Topology-Hiding Computation

Tal Moran [*] Ilan Orlov [†] Silas Richelson [‡]

January 1, 2015

Abstract

Secure Multi-party Computation (MPC) is one of the foundational achievements of modern cryptography, allowing multiple, distrusting, parties to jointly compute a function of their inputs, while revealing nothing but the output of the function. Following the seminal works of Yao and Goldreich, Micali and Wigderson and Ben-Or, Goldwasser and Wigderson, the study of MPC has expanded to consider a wide variety of questions, including variants in the attack model, underlying assumptions, complexity and composability of the resulting protocols.

One question that appears to have received very little attention, however, is that of MPC over an underlying communication network whose structure is, *in itself*, sensitive information. This question, in addition to being of pure theoretical interest, arises naturally in many contexts: designing privacy-preserving social-networks, private peer-to-peer computations, vehicle-to-vehicle networks and the "internet of things" are some of the examples.

In this paper, we initiate the study of "topology-hiding computation" in the computational setting. We give formal definitions in both simulation-based and indistinguishability-based flavors. We show that, even for fail-stop adversaries, there are some strong impossibility results. Despite this, we show that protocols for topology-hiding computation can be constructed in the semi-honest and fail-stop models, if we somewhat restrict the set of nodes the adversary may corrupt.

[*]Efi Arazi School of Computer Science, IDC Herzliya. Email: talm@idc.ac.il

[†]Efi Arazi School of Computer Science, IDC Herzliya. Email: iorlov@idc.ac.il

[‡]UCLA Department of Computer Science. Email: SiRichel@ucla.edu

1 Introduction

Secure multi party computation (MPC) has occupied a central role in cryptography since its inception in the '80s. The unifying question can be stated simply:

> *Can mutually distrusting parties compute a function of their inputs, while keeping their inputs private?*

Classical feasibility results [26, 18, 4, 23] paved the way for a plenitude of research which has over time simplified, optimized and generalized the original foundational constructions. Some particularly rich lines of work include improving the complexity (round/communication/computation) of MPC protocols (e.g., [3, 15, 12, 11] and many more) and striving to achieve security in the more difficult (but realistic) setting where the adversary may execute many instantiations of the protocol along with other protocols (e.g., [22, 20, 8, 6] and many more).

Common to nearly all prior work, however, is the assumption that the parties are all capable of exchanging messages with one another. That is to say, most work in the MPC literature assumes that the underlying network topology is that of a complete graph. This is unrealistic as incomplete or even sparse networks are much more common in practice. Moreover, the comparably small body of MPC work that deals with incomplete networks concerns itself with the classical goal of hiding the parties' inputs. In light of the growing impact of networking on today's world, this traditional security goal is insufficient. Consider, for example, the graph representing a social network: nodes representing people, edges representing relationships. Most computation on social networks today is performed in a centralized way—Facebook, for example, performs computations on the social network graph to provide popular services (e.g., recommendations that depend on what "similar" people liked). However, in order to provide these services Facebook must "know" the entire graph.

One could imagine wanting to perform such a computation in a *distributed* manner, where each user communicates only with their own friends, without revealing any additional information to third parties (there is clearly wide interest in this type of service—Diaspora*, a project that was expressly started to provide "Facebook-like" functionality in a more privacy-preserving manner [2], raised over $200,000 in 40 days via Kickstarter).

Another motivating example is the recent push by US auto safety regulators towards vehicle-to-vehicle communication [1], which envisions dynamic networks of communicating vehicles; many "global" computations seem to be interesting in this setting (such as analysis of traffic patterns), but leaking information about the structure of this network could have severe privacy implications.

The rise of the "internet of things", connected by mesh networks (networks of nodes that communicate locally with each other) is yet another case in which the topology of the communication network could reveal private information that users would prefer to hide.

It is with such applications in mind that we initiate the study of *topology-hiding MPC* in the computational setting. We consider the fundamental question:

> *Can an MPC protocol computationally hide the underlying network topology?*

1.1 Our Contributions

Formally Defining Topology-Hiding MPC: In keeping with tradition we give both an indistinguishability game-based definition and a simulation-based one. Very briefly, in the game-based definition the adversary corrupts $A \subset V$ and sends two network topologies G_0, G_1 on vertices V. These graphs must be so that the neighborhoods of A are the same. The challenger then picks G_b at random and returns the collective view

of the parties in A resulting from the execution of the protocol on G_b. The adversary outputs b' and wins if $b' = b$. We say a protocol is *secure against chosen topology attack* (IND-CTA−secure) if no PPT adversary can win the above game with probability negligibly greater than if it simply guesses b'.

We then give a simulation-based definition of security using the UC framework. We define an ideal functionality $\mathcal{F}_{\text{graph}}$ and say that a protocol is "topology hiding" if it is UC secure in the $\mathcal{F}_{\text{graph}}$−hybrid model. The functionality $\mathcal{F}_{\text{graph}}$ models a network with private point-to-point links (private in the sense that the adversary does not know the network topology). It receives G as input, and outputs to each party a description of its neighborhood. It then acts as an "ideal communication channel" allowing neighbors to send messages to each other. For more details on $\mathcal{F}_{\text{graph}}$ and the motivations behind our definition see the discussion below. Finally, we relate the two new notions by proving that simulation-based security implies game-based security.

Feasibility of topology-hiding MPC against semi-honest adversary:

Theorem 1. *Assume trapdoor permutations exist. Let G be the underlying network graph and d a bound on the degree of every vertex in G. Then every multiparty functionality may be realized by a topology hiding MPC protocol which is secure against a semi-honest adversary who does not corrupt all parties in any k−neighborhood of the underlying network graph where k is such that* $d^k = \text{poly}(n)$.

We point out that many naturally occurring graphs satisfy $d^D = poly(n)$ where D is the diameter. Examples of such graphs include binary trees, hypercubes, expanders, and generally graphs with relatively high connectivity such as those occurring from social networks. For such graphs theorem 1 is a feasibility result against a general semi-honest adversary.

Impossibility in fail-stop model:

Theorem 2. *There exists a functionality $\mathcal{F}$ and a network graph G such that realizing $\mathcal{F}$ while hiding G is impossible.*

Our proof uses the ability of the adversary to disconnect G with his aborts; we then prove this is inherent.

Sufficient conditions in fail-stop model:

Theorem 3. *Assume TDP exist. Every multiparty functionality may be realized by a topology hiding MPC protocol which is secure against a fail-stop adversary who does not corrupt all parties in any neighborhood of the underlying network graph and who's aborts do not disconnect the graph.*

1.2 Related Work

MPC on Incomplete Network Topologies One line of work which is in exception to the above began with Dolev's paper [13] proving impossibility of Byzantine agreement on incomplete network topologies with too low connectivity. Dwork et. al. [14] coined the term "almost everywhere Byzantine agreement" to be a relaxed variant of Byzantine agreement where agreement is reached by *almost* all of the honest parties. Garay and Ostrovsky [16] used this to achieve almost everywhere (AE) MPC. Recently [9] gave an improved construction of AE Byzantine agreement translating to an improved feasibility result for AE MPC. These works are all in the information theoretic setting. We refer the curious reader to [9] and the references therein for more details.

Another recent line of work is that of Goldwasser et. al. [5] who consider MPC while minimizing the communication *locality*, the number of parties each player must exchange messages with. Their work is in

the cryptographic setting and they give a meaningful upper bound on the locality and overall communication complexity. Their work does not address the notion of hiding the graph. Moreover they employ techniques such as leader election which seem inherently *not* to hide the graph.

Finally, we mention the two classical techniques of *mix-net* and *onion routing*. The mix-net technique introduced by Chaum [10] uses public key encryption to implement a "message transmit" scheme allowing a sender and receiver to using in a message transmit using an additional shuffling mechanism. The onion routing technique [25, 24] and its extensions is a useful technique for anonymous communication over the Internet. Its basic idea is establishing paths of entities called proxies that know the topology in order to transmit massages.

