sim \pi_{3,4}(G_b)$, we have that $\mathcal{A}$ calls invoke $\mathcal{A}'$ on inputs distributed according to $\pi_{3,4}(G_b) \times \pi_{3,4}(G_b)$ with probability 1/2, and inputs distributed according to $\pi_{3,4}(G_0) \times \pi_{3,4}(G_1)$ and $\pi_{3,4}(G_1) \times \pi_{3,4}(G_0)$, each with probability 1/4. In other words,

$$\begin{aligned}
&\left|\Pr_{\tau_0 \sim \pi_{3,4}(G_0)}[\mathcal{A}(\tau_0) = 1] - \Pr_{\tau_1 \sim \pi_{3,4}(G_1)}[\mathcal{A}(\tau_1) = 1]\right| \\
&= \Bigg|\Bigg(\tfrac{1}{2} \cdot \Pr_{\tau_0^1, \tau_0^2 \sim \pi_{3,4}(G_0)}[\mathcal{A}'(\tau_0^1, \tau_0^2) = 1] \\
&\qquad + \tfrac{1}{4} \cdot \Pr_{\tau_0 \sim \pi_{3,4}(G_0), \tau_1 \sim \pi_{3,4}(G_1)}[\mathcal{A}'(\tau_0, \tau_1) = 1] \\
&\qquad + \tfrac{1}{4} \cdot \Pr_{\tau_0 \sim \pi_{3,4}(G_0), \tau_1 \sim \pi_{3,4}(G_1)}[\mathcal{A}'(\tau_1, \tau_0) = 1]\Bigg) \\
&\quad - \Bigg(\tfrac{1}{4} \cdot \Pr_{\tau_1^1, \tau_1^2 \sim \pi_{3,4}(G_1)}[\mathcal{A}(\tau_1^1, \tau_1^2) = 1] \\
&\qquad + \tfrac{1}{4} \cdot \Pr_{\tau_0 \sim \pi_{3,4}(G_0), \tau_1 \sim \pi_{3,4}(G_1)}[\mathcal{A}'(\tau_0, \tau_1) = 1] \\
&\qquad + \tfrac{1}{4} \cdot \Pr_{\tau_0 \sim \pi_{3,4}(G_0), \tau_1 \sim \pi_{3,4}(G_1)}[\mathcal{A}'(\tau_1, \tau_0) = 1]\Bigg)\Bigg| \\
&= \tfrac{1}{2} \cdot \left|\Pr_{\tau_0^1, \tau_0^2 \sim \pi_{3,4}(G_0)}[\mathcal{A}'(\tau_0^1, \tau_0^2) = 1] - \Pr_{\tau_1^1, \tau_1^2 \sim \pi_{3,4}(G_1)}[\mathcal{A}'(\tau_1^1, \tau_1^2) = 1]\right| \\
&\geq \epsilon.
\end{aligned}$$

□

This concludes the proof of Theorem 3.1. □

Next, we extend the lower bound to more classes of graphs using the player-partitioning technique.

Corollary 3.4. *Let $\mathcal{G}$ be a class of (connected) graphs with n nodes such that there exists a partition of the nodes into four subsets $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4$, and there exists graphs $\tilde{G}_0, \tilde{G}_1 \in \mathcal{G}$ such that:*

- *In $\tilde{G}_0$: there are no edges $(i,j) \in \mathcal{P}_1 \times \mathcal{P}_3$, or $(i,j) \in \mathcal{P}_2 \times \mathcal{P}_4$, or $(i,j) \in \mathcal{P}_1 \times \mathcal{P}_4$,*
- *In $\tilde{G}_1$: there are no edges $(i,j) \in \mathcal{P}_1 \times \mathcal{P}_3$, or $(i,j) \in \mathcal{P}_2 \times \mathcal{P}_4$, or $(i,j) \in \mathcal{P}_1 \times \mathcal{P}_2$.*

Let $t = |\mathcal{P}_3|$. Then, a THB protocol with respect to $\mathcal{G}$ tolerating semi-honest, static t-adversaries, implies the existence of key agreement.

Proof. Let π be such a THB protocol, and without loss of generality assume that the broadcaster is in $\mathcal{P}_1$. We will construct the following four-party broadcast protocol on a line with respect to the class of two graphs $\mathcal{G}_{\mathsf{line}} = (G_0, G_1)$, where

- $G_0 = (1 - 2 - 3 - 4)$,
- $G_1 = (2 - 3 - 4 - 1)$.

To define the protocol, every party i, for $i \in [4]$, emulates in its head the parties in $\mathcal{P}_i$ executing protocol π. Whenever a party $P_j \in \mathcal{P}_i$ wishes to send a message m to a party $P_{j'}$, proceed as follows: (1) If $P_{j'} \in \mathcal{P}_i$, party i simulates in its head party $P_{j'}$ receiving the message m from party P_j. (2) If $P_{j'} \in \mathcal{P}_{i'}$ for some $i' \neq i$, send the message (j, j', m) to party i'; in this case, party i' simulates party $P_{j'}$ receiving the message m from party P_j.

Note that for $b \in \{0,1\}$, an execution of the four-party THB over G_b corresponds to an execution of the protocol π over $\tilde{G}_b$. Since $|\mathcal{P}_3| = t$ and π is t-secure, it holds that the new protocol is secure tolerating a single corruption of party P_3. The proof now follows from Theorem 3.1. □

An example for a family of graphs that satisfies the above requirements are cycles of seven nodes tolerating two corruptions. Indeed, consider the partition

$$\mathcal{P}_1 = \{1\}, \quad \mathcal{P}_2 = \{2,3\}, \quad \mathcal{P}_3 = \{4,5\}, \quad \mathcal{P}_4 = \{6,7\}.$$

Then, the two cycle-graphs $\tilde{G}_0 = (2 - 1 - 3 - 4 - 6 - 7 - 5)$ and $\tilde{G}_1 = (2 - 3 - 4 - 6 - 1 - 7 - 5)$ satisfy the properties of the corollary, as illustrated in Figure 2.

4 Perfect THC on a Cycle

In this section, we show a *perfectly secure* topology-hiding *computation* protocol tolerating a single semi-honest corruption with respect to cycles. Note that in this setting we are only hiding a permutation of the nodes.

Theorem 4.1. *Let $n > 2$, and let f be an efficiently computable n-party function. Then, $\mathcal{F}^f_{\mathsf{sfe}}$ can be securely realized in a topology-hiding manner with respect to the class of graphs that includes all cycles on n nodes, tolerating a single, semi-honest corruption. Moreover, the protocol is perfectly correct and perfectly secure.*

To prove Theorem 4.1, we will first show how to realize anonymous secure channels without revealing the topology of the cycle. Given anonymous secure channels, it is not difficult to realize general THC on a cycle.

Private Anonymous Communication over a Cycle

We begin by defining the anonymous communication functionality. This randomized functionality initially assigns aliases to all parties based on their location, so that the parties can address each other by these aliases. Specifically, the functionality will choose a random party to be assigned with the alias '1', and will choose a random orientation of "left" and "right" for its outgoing edges. This will define an alias for each other party, in an increasing order going to the left. Each party will receive its alias and orientation (hence allowing it to compute the alias of any party that is a certain number of hops away in each direction). Then, each party can privately send messages to any alias of their choice. The party associated with the alias will receive the message along with the alias of the sender. A full description is provided in Figure 4.

The functionality $\mathcal{F}_{\mathsf{anon}}$

The n-party randomized reactive functionality $\mathcal{F}_{\mathsf{anon}}$ proceeds with parties $P_1, \ldots, P_n$ as follows.

Initialization Phase:

Input: Each party invokes the functionality by providing his neighbor-set (i.e., the pair of neighbors).

Alias assignment: Upon invocation, the functionality reconstructs the cycle. The functionality proceeds by selecting $u \in [n]$ at random, setting $\mathrm{id}_u = 1$, and randomly selecting an orientation of the two neighbors of P_u as either "left" or "right." This defines the orientation of the rest of the cycle edges for each other party.

Output: Each P_v receives an alias $\mathrm{id}_v \in [n]$ such that if P_v is $k \in \{0, \ldots, n-1\}$ hops to the left of P_u, then $\mathrm{id}_v = k + 1$. Further, each P_v receives a bit pointing out which of its two neighbors is "left" (increasing alias order).

Communication Phase:

Input: Each party P_v (with alias id_v) specifies a set of messages to all other parties $\{(\mathrm{id}_v, j, m_j^{\mathrm{id}_v})\}_{j \neq \mathrm{id}_v}$, where $m_j^{\mathrm{id}_v}$ is a message from the party with alias id_v intended for the party with alias j.

Output: Each party P_v assigned the alias id_v in the initialization phase receives a set of messages from every other alias $\{(j, \mathrm{id}_v, m_{\mathrm{id}_v}^j)\}_{j \neq \mathrm{id}_v}$ indicating the party assigned with alias j sent to the party with id_v the message $m_{\mathrm{id}_v}^j$ (for all $j \neq \mathrm{id}_v$).

Figure 4: The anonymous communication functionality

Let $\mathcal{G}_{\mathsf{cycle}}(n)$ be the class of cycles over n nodes. Next, we show how to perfectly securely realize $\mathcal{W}_{\mathsf{graph\text{-}info}}^{\mathcal{G}_{\mathsf{cycle}}(n)}(\mathcal{F}_{\mathsf{anon}})$ in the $\mathcal{F}_{\mathsf{graph}}^{\mathcal{G}_{\mathsf{cycle}}(n)}$-hybrid model.

High-level idea. At a high level, our protocol proceeds in two phases. In the first phase, a fixed designated party P^*, will randomly assign an alias in $[n]$ for itself, as well as an orientation of left and right. This defines aliases for the rest of the parties based on their distance from P^* and its randomly chosen alias. A protocol is then performed to securely provide all parties with their alias and orientation. In the communication phase, parties use their output from the initialization phase and the 2-connectedness of the cycle to securely communicate with other parties (specified via their

aliases, indicating how many hops away they are). Before specifying the full protocol in Figure 5, we give some intuition.

To begin, suppose that $n = 2k$ is even, and that a party P_v wishes to send a message m to the party directly opposite it on the cycle (denote that party as P_u, although P_v does not know P_u's identity, just location). This can be done easily by uniformly sampling r and forwarding $m \oplus r$ to the right and r to the left (where right and left are from some arbitrary orientation). If other parties forward received messages in the same direction, after exactly k rounds, P_u will receive $m \oplus r$ and r simultaneously and can compute $(m \oplus r) \oplus r = m$. Because every other party sees either $m \oplus r$ or r, but not both, this will be uniformly distributed ensuring privacy of the message. By delaying timing of messages to left and right appropriately, we can adjust the protocol to allow any party to deliver a message to any other party that is a given number of hops away.

Once aliases have been agreed upon by all parties, the above will in fact suffice for the communication phase, as there will be nothing more to hide. However, for the initialization phase, if the designated party P^* simply uses the above to deliver aliases to other parties, this will leak the distance from the sender P^*. Hence, we will have all parties perform the above secure message sending protocol to all other parties, in parallel. The designated party P^* will send the actual aliases to each other party, while all other parties will perform the above as if sending 0 to all other parties. Note that message privacy here is only being used to hide the location of the designated party.

To perform the above in parallel, in each round parties will take the message received from the left in the previous round, XOR it with what they should send to the right themselves according to the secure message passing above, and then send the result to the right. They behave identically with respect to messages travelling in the other direction. In the final round, all parties simply XOR what they received from the left and right to receive their own alias. Moreover, because up to that point the view of any single party is simply a sequence of random messages, the location of P^* remains hidden. (The final messages of P^* will not be uniform, but XOR to 0.)

Lemma 4.2. *Let $n > 2$. Protocol $\pi_{\mathsf{anon\text{-}cycle}}$ perfectly securely realizes $\mathcal{F}_{\mathsf{anon}}$ in a topology-hiding manner with respect to the class of graphs that includes all cycles on n nodes, tolerating a single semi-honest adversary.*

Proof (sketch). Let $\pi : [n] \to [n]$ denote the map such that $\mathrm{id}_i \mapsto i$ for the id's implicitly defined by P^*. Let $\alpha : [n] \to [n]$ denote the cyclic permutation such that $i \mapsto i + 1$ for $i < n$ and $n \mapsto 1$. Additionally, let $\alpha^{(k)}$ denote k sequential applications of α. We take left and right to denote the orientation selected by P^* in the initialization phase.

For correctness, consider the sequence of messages: the message sent left by the party left of P_u in the first round, the message send left by the party two nodes left of P_u, and so on until the message that is delivered to P_u from the right in final round. Each subsequent message is formed by XORing with the previous message in the sequence. Because all parties other than P^* behave identically with respect to each direction in this phase, we may assume they all chose an orientation consistent with P^*. Then, we can observe that P_u receives $\pi^{-1}(u) \oplus \bigoplus_{j=1}^{n-1} r_j^{\pi(\alpha^{(j)}(\pi^{-1}(u)))}$ on the right in the final round of the initialization phase. Via the same argument, we can see P_u receives $r_j^{\pi(\alpha^{(j)}(\pi^{-1}(u)))}$ on the left of the initialization phase.

Protocol $\pi_{\mathsf{anon\text{-}cycle}}$

Hybrid Model: The n party protocol is defined in the $\mathcal{F}_{\mathsf{graph}}$-hybrid model.

Common Input: A designated party $P^* \in \{P_1, \ldots, P_n\}$ responsible for selecting the aliases.

The Protocol:

Initialization Phase: In the initialization phase the parties establish their aliases.

- Party $P^* = P_{j^*}$ picks a random orientation of its neighbors as "left" and "right." It samples a random alias $\sigma \in [n]$ for itself. Then, for $i = 0, \ldots, n-\sigma$, sets $m_i^{j^*} = \sigma + i$, and for $i = n-\sigma+1, \ldots, n-1$ sets $m_i^{j^*} = i - \sigma$. (Thus, for all i, $m_i^{j^*}$ is the alias of whichever party is i hops left of P^*)
- Each other party $P_k \neq P^*$ arbitrarily picks an orientation of its neighbors as "left" and "right," and sets $m_i^k = 0$ for $i = 0, \ldots, n-1$.
- Every party P_ℓ (including P^*) additionally samples $n-1$ independent uniform random values $r_1^\ell, \ldots, r_{n-1}^\ell$.
- For rounds $i = 1, \ldots, n-1$, every party P_ℓ samples random r_ℓ^i and sends $m_i^\ell \oplus r_i^\ell \oplus (p_R^\ell)$ to the left and $r_{n-i}^\ell \oplus p_L^\ell$ to the right, where p_L^ℓ and p_R^ℓ are the messages received by P_ℓ in the previous round from the left and right, respectively.
- Party P^* outputs alias σ. Every other party $P_k \neq P^*$ outputs $p_L^k \oplus p_R^k$, where p_L^k and p_R^k are the messages received by P_k from the left and right (respectively) in the last round.

  Additionally, parties exchange aliases with their neighbors to get a consistent orientation. We take right to denote direction of decreasing id's. (Notice that, relative to alias j, alias i is distance $i - j \pmod n$ to the left. Therefore, we can simply assume all parties subsequently share the same orientation as that chosen by P^*.)

Communication Phase: Each "round" of the communication phase is performed in n sub-phases, each corresponding to a different alias responsible for sending. Each subphase lasts exactly $n-1$ rounds. (The subphases can be performed in parallel, but the sequential presentation is simpler.)

- **Private input:** Each party P_i has input of the form $\{(\mathrm{id}_i, \ell, m_\ell^{\mathrm{id}_i})\}_{\ell \neq \mathrm{id}_i}$ (where ℓ denotes an alias and $m_\ell^{\mathrm{id}_i}$ the message to be sent to the party with alias ℓ by the party with alias id_i). We take id_i to denote the alias output by P_i in the initialization phase.
- **Protocol:** For sub-phases $j = 1, \ldots, n$:
  - Party P_i with $\mathrm{id}_i = j$ samples $n-1$ independent uniform random values $r_1, \ldots, r_{n-1}$.
  - For rounds $k = 1, \ldots, n-1$:
    - P_i sends $m_{j-k \pmod n}^j \oplus r_k$ to the left and r_{n-k} to the right.
    - All other parties forward messages received from the left in the previous round to the right, and messages received from the right in the previous round to the left.
  - After receiving in the final round, each party P_ℓ with an alias $\mathrm{id}_\ell \neq j$ (locally) sets $\hat{m}_\ell^j$ as the XOR of the last message received from the left and right.
- **Output:** Each party P_i outputs $\{(j, \mathrm{id}_i, \hat{m}_{\mathrm{id}_i}^j)\}_{j \neq \mathrm{id}_i}$.

Figure 5: Securely realizing $\mathcal{F}_{\mathsf{anon}}$ in a topology-hiding manner, for cycles

For security, note that if we view the r_j^i values each party sends to the right and left as traveling around the cycle in either direction (having values XORed with them), the only other party that sees both is the one that they arrive at simultaneously in the last round. Thus, to all other parties, they are uniformly distributed.

Therefore, the view of the corrupted party P_c is simply uniformly distributed messages in each round, until the last. In the last round, if $P_c \neq P^*$, party P_c receives two random messages (one from each side) that XOR to a random $i \in [n]$. If $P_c = P^*$, party P_c receives two random messages that XOR to zero. This can be simulated by simply sending random messages until the last round, where the messages XOR to a uniformly drawn $i \leftarrow [n]$ if $P_c \neq P^*$ and 0 otherwise. Because of this simulation the view of any party is clearly independent of the ordering of parties outside that party's immediate neighborhood.

The correctness and security of the communication phase proceed similarly, except here we will use the fact that the relative positions of id_c and the id_i to start simulating via uniformly random values on a given side. The full simulator is described below.

Initialization Phase:

- Let P_u be the corrupted party. Get $\mathcal{N}_G[u]$ from $\mathcal{W}^{\mathcal{G}}_{\mathsf{graph\text{-}info}}(\mathcal{F}_{\mathsf{anon}})$.
- Invoke $\mathcal{W}^{\mathcal{G}}_{\mathsf{graph\text{-}info}}(\mathcal{F}_{\mathsf{anon}})$ with $\mathcal{N}_G[u]$ (as the input to $\mathcal{F}_{\mathsf{anon}}$) and receive back id_u and the orientation.
- If $P_u \neq P^*$, deliver uniformly random messages from neighbors for first $n-2$ rounds. In final round, deliver uniformly random messages conditioned on them XORing to id_u.

  Otherwise, deliver uniformly random messages from neighbors to either side for the first $n-2$ rounds. In the final round, deliver uniformly random messages that XOR to 0.

Communication Phase: In each communication "round," get from the environment the tuples $\{(\mathrm{id}_u, i, m^i_{\mathrm{id}_u})\}_{i \neq u}$ (as the input of P_u).

In the i'th sub-phase of each "round,"

- From round $\mathrm{id}_u - i \pmod n$ of the sub-phase until the penultimate round, give P_u uniformly random messages from the right.
- From round $i - \mathrm{id}_u \pmod n$ of the sub-phase until the penultimate round, give P_u uniformly random messages from the left.
- In the final round if $\mathrm{id}_u \neq i$, give P_u random messages conditioned on them XORing to $m^i_{\mathrm{id}_u}$.

  If $\mathrm{id}_u = i$, P_u doesn't receive anything throughout the phase.

□

THC from Secure Anonymous Channels

Equipped with secure anonymous point-to-point channels, we can now use standard honest-majority MPC techniques to achieve general THC.

Lemma 4.3. *Let $n \in \mathbb{N}$, let $t \leq n/2$, and let f be an efficiently computable n-party function. Then, $\mathcal{F}^f_{\mathsf{sfe}}$ can be UC-realized with perfect security in $\mathcal{F}_{\mathsf{anon}}$-hybrid model, tolerating t semi-honest corruptions.*

Proof (sketch). Without loss of generality, it suffices to consider functionalities that give the same output to all parties. Let f' denote the *symmetric* functionality that takes in n tuples of the form $(i, x_i) \in [n] \times \{0,1\}^n$ and outputs $f(x_1, \ldots, x_n)$ if all i are distinct, and $\perp$ otherwise. Note that for any permutation π of $[n]$ (describing $i \mapsto \mathrm{id}_i$, the alias of P_i), it holds that

$$f'\left(\left(\pi^{-1}(1), x_{\pi^{-1}(1)}\right), \ldots, \left(\pi^{-1}(n), x_{\pi^{-1}(n)}\right)\right) \equiv f(x_1, \ldots, x_n).$$

So, to complete the proof, parties simply securely evaluate f' under their aliases, where the input of P_i with alias id_i is (i, x_i), using the BGW protocol [9] over secure anonymous channels (between aliased identities) provided by $\mathcal{F}_{\mathsf{anon}}$. □

Putting together Lemma 4.2 and Lemma 4.3, and using UC-composition (see Theorem A.1 in appendix), completes the proof of Theorem 4.1, our positive result for cycles with one corruption.

5 Distributional-Topology-Hiding Computation

In this section, we present a relaxed notion of topology-hiding computation. Namely, it is not required that *all* of the topology of the graph will remain hidden, but only certain properties of the graph. The crucial difference to THC is that the functionality does not receive the graph from a graph party; rather, the communication-graph functionality is parametrized by a distribution over graphs and locally samples a graph from this distribution. As a result of this modification, the environment is ignorant of the actual graph that is used during the communication phase.

The functionality $\mathcal{F}^{\mathcal{D}}_{\mathsf{dist\text{-}graph}}$

The n-party functionality $\mathcal{F}^{\mathcal{D}}_{\mathsf{dist\text{-}graph}}$, parametrized by a distribution $\mathcal{D}$ over graphs of n nodes, proceeds with parties $P_1, \ldots, P_n$ and a special graph party P_{graph} as follows.

Initialization Phase:

- **Input:** $\mathcal{F}^{\mathcal{D}}_{\mathsf{dist\text{-}graph}}$ receives an initialization input from every party P_v. Upon receiving the first input, $\mathcal{F}^{\mathcal{D}}_{\mathsf{dist\text{-}graph}}$ samples a graph $G = (V, E) \leftarrow \mathcal{D}$.
- **Output:** $\mathcal{F}^{\mathcal{D}}_{\mathsf{dist\text{-}graph}}$ outputs $\mathcal{N}_G[v]$ to each P_v.

Communication Phase:

- **Input:** $\mathcal{F}^{\mathcal{D}}_{\mathsf{dist\text{-}graph}}$ receives from a party P_v a destination/data pair (w, m) where $w \in \mathcal{N}_G[v]$ and m is the message P_v wants to send to P_w. (If w is not a neighbor of v, $\mathcal{F}^{\mathcal{D}}_{\mathsf{dist\text{-}graph}}$ ignores this input.)
- **Output:** $\mathcal{F}^{\mathcal{D}}_{\mathsf{dist\text{-}graph}}$ gives output (v, m) to P_w indicating that P_v sent the message m to P_w.