Topology-Hiding MPC: While most of the cryptographic MPC literature disregards the interplay between multiparty computation and networking, the above works give a relatively satisfactory view of the landscape. Hiding the topology of the network in secure computation, on the other hand, is somewhat of a novel goal. The only work in the MPC literature of which we are aware that has considered this question is that of Hinkelmann and Jakoby [19] who focused on the information theoretic setting. Their main observation can be summarized:

> *If vertices v and w are not adjacent in G then P_v cannot send a message to P_w without some intermediate P_z learning that it sits between P_v and P_w.*

They use this observation to prove that any MPC protocol in the information theoretic setting must inherently leak information about G to an adversary. They do, however, prove a nice positive result: given some minimal amount of network information to leak (formalized as a routing table of the network), one can construct an MPC protocol which leaks no further information.

Their work leaves open the interesting possibility that, using cryptographic techniques, one could construct an MPC protocol which (computationally) hides the network topology. In this work we explore this possibility.

Organization of the Rest of the paper The rest of the paper is organized as follows. In section 2 we go over the background and general definitions which are required to understand the technical portions which follow. In section 3 we formally define our new notions of "topology hiding" computation. This includes our game-based and simulation-based definitions as well as a proof that simulation-based security implies game-based security. In section 4 we consider achieving topology-hiding MPC against a semi-honest adversary. Our basic protocol is secure as long as the adversary does not corrupt any whole neighborhoods of the network graph. We then lessen this requirement showing how to transform a protocol which is secure against an adversary who doesn't corrupt an entire $k-$neighborhood into one secure as long as $\mathcal{A}$ does not corrupt any $(k+1)-$neighborhood. This proves theorem 1. In section 5 we consider a fail-stop adversary and give a somewhat complete picture of the landscape in this setting, proving theorem 2 and theorem 3.

2 Preliminaries

2.1 General Definitions

We model a network by a directed graph $G = (V, E)$ that is not fully connected. We consider a system with $m = \text{poly}(n)$ parties, denoted $P_1, \ldots, P_m$. We often implicitly identify V with the set of parties $\{P_1, \ldots, P_m\}$. We consider a static and computationally bounded (PPT) adversary that controls some subset of parties. That is, at the beginning of the protocol, the adversary corrupts a subset of the parties and may instruct them to

deviate from the protocol according to the corruption model. Through this work, we consider mostly semi-honest and fail-stop adversaries, though we discuss the implications of our fail-stop impossibility result on the hope of achieving topology-hiding MPC in the malicious model. In addition, we assume that the adversary is rushing; that is, in each round the adversary sees the messages sent by the honest parties before sending the messages of the corrupted parties for this round. For general MPC definitions including descriptions of the adversarial models we consider see [17].

2.2 Definitions of Graph Terms

Let $G = (V, E)$ be an undirected graph. For $v \in V$ we let $N(v) = \{w \in V : (v, w) \in E\}$ denote the *neighborhood of* v by; and similarly, the *closed neighborhood of* v, $\mathrm{N}[v] = N(v) \cup \{v\}$. We sometimes refer to $\mathrm{N}[v]$ as the closed $1-$neighborhood of v, and for $k \geq 1$ we define the $k-$neighborhood of v as

$$N^{k+1}[v] = \bigcup_{w \in N^k(v)} \mathrm{N}[w].$$

2.3 UC Security

We employ the UC model [7] in order to abstract away many of the implementation details and get at the core of our definition. Our protocol for hiding the topology in MPC is local in nature, and our final protocol is the result of composing many local subprotocols together. This motivates the need for using subprotocols which are secure under some form of composition. UC security offers strong composability guarantees and thus is well suited to our setting. One of the appealing features of our definition is that it fits entirely within the existing UC framework, hence the UC composition theorem can be applied directly.

The downside of the UC model is that it requires setup [7] and opponents argue that it is "unrealistic". We have two responses to this. First, we encapsulate our setup into realizing the $\mathcal{F}_{\text{graph}}$ functionality. This functionality (defined formally in the next section) models the underlying communication network and so we think of the setup required in order to realize it as an implementation issue. Second, we point out that our subroutines need only be secure against bounded self-composition in order to obtain stand-alone security in the $\mathcal{F}_{\text{graph}}-$hybrid model, corresponding to a stand-alone variant of topology hiding security. This allows us to instantiate our protocol in the plain model on top of (for example) [20] in order to obtain stand-alone topology hiding MPC.

3 Our Model of Security

3.1 Topology Hiding—The Simulation-Based Definition

In this section, we propose a simulation-based definition for topology hiding computation in the UC framework. Generally, in the UC model, the communication between all parties passes through the environment, so it seems the environment implicitly knows the structure of the underlying communication network. We get around this by working in the $\mathcal{F}_{\text{graph}}-$hybrid model. The $\mathcal{F}_{\text{graph}}$ functionality (shown in Figure 1) takes as input the network graph from a special "graph party" P_{graph} and returns to each other party a description of their neighbors. It then handles communication between parties, acting as an "ideal channel" functionality allowing neighbors to communicate with each other without this communication going through the environment.

In a real-world implementation, $\mathcal{F}_{\text{graph}}$ models the actual communication network; i.e., whenever a protocol specifies a party should send a message to one of its neighbors using $\mathcal{F}_{\text{graph}}$, this corresponds to the real-world party directly sending the message over the underlying communication network.

Participants/Notation:
This functionality involves all the parties $P_1, \ldots, P_m$ and a special graph party P_{graph}.

Initialization Phase:

Inputs: $\mathcal{F}_{\text{graph}}$ waits to receive the graph $G = (V, E)$ from P_{graph}.

Outputs: $\mathcal{F}_{\text{graph}}$ outputs $\text{N}_G[v]$ to each P_v.

Communication Phase:

Inputs: $\mathcal{F}_{\text{graph}}$ receives from a party P_v a destination/data pair (w, m) where $w \in \text{N}(v)$ and m is the message P_v wants to send to P_w.

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

Figure 1: The functionality $\mathcal{F}_{\text{graph}}$.

Since $\mathcal{F}_{\text{graph}}$ provides local information about the graph to all corrupted parties, *any* ideal-world adversary must have access to this information as well (regardless of the functionality we are attempting to implement). To capture this, we define the functionality $\mathcal{F}_{\text{graphInfo}}$, that is identical to $\mathcal{F}_{\text{graph}}$ but contains only the initialization phase. For any functionality $\mathcal{F}$, we define a "composed" functionality $(\mathcal{F}_{\text{graphInfo}} \| \mathcal{F})$ that adds the initialization phase of $\mathcal{F}_{\text{graph}}$ to $\mathcal{F}$. We can now define topology-hiding MPC in the UC framework:

Definition 3.1. *We say that a protocol* Π *securely realizes a functionality* $\mathcal{F}$ hiding topology *if it UC-realizes* $(\mathcal{F}_{graphInfo} \| \mathcal{F})$ *in the* $\mathcal{F}_{graph}$*-hybrid model.*

Note that this definition can also capture protocols that realize functionalities depending on the graph (e.g., find a shortest path between two nodes with the same input, or count the number of triangles in the graph).

3.2 Topology Hiding - The Indistinguishability-based Security Definition

In this section, we propose another definition for topology-hiding security that is not restricted to secure multi-party computation. The definition is formalized using a security game between an adversary $\mathcal{A}$ and a challenger C. In addition, we prove that this definition is implied by the simulation-based definition from subsection 3.1. The basic structure of the game fits several types of adversarial behaviors, e.g., semi-honest, fail-stop, and malicious, thus, we do not emphasize the exact behavior of the adversary during the execution of the protocol.

- Setup: Let $\mathcal{G}$ be a set of graphs. Let Π be a protocol capable of running over any of the communication graphs in $\mathcal{G}$ according to the adversarial model of $\mathcal{A}$ (semi-honest, fail-stop, or malicious). Each P_i gets an input $x_i \in X_i$.

- $\mathcal{A}$ chooses a corrupt subset S, inputs x_j for the corrupted parties $P_j \in S$ and, for $i \in \{0, 1\}$, two graphs $G_i = (V_i, E_i) \in \mathcal{G}$, such that $S \subset V_0 \cap V_1$ and $\text{N}_{G_0}[S] = \text{N}_{G_1}[S]$ (equality of graphs). It outputs $(S; G_0, G_1; \{x_j\})$. If $S \not\subset V_0 \cap V_1$ or if some input x_j is invalid C wins automatically.