Termination Phase:

- **Input:** $\mathcal{F}^{\mathcal{D}}_{\mathsf{dist\text{-}graph}}$ receives a termination input from P_{graph}.
- **Output:** $\mathcal{F}^{\mathcal{D}}_{\mathsf{dist\text{-}graph}}$ outputs the graph G to P_{graph} and stops processing further messages.

Figure 6: The distributional-graph-communication functionality

As discussed in Section 1.2, we require strong composition capabilities from this definition. Therefore, the environment is allowed to ask for the graph. This is done via a special graph party P_{graph}. Unlike in classical THC, where P_{graph} is used to give the graph to the functionality, here P_{graph} is used to ask the graph from the functionality. Once the environment asks for the graph, the communication functionality enters an "out of order" state and stops processing other messages.

As before, the ideal-model computation of a functionality $\mathcal{F}$ needs to be augmented to provide the simulator with the appropriate leakage on the graph, i.e., the neighbor-set of each corrupted party. Toward this purpose, we define a graph-information wrapper functionality around $\mathcal{F}$, denoted $\mathcal{W}^{\mathcal{D}}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F})$. Initially, the wrapper samples a graph from the distribution and provides every corrupted party with the neighbor-set. All subsequent input messages are forwarded to $\mathcal{F}$ and all messages from $\mathcal{F}$ are delivered to their recipients.

To keep the graph hidden from the environment during the computation phase, $\mathcal{W}^{\mathcal{D}}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F})$ does not send the neighbor-set to honest parties. The environment can adaptively issue corruption requests, and upon any such adaptive corruption $\mathcal{W}^{\mathcal{D}}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F})$ outputs the neighbor-set of the newly corrupted party.

As before, the environment can request the communication graph via a special graph party P_{graph}. After receiving this request from P_{graph}, the wrapper functionality stops processing further messages, other than corruption requests. As explained in Section 1.2, to balance the advantage given to the environment, that now can corrupt parties as a function of the graph, after giving the graph to P_{graph}, the wrapper gives the simulator all of the input messages it received.

The wrapper functionality $\mathcal{W}^{\mathcal{D}}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F})$

The n-party wrapper functionality $\mathcal{W}^{\mathcal{D}}_{\mathsf{dist\text{-}graph\text{-}info}}$, parametrized by a distribution $\mathcal{D}$ over graphs of n nodes, internally runs a copy of $\mathcal{F}$ and proceeds with parties $P_1, \ldots, P_n$ and a special graph party P_{graph} as follows.

Initialization Phase:

Input: $\mathcal{W}^{\mathcal{D}}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F})$ receives an initialization input from every party P_v. Upon receiving the first input, $\mathcal{W}^{\mathcal{D}}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F})$ samples a graph $G = (V, E) \leftarrow \mathcal{D}$.

Output: $\mathcal{W}^{\mathcal{D}}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F})$ outputs $N_G[v]$ to each corrupted P_v.

Computation Phase:

Input: $\mathcal{W}^{\mathcal{D}}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F})$ forwards every message it receives to $\mathcal{F}$.

Output: Whenever $\mathcal{F}$ sends a message, $\mathcal{W}^{\mathcal{D}}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F})$ forwards the message to the recipient.

Termination Phase:

Input: $\mathcal{W}^{\mathcal{D}}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F})$ receives a termination input from P_{graph}.

Output: $\mathcal{W}^{\mathcal{D}}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F})$ sends a termination message to the simulator, including all input messages sent to $\mathcal{F}$, outputs the graph G to P_{graph} and stops processing further messages, except for corruption requests.

Corruption request: Once a party P_v gets corrupted, $\mathcal{W}^{\mathcal{D}}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F})$ sends $N_G[v]$ to P_v.

Figure 7: The distributional-graph-information wrapper functionality

Definition 5.1 (Distributional topology hiding)**.** *Let $\mathcal{D}$ be a distribution over graphs with n nodes. A protocol π securely realizes a functionality $\mathcal{F}$ in a* distributional-topology-hiding manner with respect to $\mathcal{D}$ *tolerating semi-honest t-adversaries, if π securely realizes $\mathcal{W}^{\mathcal{D}}_{\text{dist-graph-info}}(\mathcal{F})$ in the $\mathcal{F}^{\mathcal{D}}_{\text{dist-graph}}$-hybrid model tolerating semi-honest t-adversaries.*

The relation between the definitions. We show that Definition 5.1 is indeed a relaxation of Definition 2.1. We start by showing that every protocol that satisfies Definition 2.1 will also satisfy Definition 5.1, at least as long as the functionality does not depend on the graph. Next, in Section 6, we will show a separation between the definitions.

Consider an environment $\mathcal{Z}$ of the form $\mathcal{Z} = (\mathcal{Z}_1, \mathcal{Z}_2)$, where $\mathcal{Z}_1$ invokes P_{graph} with a graph $G \in \mathcal{G}$ and receives back its output, and $\mathcal{Z}_2$ interacts with the parties and the adversary (without knowing the output received by $\mathcal{Z}_1$) and outputs the decision bit. We say that an n-party functionality $\mathcal{F}$ does not depend on the communication graph if for every family $\mathcal{G}$ of graphs with n nodes and every environment $\mathcal{Z} = (\mathcal{Z}_1, \mathcal{Z}_2)$ as described above, the output of $\mathcal{Z}$ (i.e., the output of $\mathcal{Z}_2$) in an ideal computation of $\mathcal{W}^{\mathcal{G}}_{\text{graph-info}}(\mathcal{F})$ is identically distributed as the output of $\mathcal{Z}$ in an ideal computation of $\widetilde{\mathcal{W}}^{\mathcal{G}}_{\text{graph-info}}(\mathcal{F})$, where $\widetilde{\mathcal{W}}^{\mathcal{G}}_{\text{graph-info}}$ acts like $\mathcal{W}^{\mathcal{G}}_{\text{graph-info}}$ with the exception that it ignores the graph G it receives and chooses an arbitrary graph from $\mathcal{G}$ instead. In the modified functionality, the input provided by the environment is independent of the communication graph; hence, if the output of the functionality is identically distributed in both cases, it can't be dependent on the graph structure.

Theorem 5.2. *Let $\mathcal{F}$ be functionality that does not depend on the communication graph and let $\mathcal{D}$ be an efficiently sampleable distribution over graphs with n nodes. If $\mathcal{F}$ can be securely realized in a topology-hiding manner with respect to* $\text{supp}(\mathcal{D})$*, then $\mathcal{F}$ can be securely realized in a distributional-topology-hiding manner with respect to $\mathcal{D}$.*

Proof. Assume that $\mathcal{F}$ cannot be securely realized in a distributional-topology-hiding manner with respect to $\mathcal{D}$, i.e., for every protocol and every simulator for the dummy adversary, there exists an environment $\mathcal{Z}$ that can create a non-negligible distinguishing advantage. Note that initially, $\mathcal{Z}$ knows only the distribution $\mathcal{D}$ but not the actual graph, but at any point can invoke P_{graph} to obtain the graph. We will show that $\mathcal{F}$ cannot be securely realized in a topology-hiding manner with respect to $\text{supp}(\mathcal{D})$.

We use $\mathcal{Z}$ to construct an environment $\mathcal{Z}'$ as follows. Initially, $\mathcal{Z}'$ samples a graph $G \leftarrow \mathcal{D}$ and sends it to P_{graph} to initialize the communication graph functionality (or the graph-information functionality). Next, $\mathcal{Z}'$ invokes $\mathcal{Z}$ and forwards any message from $\mathcal{Z}$ to the parties or the adversary, and vice versa. Once an honest party receives its neighbor-set from the functionality, $\mathcal{Z}'$ does not forward the message to $\mathcal{Z}$, but upon a corruption of a party $\mathcal{Z}'$ provides its neighbor-set to $\mathcal{Z}$. If $\mathcal{Z}$ asks P_{graph} to get the graph, $\mathcal{Z}'$ responds with the graph G and proceed to process only corruption requests from $\mathcal{Z}$. Finally, $\mathcal{Z}'$ outputs the output of $\mathcal{Z}$ and halts. Clearly, $\mathcal{Z}'$ has the same distinguishing probability as $\mathcal{Z}$, and the proof follows. □

6 Distributional-THC with Hidden Sublinear Cuts

In this section, we show a distributional-THC protocol that hides sublinear cuts between two linear-size cliques in the communication graph, and tolerates a linear number of adaptive semi-honest corruptions. The protocol is based on a recent work by Boyle et al. [13], that constructed

an adaptively secure MPC protocol in the dynamic-graph setting (where every party can talk to every other party, but dynamically decides on its neighbor-set).

In Section 6.1, we present the protocol in the distributional-THC setting that can hide sublinear cuts against adaptive corruptions, and in Section 6.2 we show that a similar result cannot be achieved in the classical THC setting.

6.1 Feasibility in the Distributional-THC Model

We start by defining the distribution of potential communication graphs in the n-party protocol.

Definition 6.1. *Let $n = 4m + 1$ for $m \in \mathbb{N}$, and let $n' = \log^c n$ for a constant $c > 1$. Denote*

$$\mathcal{P}_1 = \{1, \ldots, m\}, \quad \mathcal{P}_2 = \{m+1, \ldots, 2m\}, \quad \mathcal{P}_3 = \{2m+1, \ldots, 3m\}, \quad \mathcal{P}_4 = \{3m+1, \ldots, 4m\}.$$

Given a bit $b \in \{0, 1\}$ and two vectors $\boldsymbol{i} = (i_1, \ldots, i_{n'})$ and $\boldsymbol{j} = (j_1, \ldots, j_{n'})$ in $[m]^{n'}$ with distinct coordinates, i.e., $i_k \neq i_{k'}$ and $j_k \neq j_{k'}$ for $k \neq k'$, define the graph $G_n(b; \boldsymbol{i}; \boldsymbol{j})$ as follows:

- *Two cliques of size $2m$, $\mathcal{P}_1 \cup \mathcal{P}_2$ and $\mathcal{P}_3 \cup \mathcal{P}_4$.*
- *The edges $(m + i_k, 2m + j_k)$ for every $k \in [n']$ (i.e., a sublinear cut between $\mathcal{P}_2$ to $\mathcal{P}_3$).*
- *The edges $(4m + 1, i)$, for every $i \in \mathcal{P}_1$ if $b = 0$, or for every $i \in \mathcal{P}_4$ if $b = 1$ (i.e., connecting $4m + 1$ to either $\mathcal{P}_1$ or $\mathcal{P}_4$).*

We define the distribution $\mathcal{D}_{\mathsf{cut}}(n, c)$ over graphs of n nodes by uniformly sampling a bit $b \in \{0, 1\}$ and $\boldsymbol{i}, \boldsymbol{j} \leftarrow [m]^{n'}$ with distinct coordinates, and returning $G_n(b; \boldsymbol{i}; \boldsymbol{j})$.