- Now C chooses a random $b \in \{0, 1\}$ and runs Π in the communication graph G_b, where each honest P_i gets x_i and each dishonest party gets the input prescribed by $\mathcal{A}$. $\mathcal{A}$ receives the collective view of all parties in S during the protocol execution. [1]

[1] In the semi-honest model, the joint view of the corrupted parties is given to $\mathcal{A}$ by the end of the execution of P_i, while in the active models such as fail-stop and malicious, $\mathcal{A}$ sends instructions during the execution of Π and can deviate from the prescribed protocol.

- Finally $\mathcal{A}$ must output $b' \in \{0,1\}$. If $b' = b$ we say that $\mathcal{A}$ wins the security game. Otherwise $\mathcal{A}$ loses.

Definition 3.2. We say that an MPC protocol Π is *Indistinguishable under Chosen Topology Attack* (*IND-CTA secure*) over $\mathcal{G}$ if for any PPT adversary $\mathcal{A}$ there exists negligible function $\mu(\cdot)$, such that for every n it holds

$$\left|\Pr(\mathcal{A} \text{ wins}) - \frac{1}{2}\right| \leq \mu(n).$$

We prove below that IND-CTA security is weaker than security with respect to the simulation-based security definition (Definition 3.1); thus, our impossibility results (in subsection 5.1) imply impossibility of the simulation based definition as well.

Claim 3.3. *For every functionality $\mathcal{F}$ that does not depend on the communication graph structure, if Π securely realizes $\mathcal{F}$ with topology-hiding security (under Definition 3.1) then Π is IND-CTA secure.*

Proof Sketch. Let Π be a topology-hiding secure-computation protocol with respect to Definition 3.1 and let G_0 and G_1 be two graphs. We consider two specific executions of Π on network topologies G_0 and G_1 with corrupt parties given the same inputs. We define random variables $(\text{HYBRID}_{G_0}, \text{IDEAL}_{G_0})$ and $(\text{HYBRID}_{G_1}, \text{IDEAL}_{G_1})$ as usual. We observe that IDEAL_{G_0} are IDEAL_{G_1} are identically distributed, as in both cases the adversary gets the same final output, in addition to the same set of local closed neighborhoods. It follows that if Π realizes $\mathcal{F}$ with topology hiding security then HYBRID_{G_0} and HYBRID_{G_1} are indistinguishable. It follows that $\mathcal{A}$ cannot win the IND-CTA game with probability that is noticeably better than $1/2$. So Π meets also the IND-CTA security definition.

□

4 Topology Hiding MPC Against a Semi-Honest Adversary

In this section we describe a protocol for topology-hiding MPC against a semi-honest adversary. This construction is the heart of the main positive result in the paper:

Theorem 4. *Let d be a bound on the degree of any vertex in G. Then for every k satisfying $d^k = \text{poly}(n)$, there exists a protocol Π that securely realizes the broadcast functionality hiding topology against a semi-honest adversary $\mathcal{A}$ that does not corrupt all parties in any closed $k-$neighborhood of G.*

Note that this gives us security against a general semi-honest adversary when the graph has constant degree and a logarithmic bound on the diameter (by setting k to be anything larger than the graph diameter). We point out that many natural families of graphs are of this sort, including binary trees, hypercubes, expanders and more. Theorem 1 follows from theorem 4 by standard methods for compiling broadcast into MPC.

4.1 High-Level Protocol Overview of Our Basic Protocol

Below we give a top-down description of our basic broadcast protocol: one that is secure against adversaries that do not corrupt any complete 1-neighborhood in the graph (i.e., in every star there is at least one honest node). This basic protocol can then be used to construct a broadcast protocol that tolerates larger corrupt neighborhoods (more details of this transformation appear in section 4.6).

A Naïve Broadcast Protocol

To understand the motivation for the construction, first consider a naïve broadcast protocol for a single bit:

1. In the first round, the broadcaster sends the broadcast bit b to all of its neighbors. Every other party sends 0 to all of their neighbors.

2. In each successive round, every party computes the OR of all bits received from their neighbors in the previous round, and sends this bit to all neighbors.

After j rounds, every party at distance j or less from the broadcaster will be sending the bit b (this can be easily shown by induction); after $diam(G)$ rounds all parties will agree on the bit b. This protocol realizes the broadcast functionality, but it is not topology-hiding: a party can tell how far it is from the broadcaster by counting the number of rounds until it receives a non-zero bit (assume $b = 1$ for this attack). It can also tell in which direction the broadcaster lies by noting which neighbor first sent a non-zero bit.

Using Local MPC to Hide Topology

Our construction hides the sensitive information by secret-sharing it among the nodes in a local neighborhood. Essentially, our basic protocol replaces each node in the naïve protocol above with a secure computation between the node and all of its direct neighbors in the graph.

In order to communicate a bit between one local neighborhood and another, without revealing the bit to the vertex connecting the two neighborhoods, each local neighborhood generates a public/private key pair, for which the private key is secret-shared between the parties in the neighborhood. The input to each local MPC includes the private key shares. The output to each party is encrypted under the public key of the neighborhood represented by that party (i.e., of which the party is the center node).

Since no local neighborhood is entirely corrupted, the adversary does not learn any of the plaintext bits. In the final round (at which point the broadcast bit has "percolated" to all the neighborhoods in the graph). a secure computation is used to decrypt the bits and output the plaintext to all the parties.

The protocol is formally specified as two separate functionalities, each instantiated using a local secure computation: the KeyGen functionality ($\mathcal{L}_{\text{KeyGen}}$), handles the generation and distribution of the public/private key-pair shares in each local neighborhood, and the "broadcast-helper" functionality ($\mathcal{L}_{\text{bc-helper}}$), handles the encryption/decryption and ORing of the bits. The details of the construction are in section 4.4.

Implementing Local MPC

To implement $\mathcal{L}_{\text{KeyGen}}$ and $\mathcal{L}_{\text{bc-helper}}$, the basic protocol uses a general MPC functionality, $\mathcal{L}_{\text{MPC}}$, that allows the local neighborhoods to perform secure computation protocols (i.e., among parties connected in a star graph). Realizing $\mathcal{L}_{\text{MPC}}$ ammounts to constructing a compiler which transforms a standard MPC protocol which runs on a complete graph into one which runs on a star graph. We achieve this by having players in the star who are not connected pass messages to each other through the center. The messages are encrypted to ensure privacy. One subtle point is that the protocol must not leak how many players are in the local neighborhood, as parties are not supposed to learn the degrees of their neighbors. We sidestep this issue by having the center node "invent" fake nodes so that parties learn only that the degree is at most d, some public upper bound on the degree of any node in G. The functionality $\mathcal{L}_{\text{MPC}}$ is shown in Figure 2.

4.2 Formal Protocol Construction and Proofs of Security

Below, we give the formal protocol definitions and sketch their proofs of security in the UC framework.

Notation

The protocols and functionalities in the remainder of this section involve parties $P_1, \ldots, P_m$ and a special graph party P_{graph} whose role is always simply to pass his input, G, to $\mathcal{F}_{\text{graph}}$. For $v \in V$ we let P_v be the player corresponding to v. Many of these protocols/functionalities begin with a KeyGen phase which uses a public key encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$. Finally, we will make repeated use of "local" MPCs which

are executed by the parties in a local neighborhood of G. We will use repeated parallel executions of local MPCs to realize global functionalities. We reserve the letter $\mathcal{L}$ for local functionalities realized by local MPCs, and use $\mathcal{F}$ for global functionalities. For simplicity when describing a local functionality or local MPC protocol, we will describe only the singular execution running in $\mathrm{N}[v]$ (involving P_v and $\{P_w\}_{w \in N(v)}$), even though the same process is occurring in every closed neighborhood in G simultaneously.

4.3 Realizing $\mathcal{L}_{\mathrm{MPC}}$ in the $\mathcal{F}_{\mathbf{graph}}$−hybrid Model

The local MPC functionality $\mathcal{L}_{\mathrm{MPC}}$ is shown in Figure 2. As we have already mentioned, it is sufficient to securely realize message passing between all parties in $\mathrm{N}[v]$ in the $\mathcal{F}_{\mathrm{graph}}$−hybrid model. This is because, once all parties can send messages to each other, they can simply run their favorite UC secure MPC protocol as if the network topology is that of a complete graph. Note that as we are in the semi-honest model here, UC secure MPC does not require setup. We will use the constant round, protocol of [21], as it is UC secure against a semi-honest adversary (against general adversaries it is bounded concurrent secure).

Graph Entry Phase:

Input: $\mathcal{L}_{\mathrm{MPC}}$ receives the graph G from P_{graph}.

Output: $\mathcal{L}_{\mathrm{MPC}}$ outputs $\mathrm{N}[v]$ to P_v.

MPC Phase:

Input: $\mathcal{L}_{\mathrm{MPC}}$ receives from P_v a d−party protocol Π and input x_w from P_w, for each $w \in \mathrm{N}[v]$.

Computation: $\mathcal{L}_{\mathrm{MPC}}$ simulates Π with inputs $\{x_w\}_w$ obtaining outputs $\{y_w\}_w$.

Output: $\mathcal{L}_{\mathrm{MPC}}$ gives y_w to each P_w.

Figure 2: The functionality $\mathcal{L}_{\mathrm{MPC}}$.

For simplicity we describe the protocol allowing P_w to securely send a message to P_u for $w, u \in \mathrm{N}[v]$.

1. P_u generates a key pair and sends the public key to P_w through P_v;
2. P_w encrypts its message and sends the ciphertext back to P_u through P_v.

Such a protocol naturally extends to allow all parties in $\mathrm{N}[v]$ to exchange messages with each other (as long as P_v invents enough nodes to ensure that his neighbors do not learn his degree, but just the preselected bound d). As we mention above, this is sufficient for securely realizing $\mathcal{L}_{\mathrm{MPC}}$ in the $\mathcal{F}_{\mathrm{graph}}$−hybrid model.

Security of $\mathcal{L}_{\mathrm{MPC}}$. The proof is very simple so we suffice it to briefly describe $\mathcal{S}$, and leave checking that it accurately emulates $\mathcal{A}$'s view in the real world to the reader. Since Π is a UC secure MPC protocol on a complete graph, there exists a simulator $\mathcal{S}'$ who can replicate any adversary $\mathcal{A}$'s real-world view in the ideal world. The only difference between the view $\mathcal{S}'$ outputs and the view we need to output is that we must take into account that our messages are encrypted and passed through P_v. Therefore, $\mathcal{S}$ generates key pairs $\{(\mathrm{pk}_{w,u}, \mathrm{sk}_{w,u})\}_{w,u \in \mathrm{N}[v]}$, where P_w will use $\mathrm{pk}_{w,u}$ to send messages to P_u and computes encryptions of the messages in the view output by $\mathcal{S}$, and distributes them accordingly to the players. Security follows from the security of the encryption scheme.

4.4 The Functionalities $\mathcal{L}_{\mathbf{KeyGen}}$ and $\mathcal{L}_{\mathbf{bc\text{-}helper}}$

The functionality $\mathcal{L}_{\mathrm{MPC}}$ of the previous section is a general functionality that compiles an MPC protocol Π on a complete graph into an analogous one which can be executed by the parties in $\mathrm{N}[v]$, without compromis-

Participants/Notation: This protocol involves players P_v, P_w, P_u, for $w, u \in \mathrm{N}[v]$, and allows P_w to send the message msg to P_u. Let $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ be a public key encryption scheme.

Input: P_v and P_u use no input, P_w uses input msg.

Protocol for Message Passing:

- P_u chooses a key pair $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{Gen}(1^n)$ and sends $(v; (w, \mathrm{pk}))$ to $\mathcal{F}_{\mathrm{graph}}$; P_v receives $(u; (w, \mathrm{pk}))$.
- P_v sends $(w; (u, \mathrm{pk}))$ to $\mathcal{F}_{\mathrm{graph}}$; P_w receives $(v; (u, \mathrm{pk}))$.
- P_w computes encryption $y = \mathrm{Enc}_{\mathrm{pk}}(\mathsf{msg})$ and sends $(v; (u, y))$ to $\mathcal{F}_{\mathrm{graph}}$; P_v receives $(w; (u, y))$.
- P_v sends $(u; (w, y))$ to $\mathcal{F}_{\mathrm{graph}}$; P_u receives $(v; (w, y))$.
- P_u decrypts and outputs $\mathsf{msg} = \mathrm{Dec}_{\mathrm{sk}}(y)$.

Figure 3: The $\mathcal{F}_{\mathrm{graph}}$−hybrid protocol $\Pi_{\text{msg-transmit}}$.

ing the security of Π, and also without leaking any information about the topology. We will be interested in two specific local functionalities, $\mathcal{L}_{\mathrm{KeyGen}}$ and $\mathcal{L}_{\text{bc-helper}}$. These can be securely realized in the $\mathcal{F}_{\mathrm{graph}}$−hybrid model by simply instantiating $\mathcal{L}_{\mathrm{MPC}}$ with two specific MPC protocols.

Recall that our underlying idea is to replace the role of P_v in a usual broadcast protocol with an MPC to be performed by the parties in P_v's neighborhood. This will hide each player's distance from the broadcaster because even though the bit might have been received by P_v's neighborhood, it will not be known to any individual player. Our first functionality, $\mathcal{L}_{\mathrm{KeyGen}}$, is useful towards this end. Intuitively, it generates a key pair $(\mathrm{pk}, \mathrm{sk})$ for the neighborhood $\mathrm{N}[v]$ and gives pk to P_v and distributes secret shares of the secret key among P_v's neighbors. Our second functionality, $\mathcal{L}_{\text{bc-helper}}$ will allow the broadcast bit to spread from neighborhood to neighborhood once the neighborhoods have keys distributed according to $\mathcal{L}_{\mathrm{KeyGen}}$. The functionalities $\mathcal{L}_{\mathrm{KeyGen}}$ and $\mathcal{L}_{\text{bc-helper}}$ are shown in Figure 4 and Figure 5, respectively. Let $\mathcal{L}_{\mathrm{KeyGen}}(v)$ and $\mathcal{L}_{\text{bc-helper}}(v)$ denote the copies of $\mathcal{L}_{\mathrm{KeyGen}}$ and $\mathcal{L}_{\text{bc-helper}}$, respectively, which take place in $\mathrm{N}[v]$

Participants/Notation: Let $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$ be a public key encryption scheme.

Graph Entry Phase: same as in $\mathcal{L}_{\mathrm{MPC}}$

KeyGen Phase:

- $\mathcal{L}_{\mathrm{KeyGen}}$ generates a key pair $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{Gen}(1^n)$.
- $\mathcal{L}_{\mathrm{KeyGen}}$ computes random shares $\{\mathrm{sk}_w\}_{w \in \mathrm{N}[v]}$ such that $\bigoplus_w \mathrm{sk}_w = \mathrm{sk}$.

Output: $\mathcal{L}_{\mathrm{KeyGen}}$ gives outputs $(\mathrm{pk}, \mathrm{sk}_v)$ to P_v, and sk_w to each P_w such that $w \in \mathrm{N}[v]$.

Figure 4: The functionality $\mathcal{L}_{\mathrm{KeyGen}}$.

4.5 Realizing $\mathcal{F}_{\mathbf{broadcast}}$ in $\mathcal{L}_{\mathbf{MPC}}$−hybrid model

Our $\mathcal{L}_{\mathrm{MPC}}$−hybrid protocol for broadcast, $\Pi_{\mathrm{broadcast}}$ uses the ideal functionalities $\mathcal{L}_{\mathrm{KeyGen}}$ and $\mathcal{L}_{\text{bc-helper}}$ described above. As mentioned in the previous section, these functionalities are obtained from $\mathcal{L}_{\mathrm{MPC}}$ by instantiating $\mathcal{L}_{\mathrm{MPC}}$ with specific MPC protocols. A description of $\Pi_{\mathrm{broadcast}}$ is given in Figure 11. Note that $\Pi^r_{\mathrm{broadcast}}$ is correct as long $r > \mathrm{diam}(G)$, the diameter of the network graph G. Our statement and proof of security are below.