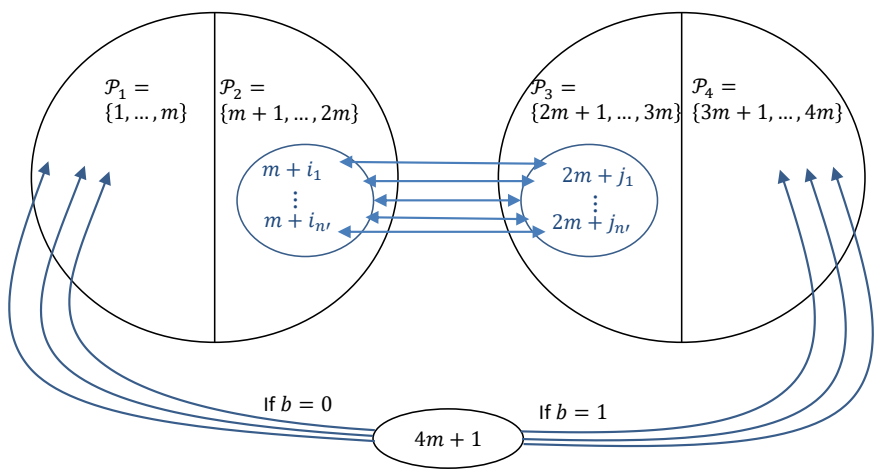

Figure 8: A graph $G_n(b; \boldsymbol{i}; \boldsymbol{j})$ with $n = 4m + 1$ nodes in support of the distribution $\mathcal{D}_{\mathsf{cut}}(n, c)$.

Theorem 6.2. *Let $n \in \mathbb{N}$, let $\beta < 1/4$ and $c > 1$ be constants, and let f be an efficiently computable n-party function. Then, $\mathcal{F}_{\mathsf{sfe}}^{f}$ can be securely realized in a distributional-topology-hiding manner with respect to $\mathcal{D}_{\mathsf{cut}}(n, c)$ with statistical security tolerating an adaptive, semi-honest, computationally unbounded βn-adversary.*

To prove Theorem 6.2, we construct a protocol $\pi_{\mathsf{hide\text{-}cuts}}$ in the $\mathcal{F}_{\mathsf{dist\text{-}graph}}^{\mathcal{D}_{\mathsf{cut}}(n,c)}$-hybrid model that securely realizes $\mathcal{W}_{\mathsf{dist\text{-}graph\text{-}info}}^{\mathcal{D}_{\mathsf{cut}}(n,c)}(\mathcal{F}_{\mathsf{sfe}}^{f})$ (see Figure 12). More specifically, the protocol is defined in a hybrid model with the additional ideal functionalities $\mathcal{F}_{\mathsf{share\text{-}to\text{-}committee}}$, $\mathcal{F}_{\mathsf{recon\text{-}compute}}$, and $\mathcal{F}_{\mathsf{out\text{-}dist}}$ (all functionalities are explained and formally defined in Section 6.1.1). These functionalities need *not* be defined and realized in a topology-hiding manner, since each such functionality will be called by a pre-defined subsets of parties that forms a clique in the communication graph, and so they can be instantiated using a "standard" MPC protocol such as BGW.

In Lemma 6.3 (below) we prove that the protocol $\pi_{\mathsf{hide\text{-}cuts}}$ securely realizes $\mathcal{F}_{\mathsf{sfe}}^{f}$ in a distributional-topology-hiding manner with respect to $\mathcal{D}_{\mathsf{cut}}(n, c)$. We start by defining the ideal functionalities that are used to define the protocol.

6.1.1 Ideal Functionalities used in the Construction

The share-to-committee functionality. In the share-to-committee m-party functionality, $\mathcal{F}_{\mathsf{share\text{-}to\text{-}committee}}$, every party $P_i \in \{P_1, \ldots, P_{2m}\}$ sends his input $x_i \in \{0,1\}^*$, a share s_i (that can be the empty string), and a bit $b_i \in \{0,1\}$ indicating whether P_i has a neighbor in $\{P_{2m+1}, \ldots, P_{3m}\}$. The functionality first tries to reconstruct the value x_{4m+1} from the shares $s_1, \ldots, s_m$. Next, each party secret shares its input value x_i and sends the shares to the parties with $b_i = 1$. The formal description of the functionality can be found in Figure 9.

The functionality $\mathcal{F}_{\mathsf{share\text{-}to\text{-}committee}}$

The $2m$-party functionality $\mathcal{F}_{\mathsf{share\text{-}to\text{-}committee}}$ is parametrized by an integer n' and proceeds with parties $\{P_1, \ldots, P_{2m}\}$ as follows.

1. Every party P_i sends a triplet of values (x_i, s_i, b_i) as its input, where x_i is the actual input value, s_i is potentially a share, and b_i is a bit. Let $\mathcal{C} = \{i \mid b_i = 1\}$. If $|\mathcal{C}| \neq n'$ abort.
2. If the shares $s_1, \ldots, s_m$ are not the empty string, compute $x_{4m+1} = \bigoplus_{i \in [m]} s_i$; otherwise set $x_{4m+1} = \epsilon$ to be the empty string.
3. For every $i \in [2m] \cup \{4m+1\}$, sample uniformly distributed $s_i^1, \ldots, s_i^{n'}$ conditioned on $x_i = \bigoplus_{j \in [n']} s_i^j$.
4. Denote $\mathcal{C} = \{i_1, \ldots, i_{n'}\}$. For every $i_j \in \mathcal{C}$, set $\boldsymbol{s}_{i_j} = (s_1^j, \ldots, s_{2m}^j, s_{4m+1}^j)$.
5. For every $i_j \in \mathcal{C}$, set the output of P_{i_j} to be $\boldsymbol{s}_{i_j}$ (other parties don't get an output).

Figure 9: The share-to-committee functionality

The reconstruct-and-compute functionality. The reconstruct-and-compute functionality, $\mathcal{F}_{\mathsf{recon\text{-}compute}}$, is a $2m$-party functionality. Denote the party-set by $\{P_{2m+1}, \ldots, P_{4m}\}$. Every party P_{2m+i} has an input value $x_{2m+i} \in \{0,1\}^*$, and additional values consisting of shares of $(x_1, \ldots, x_{2m}, x_{4m+1})$. The functionality starts by using the additional inputs to reconstruct

$(x_1, \ldots, x_{2m}, x_{4m+1})$. Next, the functionality computes $y = f(x_1, \ldots, x_{4m+1})$ and hands y as the output for every party. The formal description of the functionality can be found in Figure 10.

The functionality $\mathcal{F}_{\text{recon-compute}}$

The $2m$-party functionality $\mathcal{F}_{\text{recon-compute}}$ is parametrized by an integer n' and proceeds with parties $\{P_{2m+1}, \ldots, P_{4m}\}$ as follows.

1. Every party P_{2m+i} (for $i \in [2m]$) sends a pair of values (x_{2m+i}, z_{2m+i}) as its input, where x_{2m+i} is the actual input value, $z_{2m+i} = \boldsymbol{s}_{2m+i}$ for some $i \in [m]$, and potentially $z_{2m+i} = s_i$ for $i \in [2m] \setminus [m]$ (all other values are the empty string ϵ).
2. Let $\mathcal{C}_2 = \{i \in [m] \mid z_{2m+i} \neq \epsilon\}$. If $|\mathcal{C}_2| \neq n'$ then abort. Otherwise, denote $\mathcal{C}_2 = \{i_1 \ldots, i_{n'}\}$. For every $i_j \in \mathcal{C}_2$, let $\boldsymbol{s}_{2m+i_j} = (s_1^j, \ldots, s_{2m}^j, s_{4m+1}^j)$ be the input provided by P_{2m+i_j}.
3. If the inputs $z_{3m+i} = s_i$ are not empty for $i \in [m]$, then compute $x_{4m+1} = \bigoplus_{i \in [m]} s_i$. Otherwise compute $x_{4m+1} = \bigoplus_{j \in [n']} s_{4m+1}^j$.
4. For every $i \in [2m]$, reconstruct $x_i = \bigoplus_{j \in [n']} s_i^j$.
5. Compute $y = f(x_1, \ldots, x_{4m+1})$.
6. Output y to every P_{2m+i} for $i \in [2m]$.

Figure 10: The reconstruct-and-compute functionality

The output-distribution functionality. The $2m$-party output-distribution functionality receives input values from (some) of the parties and sends one of them as output to all the parties (looking ahead, in the protocol there will be a single input value). The formal description of the functionality can be found in Figure 11.

The functionality $\mathcal{F}_{\text{out-dist}}$

The $2m$-party functionality $\mathcal{F}_{\text{out-dist}}$ proceeds with parties $\{P_1, \ldots, P_{2m}\}$ as follows.

1. Every party P_i, gives (a potentially empty) input value y_i.
2. Let i be the minimal value such that $y_i \neq \epsilon$. Denote $y = y_i$.
3. Output y to every P_i.

Figure 11: The output-distribution functionality

6.1.2 The protocol

We now describe the protocol $\pi_{\text{hide-cuts}}$ and prove its security.

Lemma 6.3. *Protocol $\pi_{\text{hide-cuts}}$ UC-realizes the wrapped functionality $\mathcal{W}^{\mathcal{D}_{\text{cut}}(n,c)}_{\text{dist-graph-info}}(\mathcal{F}^f_{\text{sfe}})$ in the $(\mathcal{F}^{\mathcal{D}_{\text{cut}}(n,c)}_{\text{dist-graph}}, \mathcal{F}_{\text{share-to-committee}}, \mathcal{F}_{\text{recon-compute}}, \mathcal{F}_{\text{out-dist}})$-hybrid model tolerating an adaptive, semi-honest, computationally unbounded βn-adversary, for any constant $\beta < 1/4$.*

The proof of Lemma 6.3 can be found in Appendix B.

6.2 Impossibility in the Classical THC Model

The protocol $\pi_{\mathsf{hide\text{-}cuts}}$ was defined in the weaker distributional-THC model. To justify the weaker model, we show that a similar result cannot be achieved in the stronger (classical) THC model. The reason is that according to this model (Definition 2.1) the environment, who chooses the communication graph, knows exactly which parties are on the cut and can corrupt them. This means that without relying on cryptographic assumptions or some correlated-randomness setup phase, two honest parties from opposite sides of the cut cannot communicate privately [27].