Participants/Notation: For $w \in \mathrm{N}[v]$, let pk^w be the public key output to P_w by $\mathcal{L}_{\mathrm{KeyGen}}(w)$. Let sk_w^v denote P_w's share of sk^v (the secret key corresponding to pk^v), given as output by $\mathcal{L}_{\mathrm{KeyGen}}(v)$.

Graph Entry Phase: same as in $\mathcal{L}_{\mathrm{MPC}}$

Main Phase:

Input: $\mathcal{L}_{\text{bc-helper}}$ receives inputs:

- $\alpha_w \in$ {"cipher", "plain"} from each P_w;
- $(\mathrm{pk}^w, \mathrm{sk}_w^v)$ from each P_w;
- encryptions $\{x_w\}_{w \in \mathrm{N}[v]}$ from P_v, where $x_w = \mathrm{Enc}_{\mathrm{pk}^v}(b_w)$ for a bit $b_w \in \{0, 1\}$.

The first input α_w is a tag which determines whether $\mathcal{L}_{\text{bc-helper}}$ outputs ciphertexts or plaintests. If all parties do not agree on α_w, $\mathcal{L}_{\text{bc-helper}}$ halts giving no output.

Computation:

- $\mathcal{L}_{\text{bc-helper}}$ reconstructs the secret key $\mathrm{sk}^v = \bigoplus_{w \in \mathrm{N}[v]} \mathrm{sk}_w^v$;
- $\mathcal{L}_{\text{bc-helper}}$ decrypts the bits $b_w = \mathrm{Dec}_{\mathrm{sk}^v}(x_w)$;
- $\mathcal{L}_{\text{bc-helper}}$ computes $b = \bigvee_{w \in \mathrm{N}[v]} b_w$.

Output:

- If $\alpha_w =$ "cipher" for all $w \in \mathrm{N}[v]$ then $\mathcal{L}_{\text{bc-helper}}$ outputs $y_w = \mathrm{Enc}_{\mathrm{pk}^w}(b)$ to each P_w.
- If $\alpha_w =$ "plain" for all $w \in \mathrm{N}[v]$ then $\mathcal{L}_{\text{bc-helper}}$ outputs b to each P_w.

Figure 5: The functionality $\mathcal{L}_{\text{bc-helper}}$.

Claim 4.1. *The protocol $\Pi^r_{broadcast}$ UC securely realizes $\mathcal{F}_{broadcast}$ in the $\mathcal{F}_{graph}$−hybrid model as long as the network topology graph G is such that*

1. *Diameter(G) < r;*
2. *$\mathcal{A}$ does not corrupt any entire closed neighborhood of G.*

Simulator. Consider a corrupt party P_v. $\mathcal{S}$ simulates P_v's view as follows:

1. **KeyGen:** $\mathcal{S}$ generates $(\mathrm{pk}^v, \mathrm{sk}^v) \leftarrow \mathrm{Gen}(1^n)$. When the parties call $\mathcal{L}_{\mathrm{KeyGen}}$, $\mathcal{S}$ returns pk^v to P_v and random strings r_v^w to each P_w such that $w \in \mathrm{N}[v]$, instead of shares of P_w's secret key.
2. **Main Computation:** As output to each of the first $r - 1$ calls to $\mathcal{L}_{\text{bc-helper}}$, $\mathcal{S}$ gives output $\{x^c_{v,w}\}_{w,c}$ to P_v, where $x^c_{v,w} = \mathrm{Enc}_{\mathrm{pk}^v}(0^n)$ to P_v. To compute the output of the last call of $\mathcal{L}_{\text{bc-helper}}$, $\mathcal{S}$ inputs b_v and all other corrupt parties' input bits to $\mathcal{F}_{\text{broadcast}}$ receiving b^* which it returns to P_v.

Hybrid Argument.

H_0 − This is the real execution of $\Pi^r_{\text{broadcast}}$. Namely, each environment first runs $\mathcal{L}_{\mathrm{KeyGen}}$, after which each P_v has key data $(\mathrm{pk}^v, \{\mathrm{sk}_v^w\}_{w \in \mathrm{N}[v]})$. Then parties enter the loop, running $\mathcal{L}_{\text{bc-helper}}$ r times. Initially, parties enter their secret bit and the key data received from $\mathcal{L}_{\mathrm{KeyGen}}$. In each subsequent call to $\mathcal{L}_{\text{bc-helper}}$, the output from the previous call is also given as input. Finally, P_v receives many copies of the same bit b_v^* as output from the last call to $\mathcal{L}_{\text{bc-helper}}$, and P_v outputs this bit. The view of P_v therefore consists of the following:

Input: P_{graph} inputs the graph G, each P_v inputs a bit $b_v \in \{0,1\}$.

KeyGen: Parties call $\mathcal{L}_{\text{KeyGen}}$ and each P_v receives $\text{N}[v]$ and $(\text{pk}^v, \text{sk}_v^v)$ from $\mathcal{L}_{\text{KeyGen}}(v)$ and skv^w from $\mathcal{L}_{\text{KeyGen}}(w)$ for each $w \in \text{N}[v]$.

Main Computation:

- Each P_v sets $x_{v,w}^0 = \text{Enc}_{\text{pk}^v}(b_v)$ for each $w \in \text{N}[v]$.
- For $c = 1, \ldots, r-1$, parties call $\mathcal{L}_{\text{bc-helper}}$:
  - P_v gives input {"cipher"; $(\text{pk}^v, \text{sk}_v^v)$; $\{x_{v,w}^{c-1}\}_{w \in \text{N}[v]}$} to $\mathcal{L}_{\text{bc-helper}}(v)$;
  - For each $w \in \text{N}[v]$, P_v gives input {"cipher"; $(\text{pk}^v, \text{sk}_v^w)$} to $\mathcal{L}_{\text{bc-helper}}(w)$;
  - P_v receives output $x_{v,w}^c$ from $\mathcal{L}_{\text{bc-helper}}(w)$ for all $w \in \text{N}[v]$.
- Finally, parties call $\mathcal{L}_{\text{bc-helper}}$:
  - P_v gives input {"plain"; $(\text{pk}^v, \text{sk}_v^v)$; $\{x_{v,w}^{r-1}\}_{w \in \text{N}[v]}$} to $\mathcal{L}_{\text{bc-helper}}(v)$;
  - For each $w \in \text{N}[v]$, P_v gives input {"plain"; $(\text{pk}^v, \text{sk}_v^w)$} to $\mathcal{L}_{\text{bc-helper}}(w)$;
  - P_v receives the bit $b_{v,w}^*$ as output from $\mathcal{L}_{\text{bc-helper}}(w)$.

Output: P_v outputs $b_v^* = \bigvee_w b_{v,w}^*$.

Figure 6: The $(\mathcal{L}_{\text{KeyGen}} \| \mathcal{L}_{\text{bc-helper}})$-hybrid protocol $\Pi_{\text{broadcast}}^r$.

1. input $b_v \in \{0,1\}$, output $b_v^* \in \{0,1\}$;
2. key data $(\text{pk}^v, \{(\text{sk}_v^w)\}_{w \in \text{N}[v]})$;
3. encryptions $\{x_{v,w}^c\}_{w \in \text{N}[v]}^{c=0,\ldots,r-1}$.

Let $B \subset V$ be the set of bad parties corrupted by $\mathcal{A}$. The view of the adversary is

$$\left\{\left(b_v, b_v^*; \text{pk}^v, \{\,\text{sk}_v^w\,\}_{w \in \text{N}[v]}; \{x_{v,w}^c\}_{w,c}\right)\right\}_{v \in B}.$$

H_1 – This is the same as the above experiment except the secret key shares are replaced by random strings. The resulting view is

$$\left\{(b_v, b_v^*; \text{pk}^v, \{r_v^w\}_w; \{x_{v,w}^c\}_{w,c})\right\}_{v \in B}.$$

As the secret key sk^v is secret shared among $\text{N}[v]$ using a $\left|\text{N}[v]\right|-\text{out}-\text{of}-\left|\text{N}[v]\right|$ secret sharing scheme, and $\mathcal{A}$ does not corrupt all of $\text{N}[v]$, we have that $H_1 \approx H_0$.

H_2 – This is identical to H_1 except that all of the encryptions $x_{v,w}^c$ are changed to encryptions of 0. The resulting view is exactly the view of the ideal world adversary, and is indistinguishable from the view in H_1 by semantic security of the encryption scheme.

4.6 Allowing for Corruption of Whole Neighborhoods

Our protocol $\Pi_{\text{broadcast}}^r$ from the previous section successfully realizes the broadcast functionality while hiding the topology of the graph so long as $\mathcal{A}$ does not corrupt any entire neighborhood of G. If $\mathcal{A}$ were to corrupt $\text{N}[v]$ for some v, our protocol immediately becomes insecure, as $\mathcal{A}$ would possess all of the shares of sk^v and so could simply decrypt all of the encrypted bits P_v receives and learn when the broadcast bit reaches P_v. In this section, we show how, given a protocol Π that is secure as long as $\mathcal{A}$ does not corrupt all parties in a $k-$neighborhood, one can construct another protocol Π' for the same functionality as Π, but is secure as long as $\mathcal{A}$ does not corrupt an entire $(k+1)-$neighborhood. The round complexity of Π' will

be a constant times the round complexity of Π and so one can only repeat this process logarithmically many times.

The main ideas of this section are essentially the same as those in the previous section; showing that the technique of using local MPC to hide information as it spreads to all parties in the graph is quite general. Like $\Pi_{\text{broadcast}}$, our protocol Π' will be given in the $\mathcal{L}_{\text{MPC}}-$hybrid model, where we will use the ideal functionality $\mathcal{L}_{\text{KeyGen}}$. However, instead of using $\mathcal{L}_{\text{bc-helper}}$, we will use a similar but different local functionality, $\mathcal{L}_\Pi$, shown in Figure 7. Essentially, $\mathcal{L}_\Pi$ allows the role of P_v in Π to be computed using a local MPC by all of the parties in $\mathrm{N}[v]$. Then the protocol Π' uses $\mathcal{L}_\Pi$ to execute Π except that each party's role in Π is computed using local MPC by its local neighborhood in Π'. This ensures that if Π is such that any adversary wishing to attack Π must corrupt an entire $k-$neighborhood, then any adversary wishing to attack Π' must corrupt an entire $(k+1)-$neighborhood.

Participants/Notation: For $w \in \mathrm{N}[v]$, let pk^w be the public key output to P_w by $\mathcal{L}_{\text{KeyGen}}(w)$. Let sk_w^v denote P_w's share of sk^v (the secret key corresponding to pk^v), given as output by $\mathcal{L}_{\text{KeyGen}}(v)$.

Graph Entry Phase: same as in $\mathcal{L}_{\text{MPC}}$

Main Phase:

Input: $\mathcal{L}_\Pi$ receives inputs:

- a round number $c_w \in \{1, \ldots, r\}$ from each $w \in \mathrm{N}[v]$;
- $(\mathrm{pk}^w, \mathrm{sk}_w^v)$ from each P_w such that $w \in \mathrm{N}[v]$;
- an encrypted transcript so far $\hat{T}_v^{c-1} = (x_v, \sigma_v; \{\hat{y}_{v,w}^\ell\}_w^{\ell \le c-1})$ from P_v, where x_v and σ_v are P_v's input and randomness and $\hat{y}_{v,w}^\ell = \mathrm{Enc}_{\mathrm{pk}^v}(y_{v,w}^\ell)$ is an encryption of the message P_w sent to P_v in the $\ell-$th round of Π.

If all parties don't agree on the round number, $\mathcal{L}_\Pi$ halts giving no output.

Computation:

- $\mathcal{L}_\Pi$ reconstructs the secret key $\mathrm{sk}^v = \bigoplus_{w \in \mathrm{N}[v]} \mathrm{sk}_w^v$;
- $\mathcal{L}_\Pi$ decrypts $y_{v,w}^\ell = \mathrm{Dec}_{\mathrm{sk}^v}(\hat{y}_{v,w}^\ell)$ for all $w \in \mathrm{N}[v]$ and $\ell \le c-1$;
- $\mathcal{L}_\Pi$ computes the next message function of Π,

$$F_{v,c}^\Pi\left(x_v, \sigma_v, \{y_{v,w}^\ell\}_{w,\ell}\right) = \begin{cases} \{y_{w,v}^c\}_w, & c \le r-1 \\ z_v, & c = r \end{cases}$$

Output:

- If $c \le r-1$ then each P_w with $w \in \mathrm{N}[v]$ receives $\hat{y}_{w,v}^c = \mathrm{Enc}_{\mathrm{pk}^w}(y_{w,v}^c)$ from $\mathcal{L}_\Pi$.
- If $c = r$ then $\mathcal{L}_\Pi$ outputs z_v to P_v.

Figure 7: The functionality $\mathcal{L}_\Pi$.

Our hybrid protocol Π' is described in Figure 8. Our statement and construction of simulator are below. We leave out the hybrid argument as it is very similar to the one in subsection 4.5

Claim 4.2. *The protocol Π' realizes the same functionality as Π. Moreover if Π realizes the functionality UC securely in the $\mathcal{F}_{graph}-$hybrid model as long as $\mathcal{A}$ does not corrupt an entire $k-$neighborhood of G, then Π' is UC secure in the $\mathcal{F}_{graph}-$hybrid model as long as $\mathcal{A}$ does not corrupt an entire $(k+1)-$neighborhood of G.*

Input: P_{graph} inputs the graph G, each P_v inputs x_v, their input to Π.

KeyGen: Parties call $\mathcal{L}_{\text{KeyGen}}$ and each P_v receives $\text{N}[v]$ and $(\text{pk}^v, \{\text{sk}_v^w\}_{w\in\text{N}[v]})$.

Main Computation:

- P_v initializes $\hat{T}_v^0$ to $(x_v, \sigma_v; \emptyset)$.
- For $c = 1, \ldots, r$, parties call $\mathcal{L}_\Pi$:
  - P_v gives input $\{c;\ (\text{pk}^v, \text{sk}_v^v);\ \hat{T}_v^{c-1}\}$ to $\mathcal{L}_{\text{bc-helper}}(v)$;
  - For each $w \in \text{N}[v]$, P_v gives input $\{c;\ (\text{pk}^v, \text{sk}_v^w)\}$ to $\mathcal{L}_{\text{bc-helper}}(w)$;
  - P_v receives output $\hat{y}_{v,w}^c$ from $\mathcal{L}_{\text{bc-helper}}(w)$ for all $w \in \text{N}[v]$.
  - If $c \le r-1$, P_v updates $\hat{T}_v^c$ to include the messages $\{\hat{y}_{v,w}^c\}_j$ he just received.

Output: When $c = r$, P_v receives z_v from $\mathcal{L}_\Pi(v)$, which it outputs.

Figure 8: The $(\mathcal{L}_{\text{KeyGen}} \| \mathcal{L}_\Pi)$-hybrid protocol Π'.

Simulator. We construct a simulator $\mathcal{S}'$ which will make use of the simulator $\mathcal{S}$ for Π. Consider a corrupt party P_v. $\mathcal{S}'$ simulates P_v's view as follows:

1. **KeyGen:** $\mathcal{S}'$ generates $(\text{pk}^v, \text{sk}^v) \leftarrow \text{Gen}(1^n)$ and random shares $\{\text{sk}_w^v\}_{w\in\text{N}[v]}$ such that $\bigoplus_w \text{sk}_w^v = \text{sk}^v$. When the parties call $\mathcal{L}_{\text{KeyGen}}$, $\mathcal{S}'$ returns $(\text{pk}^v, \text{sk}_v^v)$ to P_v and sk_w^v to P_w.

2. **Main Computation:** In order to simulate P_v's view we consider two cases:

   Case 1$-$(P_v has at least one honest neighbor)**:** In this case $\mathcal{S}'$ simulates P_v's view by replacing all the messages P_v would receive with encryptions of 0^n.