Protocol $\pi_{\mathsf{hide\text{-}cuts}}$

- **Hybrid Model:** The protocol is defined in the $(\mathcal{F}_{\mathsf{dist\text{-}graph}}^{\mathcal{D}_{\mathsf{cut}}(n,c)}, \mathcal{F}_{\mathsf{share\text{-}to\text{-}committee}}, \mathcal{F}_{\mathsf{recon\text{-}compute}}, \mathcal{F}_{\mathsf{out\text{-}dist}})$-hybrid model.
- **Common Input:** A partition of the party-set $\mathcal{P}_1 = \{1, \ldots, m\}$, $\mathcal{P}_2 = \{m+1, \ldots, 2m\}$, $\mathcal{P}_3 = \{2m+1, \ldots, 3m\}$, $\mathcal{P}_4 = \{3m+1, \ldots, 4m\}$, and $\mathcal{P}_5 = \{4m+1\}$.
- **Private Input:** Every party P_i, for $i \in [n]$, has private input $x_i \in \{0,1\}^*$.
- **The Protocol:**

1. Every party P_i sends an initialization input to $\mathcal{F}_{\mathsf{dist\text{-}graph}}^{\mathcal{D}_{\mathsf{cut}}(n,c)}$ and receives the neighbor-set $\mathcal{N}_G[i]$.
2. Party P_{4m+1} samples random $s_1, \ldots, s_m$ conditioned on $x_{4m+1} = \bigoplus_{i \in [m]} s_i$ and sends one share to each of its neighbors (either in $\mathcal{P}_1$ or in $\mathcal{P}_4$).
3. Every party $P_i \in \mathcal{P}_1 \cup \mathcal{P}_2$ sets the bit $b_i = 1$ if he has a neighbor in $\mathcal{P}_3$, and $b_i = 0$ otherwise. In addition, if P_i did not receive a value s_i from P_{4n+1} in Step 2 he sets $s_i = \epsilon$. The parties in $\mathcal{P}_1 \cup \mathcal{P}_2$ invoke $\mathcal{F}_{\mathsf{share\text{-}to\text{-}committee}}$, where every $P_i \in \mathcal{P}_1 \cup \mathcal{P}_2$ sends input (x_i, s_i, b_i). Every $P_i = P_{i_j}$ (for some $j \in [n']$) with $b_{i_j} = 1$ receives back output consisting of a vector $\boldsymbol{s}_{i_j} = (s_1^j, \ldots, s_{2m}^j, s_{4m+1}^j)$.
4. Every party P_i with $b_i = 1$ sends the value received $\boldsymbol{s}_i$ to his neighbor in $\mathcal{P}_3$ (via $\mathcal{F}_{\mathsf{dist\text{-}graph}}^{\mathcal{D}_{\mathsf{cut}}(n,c)}$).
5. If a party $P_{2m+i} \in \mathcal{P}_3 \cup \mathcal{P}_4$ has a neighbor in $\mathcal{P}_2$ he sets z_{2m+i} to be the value received in Step 4, If the party received a value s_i from P_{4n+1} in Step 2 he sets $z_{2m+i} = s_i$; otherwise set $z_{2m+i} = \epsilon$. The parties in $\mathcal{P}_3 \cup \mathcal{P}_4$ invoke $\mathcal{F}_{\mathsf{recon\text{-}compute}}$, where $P_{2m+i} \in \mathcal{P}_3 \cup \mathcal{P}_4$ sends input (x_{2m+i}, z_{2m+i}). Every party in $\mathcal{P}_3 \cup \mathcal{P}_4$ receives back output y.
6. If a party $P_{2m+i} \in \mathcal{P}_3$ has a neighbor in $\mathcal{P}_2$, he sends y to his neighbor (via $\mathcal{F}_{\mathsf{dist\text{-}graph}}^{\mathcal{D}_{\mathsf{cut}}(n,c)}$).
7. The parties in $\mathcal{P}_1 \cup \mathcal{P}_2$ invoke $\mathcal{F}_{\mathsf{out\text{-}dist}}$, where party P_i, with $b_i = 1$, sends the value y he received in Step 6 as his input. Every party in $\mathcal{P}_1 \cup \mathcal{P}_2$ receives output y.
8. Every party that received a value from party P_{4m+1} in Step 2 send y to P_{4m+1}.
9. Every party outputs y and halts.

Figure 12: Hiding low-weight cuts in the $(\mathcal{F}_{\mathsf{share\text{-}to\text{-}committee}}, \mathcal{F}_{\mathsf{recon\text{-}compute}}, \mathcal{F}_{\mathsf{out\text{-}dist}})$-hybrid model

We prove this intuition using our lower bound from Section 3.

Theorem 6.4. *Let $c > 1$ be a constant and let $t = \log^c(n)$. Then, $\mathcal{F}_{\mathsf{bc}}$ cannot be securely computed in a topology-hiding manner with respect to $\mathsf{supp}(\mathcal{D}_{\mathsf{cut}}(n,c))$ tolerating computationally unbounded, semi-honest, static t-adversaries.*

Proof. Let $n = 4m + 1$ and let π be an n-party t-resilient broadcast protocol where party P_{4m+1} is the broadcaster. Let $\boldsymbol{i} = (i_1, \ldots, i_{n'})$ and $\boldsymbol{j} = (j_1, \ldots, j_{n'})$ in $[m]^{n'}$ with distinct coordinates, and consider the following partition of the nodes:

$$\begin{array}{ll} \mathcal{P}_1 = \{4m+1\} & \mathcal{P}_2 = \{1, \ldots, 2m\} \setminus \{i_1, \ldots, i_{n'}\}, \\ \mathcal{P}_3 = \{i_1, \ldots, i_{n'}\}, & \mathcal{P}_4 = \{2m+1, \ldots, 4m\}. \end{array}$$

For $b \in \{0, 1\}$, consider the graph $\tilde{G}_b = G(b; \boldsymbol{i}; \boldsymbol{j}) \in \mathsf{supp}(\mathcal{D}_{\mathsf{cut}}(n, c))$. By definition, it holds that

- In $\tilde{G}_0$: there are no edges $(i, j) \in \mathcal{P}_1 \times \mathcal{P}_3$, or $(i, j) \in \mathcal{P}_2 \times \mathcal{P}_4$, or $(i, j) \in \mathcal{P}_1 \times \mathcal{P}_4$,
- In $\tilde{G}_1$: there are no edges $(i, j) \in \mathcal{P}_1 \times \mathcal{P}_3$, or $(i, j) \in \mathcal{P}_2 \times \mathcal{P}_4$, or $(i, j) \in \mathcal{P}_1 \times \mathcal{P}_2$.

Since $t = |\mathcal{P}_3|$, by Corollary 3.4 there is no THB protocol with respect to $\mathcal{G}$ tolerating semi-honest, static t-adversaries with information-theoretic security. □

7 Sequential Composition of Low-Locality MPC

To motivate the definition of distributional THC, we show its effectiveness with respect to adaptively secure MPC with low locality. Specifically, we consider the protocol of Chandran et al. [18] that maintains sublinear locality per party, however, offers limited capabilities with respect to sequential composition. In this section, we show that the ability to adjust the communication in the *hidden-graph model* from [18] to *distributional topology-hiding* communication, would allow to maintain sublinear locality for polynomially many executions. We emphasize that we do not present a new feasibility result, as we do not know how to implement the topology-hiding communication with adaptive security for the required graph distribution. Instead, the results in this section should be treated as a motivation for the definition.

Low locality with adaptive security. We start with a brief overview of the protocol from [18]. The starting point is any adaptively secure MPC protocol in the honest-majority setting, e.g., [9, 23, 24, 16, 10, 22]. The first step is to replace any broadcast call in the MPC protocol with a broadcast protocol, so that the MPC protocol will run purely over secure pairwise point-to-point channels. This can be done in the PKI model with a polylog blowup using a simultaneous-termination broadcast protocol [36] or while maintaining the same expected running time using a probabilistic-termination protocol [21, 20]. The second step is to use adaptively secure encryption (possibly with erasures [4]) to replace secure channels with authenticated channels.

The third step is to replace every round of the pairwise point-to-point protocol by a *reliable message transmission* (RMT) protocol that supports sublinear locality and adaptive corruptions. The idea in [18] is to use a fresh communication graph for each RMT protocol. Every graph is a randomly generated Erdős-Rényi graph $G(n, p)$, where every edge appears with some fixed probability p that guarantees sublinear degree and polylog diameter with overwhelming probability (in n). Chandran et al. [18] introduced the *hidden-graph model*, where a trusted party samples the graph and gives each party its local neighbor-set, and showed two elegant techniques for instantiating the trusted party in a distributed manner. The first technique, for sampling $G(n, p)$ with $p = (\log^c(n))/n$ (for some constant $c > 1$), assumes a secret-key infrastructure and ensures polylog degree. The second technique, for sampling $G(n, p)$ with $p = (\log^c(n))/\sqrt{n}$, works in the plain model and ensures $\tilde{O}(\sqrt{n})$ degree.

Clearly, to support a sublinear degree with adaptive corruptions, authenticated and private channels do not suffice, as it is crucial to keep the mere fact that two honest parties communicate hidden from the eyes of the adversary. Otherwise, the adversary would be able to see which parties receive messages from some honest party and immediately corrupt them, thus isolating that party. To resolve this issue, Chandran et al. [18] defined their RMT protocol in the *hidden-channels* model, where the adversary is unaware of honest-to-honest communication (see further discussion in [13]).

The RMT protocol over the Erdős-Rényi graph is a simple flooding protocol, and as such is *not* topology hiding (e.g., two corrupt parties can learn their distance by a straightforward timing attack).[2] Due to this limitation, each graph can be used only for a *single* communication round of the original MPC protocol. As a result, the protocol of Chandran et al. [18] only supports bounded sequential composition (in particular, the first construction will not maintain polylog locality after $\log^{\omega(1)}(n)$ sequential executions, and the second will not maintain $\tilde{O}(\sqrt{n})$ locality after $\sqrt{n} \cdot \log^{\omega(1)}(n)$ executions).

For the remaining of this section, we will consider information-theoretic security in the plain model (without any setup assumptions). In this setting, [18] achieves $\tilde{O}(\sqrt{n})$ locality for circuits of $\tilde{O}(\sqrt{n})$ depth by compiling the semi-honest version of BGW [9]. Note that in this setting, if the adversary manages to isolate an honest party, i.e., corrupt all of its neighbors, then privacy is violated.

Sequential composition via THC. Intuitively, one might hope that if the RMT protocol would be topology hiding, the *same* Erdős-Rényi graph could be used multiple times even facing adaptive corruptions. This intuition makes sense since if the topology is kept hidden, the adversary will not be able to identify the local neighbor-sets of honest parties and isolate them. However, when using the classical definition of topology-hiding computation from [42] (which is the basis for all prior work on THC [35, 1, 2, 3, 40]) the underlying communication graph is chosen by the environment, that can use this knowledge and instruct the adversary to corrupt parties in a way that will isolate honest parties. Although the simulator will be able to corrupt the entire neighbor-set also in the ideal world, this will not give him sufficient advantage to simulate isolation of parties in the real world. We note that this phenomena is not related to adaptive security, since even in the static-corruption setting, the environment that chooses the graph and the corrupted-set in a correlated way will gain distinguishing advantage.

This situation can highlight the benefit of distributional topology-hiding computation. Recall that the first step of [18] is defined over secure pairwise point-to-point channels. This can be captured by the *parallel secure message transmission* functionality $\mathcal{F}_{\mathsf{psmt}}$, as defined in [21, 20], where every party P_i sends an input $(m_{i\to 1}, \ldots, m_{i\to n})$ and receives as output $(m_{1\to i}, \ldots, m_{n\to i})$; since the adversary is *rushing*, he learns as leakage all messages sent to corrupted parties (i.e., $m_{j\to i}$ with a corrupted i) before choosing the corrupted parties inputs.

Let $\mathcal{D}_{\mathsf{ER}}(n,p)$ be the Erdős-Rényi distribution over graphs with n nodes where every edge appears with probability p. When executing any adaptively secure protocol defined over secure point-to-point channels in the $\mathcal{W}^{\mathcal{D}_{\mathsf{ER}}(n,p)}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F}_{\mathsf{psmt}})$-hybrid model, every party learns his own neighbor-set (just like in classical THC); however, the environment only knows the *distribution* of the graphs during the course of the computation. In particular, the environment can only corrupt parties as a function of this distribution, the neighbor-sets of all corrupted parties up to that point, and

[2] In fact, since the topology of the graph is revealed during the protocol execution, [18] required atomic data erasures as part of the send operation to avoid subtle adaptive attacks.

whatever is leaked from the MPC protocol. Therefore, the *same* graph can be reused and with overwhelming probability the environment will not be able to isolate honest parties.