   Case 2$-$(all of $\text{N}[v]$ is corrupt)**:** In this case $\mathcal{A}$ can reconstruct sk^v and so will be able to distinguish if $\mathcal{S}'$ sends encryptions of zero. However, $\mathcal{A}$ does not corrupt an entire $(k+1)-$neighborhood of G which means the set $\{v \in V : \text{N}[v] \text{ is corrupt}\}$ does not contain any $k-$neighborhood. Moreover, since each neighborhood in Π' plays the role of a player in Π, we can simulate the view of such P_v using the simulator $\mathcal{S}$ for Π. Specifically, $\mathcal{S}'$ internally runs $\mathcal{S}$ in order to simulate P_v's view in Π, and encrypts with pk^v to obtain P_v's view in Π'.

5 Topology Hiding MPC Against a Fail-Stop Adversary

In this section we consider a stronger adversarial model: the fail-stop adversary. A party controlled by a fail-stop adversary must follow the honest protocol exactly, except that they may abort if the adversary instructs them to.

We have two main results in this section. In section 5.1 we give a general impossibility result, showing that any protocol that implements even a weak version of the broadcast functionality is not IND-CTA secure against fail-stop adversaries. Our proof crucially relies on the ability of the adversary to disconnect the communication graph by aborting with well-placed corrupt parties. In Section 5.2 we show that this is inherent by transforming our broadcast protocol from the previous section into one which is secure against a fail-stop adversary who does not disconnect the graph with his aborts, and who does not corrupt (even semi-honestly) any $k-$neighborhood. We give a high level overview of our techniques of this section before proceeding to the details.

In section 5.1 we consider a protocol Π realizing the broadcast functionality being executed on a line. The proof of the impossibility result is based on two simple observations. First, if some party aborts early

in the protocol then honest parties' outputs cannot depend on b. Clearly, if P^* aborts before the information about b has reached him, then no information about b will reach the honest parties on the other side of P^*. This means that the outputs of *all* honest parties must be independent of b, otherwise an adversary would be able to corrupt another party P_{det} to act as a detective. Namely, $\mathcal{A}$ will instruct P_{det} to play honestly and based on P_{det}'s output, $\mathcal{A}$ will be able to guess which side of P^* P_{det} is on. Second, if P^* aborts near the end of the protocol then all parties (other than P^*'s neighbors) must ignore this abort and output what they would have output had nobody aborted. Indeed, if P^* aborts with only k rounds remaining in the protocol, then there simply isn't time for honest parties of distance greater than k from P^* to learn of this abort. Therefore, all honest parties' outputs must be independent of the fact that P^* aborted, lest an $\mathcal{A}$ would be able to employ P_{det} to detect whether is within distance k of P^* or not. This difference in honest parties' outputs when P^* aborts early versus late means there is a round i^* such that the output distribution of P_{det} when P^* aborts in round i^* is distinguishable from P_{det}'s output distribution when P^* aborts in round $i^* + 1$. We take advantage of this by having two aborters P_1^* and P_2^* who abort in rounds i^* and $i^* + 1$. We prove that $\mathcal{A}$ will be able to distinguish the cases from when P_{det} is to the left of P_1^* with the case when he is to the right of P_2^* allowing $\mathcal{A}$ to win the IND-CTA game with non-negligible advantage.

In Section 5.2 we modify our broadcast protocol of section 4 to be secure against a fail-stop adversary who does not disconnect the graph with his aborts. The idea is to run the semi-honest protocol $2m - 1$ times. Since the adversary can corrupt and abort with at most $m - 1$ parties we are guaranteed that the majority of the executions have no aborts. We ensure that $\mathcal{A}$ learns nothing from the outputs of the executions with aborts by holding off on giving any output until all $2m - 1$ executions have occurred. Then we use a final local MPC protocol to compute all outputs, select the majority and output this to all parties.

5.1 Impossibility Result

Definition 5.1. *We say that a protocol* Π weakly realizes the broadcast functionality *if* Π *is such that when all parties execute the protocol honestly, all parties output* $\bigvee x_i$ *where* x_i *is* P_i*'s input.*

Note that in weak broadcast, there are no guarantees on the behavior of honest parties if any of the parties deviates from the honest protocol.

Theorem 5. *There does not exist an IND-CTA secure protocol* Π *that weakly realizes the broadcast functionality in the fail-stop model.*

Let G be a line with m vertices. Namely, $G = (V, E)$ with $V = \{P_1, \ldots, P_m\}$ and $E = \{(P_i, P_{i+1})\}_{i=1,\ldots,m-1}$. Let Π be a protocol executed on G that weakly realizes the broadcast functionality where P_1 (the left most node) is the broadcaster (P_1 has input b, and the inputs to all other nodes is 0). Suppose Π has r rounds. We will show that Π cannot be IND-CTA secure.

Claim 5.2. Let $H_{v,b}$ be the event that P_v's output after executing Π matches the broadcast bit b. Let E_i be the event that the first abort occurs in round i. Then either Π is not IND-CTA secure, or there exists a bit $b \in \{0, 1\}$ such that

$$\left|\Pr(H_{v,b}\middle|E_{r-1}) - \Pr(H_{v,b}\middle|E_1)\right| \geq \frac{1}{2} - \text{negl}(n)$$

for all honest P_v whose neighbors do not abort.

Proof. If some P^* aborts during the first round of Π then he disconnects the graph, making it impossible for the parties separated from P_1 to learn about b. These parties' outputs therefore must be independent of b, which implies that there exists a $b \in \{0, 1\}$ such that $\Pr(H_b|E_1) \leq \frac{1}{2}$. If Π is to be IND-CTA secure then it must be that this inequality holds (with possibly a negligible error) for all honest parties. Otherwise an

adversary could use the correlation between b and a party's output to deduce that this party is in the same connected component as P_1.

Formally, consider a fail-stop adversary $\mathcal{A}$ who corrupts three parties: the broadcaster P_1, aborter $P^* = P_{\lfloor \frac{m}{2} \rfloor}$ and detective P_{det}. $\mathcal{A}$ then submits (G_0, G_1, S), to the challenger where $G_0 = G$ and G_1 is constructed from G by exchanging the labels of the nodes (P_3, P_4, P_5) with (P_{m-2}, P_{m-1}, P_m). That is, in G_1, the nodes P_3, P_4, P_5 appear at the end of the line. We call $P_{\text{det}} = P_4$ the "detective" node. The set S consists of the nodes P_1, P^* and P_{det}. Note that $\mathcal{A}$'s neighborhoods are the same in G_0 and G_1 (for $m \geq 12$).

$\mathcal{A}$ instructs P^* to abort during the first round and observes P_{det}'s output. Since P_{m-1}'s output must be independent of b, if P_4's output depends in a non-negligible way on b, this will translate into an advantage for $\mathcal{A}$ in the CTA game.

Finally, note that $\Pr(H_{v,b} \big| E_{r-1}) = \Pr(H_{v,b} \big|$ no aborts$) = 1$ for all P_v which are not neighbors of P^*. The claim follows. □

Proof of Theorem 5. It follows from Claim 5.2 that there exists a pair $(i^*, b) \in \{1, \ldots, r\} \times \{0, 1\}$ such that

$$\left| \Pr(H_{v,b} \big| E_{i^*}) - \Pr(H_{v,b} \big| E_{i^*+1}) \right| \geq \frac{1}{2r} - \text{negl}(n). \tag{1}$$

for all honest P_v who do not have an aborting neighbor. Furthermore, assume without loss of generality that $\Pr(H_{v,b} \big| E_{i^*}) > \Pr(H_{v,b} \big| E_{i^*+1})$. We construct a fail-stop adversary $\mathcal{A}$ who can leverage this fact to win the CTA game with non-negligible advantage.