For an integer q, denote by $\mathcal{F}^q_{\mathsf{psmt}}$ the reactive functionality the invokes $\mathcal{F}_{\mathsf{psmt}}$ sequentially q times. Using the distributed sampling of Erdős-Rényi graphs from [18], we obtain the following lemma.

Lemma 7.1. *Assume that $\mathcal{W}^{\mathcal{D}_{\mathsf{ER}}(n,p)}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F}_{\mathsf{psmt}})$ can be t-securely realized with adaptive semi-honest security in the $\mathcal{F}^{\mathcal{D}_{\mathsf{ER}}(n,p)}_{\mathsf{dist\text{-}graph}}$-hybrid model, for $p = (\log^c(n))/\sqrt{n}$ with some constant $c > 1$ and $t < n/2$, by an R-round protocol. Let $q = \mathrm{poly}(n)$. Then, there exists a protocol that t-securely realizes $\mathcal{W}^{\mathcal{D}_{\mathsf{ER}}(n,p)}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F}^q_{\mathsf{psmt}})$ with adaptive semi-honest security in the plain model with hidden communication channels using $\tilde{O}(\sqrt{n})$ locality and $O(R \cdot q)$ rounds.*

Proof (sketch). Let π_{psmt} be a protocol that securely realizes $\mathcal{W}^{\mathcal{D}_{\mathsf{ER}}(n,p)}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F}_{\mathsf{psmt}})$ in the $\mathcal{F}^{\mathcal{D}_{\mathsf{ER}}(n,p)}_{\mathsf{dist\text{-}graph}}$-hybrid model. We denote by π^*_{psmt} the protocol realizing $\mathcal{W}^{\mathcal{D}_{\mathsf{ER}}(n,p)}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F}^q_{\mathsf{psmt}})$, consisting of the following two stages:

- Initially, the parties execute the graph-sampling phase from [18] (that is based on [17]). Every party P_i defines the set $S^{\mathsf{out}}_i \subseteq [n]$ by choosing every element in $[n]$ with probability $p = (\log^c(n))/\sqrt{n}$, and sends to every $j \in S^{\mathsf{out}}_i$ a message. Let S^{in}_i be the set of parties that have sent a message to P_i. The neighbor-set of P_i for the rest of the protocol is $S_i = S^{\mathsf{out}}_i \cap S^{\mathsf{in}}_i$.
- Next, for each of the sequential calls, execute an instance of π_{psmt}, where every party uses the neighbor-set established above, and ignores any message arriving from other parties.

We reduce the security of π^*_{psmt} to that of π_{psmt} in two steps. First, consider a protocol π'_{psmt} that runs the graph-sampling phase and proceed to execute a single instance of π_{psmt}. Clearly, by the security of the sampling protocol from [18], π'_{psmt} securely realizes $\mathcal{W}^{\mathcal{D}_{\mathsf{ER}}(n,p)}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F}_{\mathsf{psmt}})$ under the same assumptions as π_{psmt}.

Second, we reduce the security of π^*_{psmt} to that of π'_{psmt} using a standard hybrid argument. Assume that there exists an environment $\mathcal{Z}^*$ that can distinguish between an execution of π^*_{psmt} with the dummy adversary to q sequential calls to $\mathcal{F}_{\mathsf{psmt}}$ with noticeable probability. We will construct an environment $\mathcal{Z}$ that can distinguish between an execution of π'_{psmt} with the dummy adversary and a call to $\mathcal{F}_{\mathsf{psmt}}$.

Consider $q+1$ hybrids, where in the i'th hybrid, the environment $\mathcal{Z}^*$ is invoked and the first i calls are to an execution of π^*_{psmt} with the dummy adversary and the calls $i+1$ to q are to $\mathcal{W}^{\mathcal{D}_{\mathsf{ER}}(n,p)}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F}^{q-i}_{\mathsf{psmt}})$, with the restriction that for $i > 0$, the neighbor-set of any party who is corrupted during the i'th call remains the same in the graph sampled by $\mathcal{W}^{\mathcal{D}_{\mathsf{ER}}(n,p)}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F}^{q-i}_{\mathsf{psmt}})$. Clearly, the 0'th hybrid corresponds to the ideal model for $\mathcal{W}^{\mathcal{D}_{\mathsf{ER}}(n,p)}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F}^q_{\mathsf{psmt}})$ and the q'th hybrid to an execution of π^*_{psmt} with the dummy adversary. By the assumption that $\mathcal{Z}^*$ has noticeable distinguishing advantage there exists a hybrid that is noticeably far from its adjacent hybrid.

Initially, $\mathcal{Z}$ receives as an advice the integer $r \in [q]$ representing the iteration in which $\mathcal{Z}^*$ gains noticeable advantage (alternatively, a uniform environment can simply guess r with $1/\mathrm{poly}$ probability). Next, $\mathcal{Z}$ invokes $\mathcal{Z}^*$ and simulates the parties running π^*_{psmt} with the dummy adversary during the first $r-1$ calls. In the r'th call, $\mathcal{Z}$ forwards the inputs from $\mathcal{Z}^*$ to the parties running either the real π_{psmt} or the ideal call to $\mathcal{F}_{\mathsf{psmt}}$. For the remaining calls $\mathcal{Z}$ simulates the ideal call to $\mathcal{F}_{\mathsf{psmt}}$. Clearly, if $\mathcal{Z}^*$ has noticeable distinguishing probability, then so does $\mathcal{Z}$. □

The protocol described above realizes multiple calls to secure pairwise channels and so can be used as the communication infrastructure for BGW. In fact, we showed that the same communication network can be used to multiple execution of the BGW protocol. For an integer q, denote by $\mathcal{F}^q_{\mathsf{sfe}}$ the reactive functionality the invokes $\mathcal{F}_{\mathsf{sfe}}$ sequentially q times.

Corollary 7.2. *Assume that $\mathcal{W}^{\mathcal{D}_{\mathsf{ER}}(n,p)}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F}_{\mathsf{psmt}})$ can be t-securely realized with adaptive semi-honest security in the $\mathcal{F}^{\mathcal{D}_{\mathsf{ER}}(n,p)}_{\mathsf{dist\text{-}graph}}$-hybrid model, for $p = (\log^c(n))/\sqrt{n}$ with some constant $c > 1$ and $t < n/2$. Let $q = \mathrm{poly}(n)$. Then, $\mathcal{W}^{\mathcal{D}_{\mathsf{ER}}(n,p)}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F}^q_{\mathsf{sfe}})$ can be t-securely realized with adaptive semi-honest security in the plain model with hidden communication channels and maintain $\tilde{O}(\sqrt{n})$ locality.*

Acknowledgements. We thank Mike Rosulek for his graphical support, and the anonymous reviewers of TCC'19 for useful comments.

M. Ball's research supported by an IBM Research PhD Fellowship. Part of this work was completed while M. Ball was visiting IDC HerzliyaâĂŹs FACT center. M. Ball and T. Malkin's research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA) via Contract No. 2019-1902070006. E. Boyle's research supported by ISF grant 1861/16 and AFOSR Award FA9550-17-1-0069. E. Cohen's research supported by the Northeastern University Cybersecurity and Privacy Institute Post-doctoral fellowship, NSF grant TWC-1664445, NSF grant 1422965, and by the NSF MACS project. This work was supported in part by the Intelligence Advanced Research Project Activity (IARPA) under contract number 2019-19-020700009. T. Moran's research supported by the Bar-Ilan Cyber Center. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, DoI/NBC, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

Bibliography

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

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

[3] M. Ball, E. Boyle, T. Malkin, and T. Moran. Exploring the boundaries of topology-hiding computation. In *Advances in Cryptology – EUROCRYPT 2018, part III*, pages 294–325, 2018.

[4] D. Beaver and S. Haber. Cryptographic protocols provably secure against dynamic adversaries. In *Advances in Cryptology – EUROCRYPT '92*, pages 307–323, 1992.

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

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

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

[8] A. Beimel, A. Gabizon, Y. Ishai, E. Kushilevitz, S. Meldgaard, and A. Paskin-Cherniavsky. Non-interactive secure multiparty computation. In *Advances in Cryptology – CRYPTO 2014, part II*, pages 387–404, 2014.

[9] M. Ben-Or, S. Goldwasser, and A. 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.

[10] F. Benhamouda, H. Lin, A. Polychroniadou, and M. Venkitasubramaniam. Two-round adaptively secure multiparty computation from standard assumptions. In *Proceedings of the 16th Theory of Cryptography Conference, TCC 2018, part I*, pages 175–205, 2018.

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

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

[13] E. Boyle, R. Cohen, D. Data, and P. Hubáček. Must the communication graph of MPC protocols be an expander? In *Advances in Cryptology – CRYPTO 2018, part III*, pages 243–272, 2018.

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

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

[16] R. Canetti, O. Poburinnaya, and M. Venkitasubramaniam. Equivocating Yao: constant-round adaptively secure multiparty computation in the plain model. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC)*, pages 497–509, 2017.

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

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

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

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

[21] R. Cohen, S. Coretti, J. A. Garay, and V. Zikas. Probabilistic termination and composability of cryptographic protocols. *Journal of Cryptology*, 32(3):690–741, 2019.

[22] R. Cohen, A. Shelat, and D. Wichs. Adaptively secure MPC with sublinear communication complexity. In *CRYPTO*, 2019. (to appear).

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

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

[25] I. Damgård, P. Meyer, and D. Tschudi. Information-theoretic topology-hiding computation with setup. 2019. URL http://perso.ens-lyon.fr/pierre.meyer/docs/m2.pierre.meyer.pdf.

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

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

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

[29] M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. In *Proceedings of the 23th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 59–70, 1985.

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

[31] S. D. Gordon, T. Malkin, M. Rosulek, and H. Wee. Multi-party computation of polynomials and branching programs without simultaneous interaction. In *Advances in Cryptology – EUROCRYPT 2013*, pages 575–591, 2013.

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

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

[34] M. Hinkelmann and A. Jakoby. Communications in unknown networks: Preserving the secret of topology. *Theoretical Computer Science*, 384(2-3):184–200, 2007.

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

[36] J. Katz and C. Koo. On expected constant-round protocols for Byzantine agreement. In *Advances in Cryptology – CRYPTO 2006*, pages 445–462, 2006.

[37] J. Katz, U. Maurer, B. Tackmann, and V. Zikas. Universally composable synchronous computation. In *Proceedings of the 10th Theory of Cryptography Conference, TCC 2013*, pages 477–498, 2013.

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

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

[40] R. LaVigne, C. L. Zhang, U. Maurer, T. Moran, M. Mularczyk, and D. Tschudi. Topology-hiding computation beyond semi-honest adversaries. In *Proceedings of the 16th Theory of Cryptography Conference, TCC 2018, part II*, pages 3–35, 2018.

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

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

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

[44] A. C. Yao. Protocols for secure computations (extended abstract). In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 160–164, 1982.

A Preliminaries (Cont'd)

A.1 UC Framework

We present a highly informal overview of the UC framework and refer the reader to [15] for further details. The framework is based on the real/ideal paradigm for arguing about the security of a protocol.

The real model. An execution of a protocol π in the real model consists of n PPT *interactive Turing machines* (ITMs) $P_1, \ldots, P_n$ representing the parties, along with two additional ITMs: an *adversary* $\mathcal{A}$, describing the behavior of the corrupted parties and an *environment* $\mathcal{Z}$, representing the external network environment in which the protocol operates. The environment gives inputs to the honest parties, receives their outputs, and can communicate with the adversary at any point during the execution. It is known that security against the *dummy* adversary (that forwards every message he sees to the environment and acts according to its instructions) is sufficient to achieve security against arbitrary adversaries. Throughout, we consider *synchronous* protocols that proceeds in rounds (this can be formally modeled using the $\mathcal{F}_{\mathsf{sync}}$ functionality [15], or using the synchronous framework of [37]) and *semi-honest (passive) security* (where corrupted parties continue following the protocol, but reveal their internal state to the adversary). We will consider both *static* corruptions (where $\mathcal{A}$ chooses the corrupted parties at the onset of the protocol) and *adaptive* corruptions (where $\mathcal{A}$ can dynamically corrupt parties based on information gathered during the computation), and will explicitly mention at any section which type of corruptions are considered. An t-adversary can corrupt up to t parties during the protocol.

The ideal model. A computation in the ideal model consists of n *dummy* parties $\tilde{P}_1, \ldots, \tilde{P}_n$, an *ideal-process adversary* (simulator) $\mathcal{S}$, an *environment* $\mathcal{Z}$, and an *ideal functionality* $\mathcal{F}$. As in the real model, the environment gives inputs to the honest (dummy) parties, receives their outputs, and can communicate with the ideal-process adversary at any point during the execution. The dummy parties act as channels between the environment and the ideal functionality, meaning that they send the inputs received from $\mathcal{Z}$ to $\mathcal{F}$ and vice-versa. The ideal functionality $\mathcal{F}$ defines the desired behaviour of the computation. $\mathcal{F}$ receives the inputs from the dummy parties, executes the desired computation and sends the output to the parties. The ideal-process adversary does not see the communication between the parties and the ideal functionality, however, $\mathcal{S}$ can corrupt dummy parties (statically or dynamically) and may communicate with $\mathcal{F}$ according to its specification.

Security definition. We present the definition for adaptive and semi-honest adversaries. Adjusting the definition to other types of adversaries (e.g., static corruptions) is straightforward.

We say that a protocol π UC-realizes (with computational security) an ideal functionality $\mathcal{F}$ in the presence of adaptive semi-honest t-adversaries, if for any PPT adaptive semi-honest t-adversary $\mathcal{A}$ and any PPT environment $\mathcal{Z}$, there exists a PPT ideal-process t-adversary $\mathcal{S}$ such that the output distribution of $\mathcal{Z}$ in the ideal-model computation of $\mathcal{F}$ with $\mathcal{S}$ is *computationally indistinguishable* from its output distribution in the real-model execution of π with $\mathcal{A}$.

We say that a protocol π UC-realizes (with information-theoretic security) an ideal functionality $\mathcal{F}$ if the above holds even for computationally unbounded $\mathcal{A}$, $\mathcal{Z}$, and $\mathcal{S}$. In that case the requirement is for the output distribution of $\mathcal{Z}$ in the ideal-model computation to be *statistically close* to its output distribution in the real-model execution. If the environment's outputs are identically distributed, we say that π UC-realizes $\mathcal{F}$ with perfect security.

The hybrid model. The *$\mathcal{F}$-hybrid model* is a combination of the real and ideal models, it extends the real model with an ideal functionality $\mathcal{F}$. The parties communicate with each other in exactly the same way as in the real model; however, they can also interact with $\mathcal{F}$ as in the ideal model. An important property of the UC framework is that the ideal functionality $\mathcal{F}$ in a $\mathcal{F}$-hybrid model can be replaced with a protocol that UC-realizes $\mathcal{F}$. The composition theorem of Canetti [15] states the following.

Theorem A.1 ([15], informal)**.** *Let ρ be a protocol that UC-realizes $\mathcal{F}$ in the presence of adaptive semi-honest t-adversaries, and let π be a protocol that UC-realizes $\mathcal{G}$ in the $\mathcal{F}$-hybrid model in the presence of adaptive semi-honest t-adversaries. Then, for any* PPT *adaptive semi-honest t-adversary $\mathcal{A}$ and any* PPT *environment $\mathcal{Z}$, there exists a* PPT *adaptive semi-honest t-adversary $\mathcal{S}$ in the $\mathcal{F}$-hybrid model such that the output distribution of $\mathcal{Z}$ when interacting with the protocol π and $\mathcal{S}$ is computationally indistinguishable from its output distribution when interacting with the protocol π^ρ (where every call to $\mathcal{F}$ is replaced by an execution of ρ) and $\mathcal{A}$ in the real model.*

B Adaptively Secure Distributional THC with Hidden Sublinear Cuts (Cont'd)

We now present the proof of Lemma 6.3.

Proof. Let $\mathcal{A}$ be an adaptive, semi-honest adversary attacking $\pi_{\mathsf{hide\text{-}cuts}}$ in the aforementioned hybrid model. We will construct an ideal-process adversary $\mathcal{S}$, interacting with the ideal function-

ality $\mathcal{W}^{\mathcal{D}_{\mathsf{cut}}(n,c)}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F}^f_{\mathsf{sfe}})$ and with ideal (dummy) parties $\tilde{P}_1, \ldots, \tilde{P}_n$, such that no environment can distinguish between $\mathcal{S}$ and $\mathcal{A}$. To simplify notations, we denote the wrapped functionality $\mathcal{W}^{\mathcal{D}_{\mathsf{cut}}(n,c)}_{\mathsf{dist\text{-}graph\text{-}info}}(\mathcal{F}^f_{\mathsf{sfe}})$ simply as $\mathcal{W}(\mathcal{F}_{\mathsf{sfe}})$ for the rest of the proof.

To define $\mathcal{S}$, we first explain how to simulate the protocol towards the adversary, and later how to respond to dynamic corruption requests. Denote by $\mathcal{I}$ the set of corrupted parties, initially set to $\emptyset$. This is a dynamic set that gets updated during the simulation whenever a party gets corrupted. If at any point during the simulation, $\mathcal{S}$ receives a termination message from $\mathcal{W}(\mathcal{F}_{\mathsf{sfe}})$ (indicating the environment asked to stop the computation and learn the graph) $\mathcal{S}$ stops the simulation of the protocol and only responds to corruption requests.

Simulating the protocol.

- To simulate Step 1, $\mathcal{S}$ receives the initialization input from $\mathcal{A}$ for every corrupted party P_i, forwards the messages to $\mathcal{W}(\mathcal{F}_{\mathsf{sfe}})$, gets neighbors set $\mathcal{N}_G[i]$ for each corrupted party, and returns it to $\mathcal{A}$.
- To simulate Step 2, if $4m+1 \in \mathcal{I}$, simulate receiving messages s_i by his honest neighbors. Otherwise, for every corrupted party P_i with $4m+1 \in \mathcal{N}_G[i]$, send a random $\tilde{s}_i$ from P_{4m+1}.
- To simulate Step 3, $\mathcal{S}$ receives from $\mathcal{A}$ on behalf of every corrupted $P_i \in \mathcal{P}_1 \cap \mathcal{I}$ the input/share/indicator triplet (x_i, s_i, b_i). For every corrupted P_i with $b_i = 1$ (i.e., $\mathcal{N}_G[i] \cap \mathcal{P}_3 \neq \emptyset$), the simulator returns a vector of $2m+1$ random shares $\tilde{\boldsymbol{s}}_i = (\tilde{s}_{i,1}, \ldots, \tilde{s}_{i,2m}, \tilde{s}_{i,4m+1})$.
- To simulate Step 4, $\mathcal{S}$ receives from $\mathcal{A}$ on behalf of every corrupted $P_i \in \mathcal{P}_2 \cap \mathcal{I}$ with $\mathcal{N}_G[i] \cap (\mathcal{P}_3 \setminus \mathcal{I}) \neq \emptyset$ the vector $\tilde{\boldsymbol{s}}_i$ (i.e., a corrupted $P_i \in \mathcal{P}_2$ with an honest neighbor in $\mathcal{P}_3$). For every corrupted P_{2m+i} with $\mathcal{N}_G[2m+i] \cap (\mathcal{P}_2 \setminus \mathcal{I}) \neq \emptyset$ (i.e., a corrupted $P_{2m+i} \in \mathcal{P}_3$ with an honest neighbor $P_{i'} \in \mathcal{P}_2$), $\mathcal{S}$ sends to $\mathcal{A}$ a vector of $2m+1$ random shares $\tilde{\boldsymbol{s}}_{i'} = (\tilde{s}_{i',1}, \ldots, \tilde{s}_{i',2m}, \tilde{s}_{i',4m+1})$.
- To simulate Step 5, $\mathcal{S}$ receives from $\mathcal{A}$ on behalf of every corrupted $P_{2m+i} \in (\mathcal{P}_3 \cup \mathcal{P}_4) \cap \mathcal{I}$ the pair (x_{2m+i}, z_{2m+i}). The simulator forwards the values $\{x_i\}_{i \in \mathcal{I}}$ to $\mathcal{W}(\mathcal{F}_{\mathsf{sfe}})$ and receives back the output y. Next, $\mathcal{S}$ returns y for every corrupted P_{2m+i}.
- To simulate Step 6, $\mathcal{S}$ receives from $\mathcal{A}$ on behalf of every corrupted P_{2m+i} with $\mathcal{N}_G[2m+i] \cap (\mathcal{P}_2 \setminus \mathcal{I}) \neq \emptyset$ the value y. For every corrupted P_i with $\mathcal{N}_G[i] \cap (\mathcal{P}_3 \setminus \mathcal{I}) \neq \emptyset$ (i.e., a corrupted $P_i \in \mathcal{P}_2 \cap \mathcal{I}$ with an honest neighbor in $\mathcal{P}_3$), $\mathcal{S}$ sends to $\mathcal{A}$ the value y.
- To simulate Step 7, $\mathcal{S}$ receives from $\mathcal{A}$ on behalf of every corrupted P_i with $\mathcal{N}_G[i] \cap \mathcal{P}_3 \neq \emptyset$ the value y, and sends y to every corrupted party $P_i \in (\mathcal{P}_1 \cup \mathcal{P}_2) \cap \mathcal{I}$.
- To simulate Step 8, if $4m+1 \in \mathcal{I}$, simulate sending y to P_{4m+1} from every honest party P_i for $i \in \mathcal{N}_G[4m+1] \setminus \mathcal{I}$.

Simulating corruption requests. Upon a corruption request of a party P_i, the simulator $\mathcal{S}$ corrupts the (dummy) party $\tilde{P}_i$, learns its input x_i, and proceeds based on the timing of the corruption request as follows:

- Before Step 1: there is nothing to simulate.

- During Step 1 (before Step 2): $\mathcal{S}$ gets from $\mathcal{W}(\mathcal{F}_{\mathsf{sfe}})$ the neighbor-set $\mathcal{N}_G[i]$ for P_i.

- During Step 2 (before Step 3): if $4m+1 \notin \mathcal{I}$ and $\mathcal{N}_G[i] \cap \{4m+1\} \neq \emptyset$, sample a random $\tilde{s}_i$ as the message P_i received from P_{4m+1}. If $i = 4m+1$, sample a random $\tilde{s}_j$ as the message from P_{4m+1} to P_j, for every $j \in \mathcal{N}_G[4m+1] \setminus \mathcal{I}$, conditioned on $x_{4m+1} = \bigoplus_{j\in[m]} \tilde{s}_j$. (If the value $\tilde{s}_j$ has already been set in the simulation of Step 5, keep it.)

- During Step 3 (before Step 4): $\mathcal{S}$ proceeds as in the previous case. In addition, if $i \in \mathcal{P}_1 \cup \mathcal{P}_2$, $\mathcal{S}$ writes $(x_i, \tilde{s}_i, b_i)$ as P_i's message to $\mathcal{F}_{\mathsf{share\text{-}to\text{-}committee}}$, where $b_i = 1$ if $\mathcal{N}_G[i] \cap \mathcal{P}_3 \neq \emptyset$. If $b_i = 1$, set the output from $\mathcal{F}_{\mathsf{share\text{-}to\text{-}committee}}$ to be a vector of $2m+1$ random shares $\tilde{\boldsymbol{s}}_i = (\tilde{s}_{i,1}, \ldots, \tilde{s}_{i,2m}, \tilde{s}_{i,4m+1})$. (If P_i's neighbor in $\mathcal{P}_3$ is corrupted and his message from Step 4 has already been simulated, set $\tilde{\boldsymbol{s}}_i$ according to this value.)

- During Step 4 (before Step 5): $\mathcal{S}$ proceeds as in the previous case. In addition, if $P_i \in \mathcal{P}_2$ and $b_i = 1$, set $\tilde{\boldsymbol{s}}_i$ as the message to its neighbor in $\mathcal{P}_3$. If $P_i \in \mathcal{P}_3$ and $\mathcal{N}_G[i] \cap \mathcal{P}_2 \neq \emptyset$, set $\tilde{\boldsymbol{s}}_{i'}$ as the message received from its neighbor $P_{i'}$ in $\mathcal{P}_2$, where if $P_{i'}$ is corrupted $\tilde{\boldsymbol{s}}_{i'}$ has been set before (during the simulation) and if $P_{i'}$ is honest, set $\tilde{\boldsymbol{s}}_{i'}$ to be a vector of $2m+1$ random shares $\tilde{\boldsymbol{s}}_{i'} = (\tilde{s}_{i',1}, \ldots, \tilde{s}_{i',2m}, \tilde{s}_{i',4m+1})$.

- During Step 5 (before Step 6): $\mathcal{S}$ proceeds as in the previous case. In addition, if $P_i \in \mathcal{P}_3 \cup \mathcal{P}_4$ set (x_i, z_i) as the input to $\mathcal{F}_{\mathsf{recon\text{-}compute}}$, where if $\mathcal{N}_G[i] \cap \mathcal{P}_2 \neq \emptyset$, set $z_i = \tilde{\boldsymbol{s}}_{i'}$ to be the message simulated in Step 4. If $i \in P_4$ and $4m+1 \in \mathcal{N}_G[i]$, set $z_i = \tilde{s}_i$ as simulated in Step 2, or to a random value if it hasn't been set. Set the output from $\mathcal{F}_{\mathsf{recon\text{-}compute}}$ to be y (note that by the time $\mathcal{S}$ simulates Step 5, it has learned the output value).

- During Step 6 (before Step 7): $\mathcal{S}$ proceeds as in the previous case. In addition, if $P_i \in \mathcal{P}_2$ and $b_i = 1$, set y as the message received from its neighbor in $\mathcal{P}_3$. If $P_i \in \mathcal{P}_3$ and $\mathcal{N}_G[i] \cap \mathcal{P}_2 \neq \emptyset$, set y as the message sent to its neighbor in $\mathcal{P}_2$.

- During Step 7: (before Step 8): $\mathcal{S}$ proceeds as in the previous case. In addition, if $P_i \in \mathcal{P}_1 \cup \mathcal{P}_2$ set y as the output from $\mathcal{F}_{\mathsf{out\text{-}dist}}$, and if $b_i = 1$, set y as the input to $\mathcal{F}_{\mathsf{out\text{-}dist}}$.

- During (and after) Step 8: $\mathcal{S}$ proceeds as in the previous case. In addition, if $P_i \in \mathcal{P}_1 \cup \mathcal{P}_2$ set y as the output from $\mathcal{F}_{\mathsf{out\text{-}dist}}$, and if $b_i = 1$, set y as the input to $\mathcal{F}_{\mathsf{out\text{-}dist}}$.

- At any point during the simulation, if $\mathcal{S}$ has received a termination message with the parties' input values, and by corrupting party P_i the cut is revealed (i.e., for $n'-1$ parties in $\mathcal{P}_2$ either they are corrupted and have a neighbor in $\mathcal{P}_3$, or they are honest and have a corrupted neighbor in $\mathcal{P}_3$), then set the vector of $2m+1$ shares $\tilde{\boldsymbol{s}}_i$ in a way that completes all of the previous shares to the input values.

Proving real/ideal indistinguishability. We now turn to prove that the joint output of the honest parties and of the adversary in the ideal and real executions are statistically close. This is done by defining a sequence of hybrid games. The output of each game is the output of the environment.

The game $\text{HYB}^1_{\pi_{\text{hide-cuts}},\mathcal{A},\mathcal{Z}}$**.** In this game, the simulator has access to the internal state of $\mathcal{W}(\mathcal{F}_{\text{sfe}})$: it can see all input values and neighbor-sets, and choose all output values. The simulator emulates the honest parties in the protocol $\pi_{\text{hide-cuts}}$ towards the adversary $\mathcal{A}$ based on their input values and neighbor-sets and chooses the output for each honest party according to its output in the simulation. Clearly, HYB^1 and the execution of $\pi_{\text{hide-cuts}}$ with $\mathcal{A}$ are identically distributed.

The game $\text{HYB}^2_{\pi_{\text{hide-cuts}},\mathcal{A},\mathcal{Z}}$**.** In this game, we modify HYB^1 as follows. Instead of setting the output of every honest party according to his output from the protocol, $\mathcal{S}$ sends the input values of the corrupted parties (i.e., parties that have been corrupted up until Step 5) to $\mathcal{W}(\mathcal{F}_{\text{sfe}})$, and lets the honest parties receive the output from $\mathcal{W}(\mathcal{F}_{\text{sfe}})$.

Claim B.1. $\text{HYB}^1_{\pi_{\text{hide-cuts}},\mathcal{A},\mathcal{Z}}$ *and* $\text{HYB}^2_{\pi_{\text{hide-cuts}},\mathcal{A},\mathcal{Z}}$ *are identically distributed.*

Proof. The proof follows from the perfect correctness of additive sharing and since both $\mathcal{W}(\mathcal{F}_{\text{sfe}})$ and $\mathcal{F}_{\text{recon-compute}}$ compute the same function. □

The game $\text{HYB}^3_{\pi_{\text{hide-cuts}},\mathcal{A},\mathcal{Z}}$**.** In this game, we modify HYB^2 as follows. Instead of running the functionality $\mathcal{F}_{\text{recon-compute}}$, the simulator simulates the functionality receiving the input values, and returns the value y as received from $\mathcal{W}(\mathcal{F}_{\text{sfe}})$.

Claim B.2. $\text{HYB}^2_{\pi_{\text{hide-cuts}},\mathcal{A},\mathcal{Z}}$ *and* $\text{HYB}^3_{\pi_{\text{hide-cuts}},\mathcal{A},\mathcal{Z}}$ *are identically distributed.*

Proof. The proof follows from the same argument as Claim B.1. □

The game $\text{HYB}^4_{\pi_{\text{hide-cuts}},\mathcal{A},\mathcal{Z}}$**.** In this game, we modify HYB^3 as follows. When simulating the protocol, $\mathcal{S}$ does not simulates the honest-to-honest communication in Step 4 and Step 6. Instead,

- If a party $P_2 \in \mathcal{P}_1$ with a neighbor in $\mathcal{P}_3$ gets corrupted, $\mathcal{S}$ writes the vector of shares $\boldsymbol{s}_i$ (as computed by $\mathcal{F}_{\text{share-to-committee}}$) as its message in Step 4 and the message y as received in Step 6.
- If a party $P_i \in \mathcal{P}_3$ with a neighbor $P_{i'} \in \mathcal{P}_2$ gets corrupted, $\mathcal{S}$ writes the vector of shares $\boldsymbol{s}_{i'}$ (as computed by $\mathcal{F}_{\text{share-to-committee}}$) as its received message in Step 4 and the message y as sent in Step 6.

Claim B.3. $\text{HYB}^3_{\pi_{\text{hide-cuts}},\mathcal{A},\mathcal{Z}}$ *and* $\text{HYB}^4_{\pi_{\text{hide-cuts}},\mathcal{A},\mathcal{Z}}$ *are identically distributed.*

Proof. The claim follows since the adversary $\mathcal{A}$ doesn't see honest-to-honest communication and since $\mathcal{F}_{\text{recon-compute}}$ does not leak any information beyond y. □

The game $\text{HYB}^5_{\pi_{\text{hide-cuts}},\mathcal{A},\mathcal{Z}}$**.** In this game, we modify HYB^4 as follows. Instead of running the functionality $\mathcal{F}_{\text{share-to-committee}}$, the simulator simulates the functionality by sending a vector of random shares to every corrupted party $P_i \in \mathcal{P}_2$ with a neighbor in $\mathcal{P}_3$. Upon adaptive corruptions, $\mathcal{S}$ sets the vector of newly corrupted parties and the messages sent across the cut as random shares. If $\mathcal{S}$ received a termination signal and the set of input values from $\mathcal{W}(\mathcal{F}_{\text{sfe}})$, it sets the last share according to the input values.

Claim B.4. $\text{HYB}^4_{\pi_{\text{hide-cuts}},\mathcal{A},\mathcal{Z}}$ *and* $\text{HYB}^5_{\pi_{\text{hide-cuts}},\mathcal{A},\mathcal{Z}}$ *are statistically close.*

Proof. Note that in HYB^4 the output of $\mathcal{F}_{\mathsf{recon\text{-}compute}}$ is independent of the shares. Therefore, changing the shares does not affect correctness. Since the joint view of the shares of any strict subset of parties on the cut is uniformly distributed both in HYB^4 and in HYB^5, the only distinguishing advantage of $\mathcal{Z}$ is to corrupt the entire cut before sending the termination message. However, since the cut has super-logarithmic weight, it follows from Chernoff bound that this distinguishing advantage is negligible. □

The game $\text{HYB}^6_{\pi_{\mathsf{hide\text{-}cuts}},\mathcal{A},\mathcal{Z}}$. In this game, we modify HYB^5 as follows. In Step 2, the shares sent by an honest P_{4m+1} ro corrupt parties are set as random values, and the communication between an honest P_{4m+1} and honest neighbors is not simulated. Upon later corruptions of neighbors of an honest P_{4m+1}, set their received values to be random shares. Upon a corruption of P_{4m+1}, complete the shares according to its input value.

Claim B.5. $\text{HYB}^5_{\pi_{\mathsf{hide\text{-}cuts}},\mathcal{A},\mathcal{Z}}$ *and* $\text{HYB}^6_{\pi_{\mathsf{hide\text{-}cuts}},\mathcal{A},\mathcal{Z}}$ *are identically distributed.*

Proof. This follows by the perfect correctness of additive sharing and because P_{4m+1} has m neighbors, whereas the corruption threshold is $\beta n < m$. □

This concludes the proof of Lemma 6.3 since in HYB^6 the simulator does not need to have access to the internals of $\mathcal{W}(\mathcal{F}_{\mathsf{sfe}})$, and it behaves exactly as the simulator $\mathcal{S}$ for the adversary $\mathcal{A}$; hence, HYB^6 and the ideal-model computation are identically distributed. □