Our adversary $\mathcal{A}$ corrupts four parties: the broadcaster P_1, two aborters $(P^*_{\text{L}}, P^*_{\text{R}}) = (P_{\lfloor \frac{m}{2} \rfloor - 1}, P_{\lfloor \frac{m}{2} \rfloor + 1})$, and the detective P_{det}. $\mathcal{A}$ then submits (G_0, G_1, S) to the challenger where $G_0 = G$, G_1 is constructed from G by exchanging (P_3, P_4, P_5) with (P_{m-2}, P_{m-1}, P_m) and $S = \{P_1, P^*_{\text{L}}, P^*_{\text{R}}, P_{\text{det}} = P_4\}$. These graphs are shown in Figure 9. Note that these adversary structures have identical neighborhoods for $m \geq 14$.

Now $\mathcal{A}$ guesses $(i^*, b) \in \{1, \ldots, r\} \times \{0, 1\}$. With non-negligible probability, (i^*, b) is such that inequality (1) is satisfied. $\mathcal{A}$ gives b as input to P_1 and instructs P^*_{L} to abort on round i^*, P^*_{R} to abort on round $i^* + 1$. Notice that since the two aborting parties are at distance 2 from each other, the information about P^*_{L}'s abort does not reach P^*_{R} by the time he aborts one round later. Therefore, the information about P^*_{L}'s abort does not reach any of the parties to the right of P^*_{R} at any point during the protocol. This means that if G_0 was chosen by the challenger, P_{det}'s output will be consistent with E_{i^*} whereas if G_1 was chosen, P_{det}'s output will be consistent with E_{i^*+1}. $\mathcal{A}$ concludes by comparing P_{det}'s output bit to the broadcast bit b. If they are equal, $\mathcal{A}$ sends 0 to the challenger, otherwise he sends 1. The noticeable difference in output distributions ensured by i^* translates to a noticeable advantage for $\mathcal{A}$.

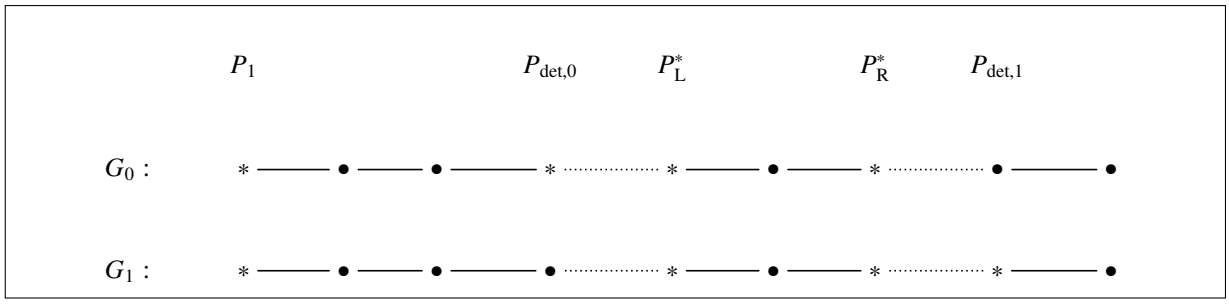

Figure 9: Graphs used by $\mathcal{A}$ in proof of theorem 5.

□

5.2 Feasibility Result

In this section we show how to modify the broadcast protocol from section 4 which is secure against a semi-honest adversary who doesn't corrupt any $k-$neighborhood into one which is secure against a fail stop adversary, who doesn't corrupt a $k-$neighborhood and whose aborts don't disconnect the graph. For simplicity in describing the protocol we take k to be 1, and point out what would have to be changed to accomodate $k = O(\log n)$. Our protocol, $\Pi_{\text{fstop-bcast}}$, is shown in Figure 11, it makes use of another local functionality $\mathcal{L}_{\text{maj}}$ shown in Figure 12. $\Pi_{\text{fstop-bcast}}$ realizes the fail-stop broadcast functionality shown in Figure 10

The main idea of our protocol is to run the semihonest protocol many times, ensuring that the majority of the executions contain no aborts. The correctness of the semi-honest protocol guarantees that these executions with no aborts result in correct output. However, to prevent parties from learning which executions contain an abort, we change our protocol so that the outputs of the individual executions are given to the parties in encrypted form, and only after all of them have been completed, N[v] runs a local MPC realizing $\mathcal{L}_{\text{maj}}$ to compute the majority of the outputs it has received. This ensures that all parties will receive the correct output.

We comment that an adversary who aborts during the final majority MPC will stop each local neighborhood he is a part of from being able to reconstruct the output. However, since $\mathcal{A}$ knows which parties he is connected to, forcing them to output $\perp$ does not tell him anything about the graph. If a corrupt party aborts during the main part of the protocol then he ruins the local MPCs running in all closed neighborhoods to which he belongs, but does not affect anything else. Neighbors of the aborting party will get an output of $\perp$ for the current run of the semi-honest protocol, but now that this corrupt party has aborted he cannot ruin any other repititions. The majority at the end will ensure that this abort does not upset the output of $\Pi_{\text{fstop-bcast}}$.

Finally, we comment that if the local MPC run at the end to compute the majority is executed by the parties in N[v], then the resulting protocol will only be secure if $\mathcal{A}$ does not corrupt any closed neighborhood in G. However, we can compile a protocol which is secure against a semi-honest $\mathcal{A}$ who does not corrupt all parties in a $k-$neighborhood of G into one that is secure against a fail-stop $\mathcal{A}$ simply by having the last MPC be computed by all the parties in the $k-$neighborhood of v. This involves implementing a message passing protocol between all parties in the $k-$neighborhood of v which may be done similarly to $\Pi_{\text{msg-transmit}}$.

Input: $\mathcal{F}_{\text{bc-failstop}}$ receives inputs:

- the graph G from P_{graph};
- a bit $b_v \in \{0, 1\}$ from each P_v;
- a value $\alpha_v \in \{\mathsf{complete}, \mathsf{abort\_early}, \mathsf{abort\_late}\}$ from each corrupt P_v.

Output: $\mathcal{F}_{\text{bc-failstop}}$ gives outputs:

- $\perp$ to all adversarial P_v such that $\alpha_v \in \{\mathsf{abort\_early}, \mathsf{abort\_late}\}$;
- $\perp$ to all honest P_v who are adjacent to an adversarial P_w such that $\alpha_w = \mathsf{abort\_late}$;
- $\bigvee_v b_v$ to all other parties.

Figure 10: The fail-stop broadcast functionality $\mathcal{F}_{\text{bc-failstop}}$.

Input: P_{graph} inputs the graph G, each P_v inputs a bit $b_v \in \{0,1\}$.

KeyGen: Parties call $\mathcal{L}_{\text{KeyGen}}$ and each P_v receives $\mathrm{N}[v]$, $(\mathrm{pk}^v, \mathrm{sk}_v^v)$ from $\mathcal{L}_{\text{KeyGen}}(v)$ and sk_v^w from $\mathcal{L}_{\text{KeyGen}}(w)$ for all $w \in \mathrm{N}[v]$. If a party aborts during this phase then start again, ignoring that party for the remainder of the protocol.

Main Computation For $j = 1, \ldots, 2m-1$ do**:**

- Each P_v sets $\hat{x}_{v,w}^0 = \mathrm{Enc}_{\mathrm{pk}^v}(b_v)$ for each $w \in \mathrm{N}[v]$.
- For $i = 1, \ldots, r$, parties call $\mathcal{L}_{\text{bc-helper}}$:
  - P_v gives input $\left\{\text{“cipher”};\ (\mathrm{pk}^v, \{\mathrm{sk}_v^w\}_w);\ \{\hat{x}_{v,w}^{i-1}\}_w\right\}$;
  - P_v receives output $\{\hat{x}_{v,w}^i\}_w$.
- P_v picks $w \in \mathrm{N}[v]$ randomly and sets $\hat{y}_v^j = \hat{x}_{v,w}^r$.
- If a party aborts during this computation all of his neighbors set $\hat{y}_v^j = \mathrm{Enc}_{\mathrm{pk}^v}(\perp)$ and ignore the aborting party for the remainder of the protocol.

Output:

- Parties call $\mathcal{L}_{\text{maj}}$:
  - P_v gives input $\left\{(\mathrm{pk}^v, \{\mathrm{sk}_v^w\}_w);\ \{\hat{y}_v^j\}_j\right\}$;
  - P_v receives output b_v^*
- Each P_v outputs b_v^*.

Figure 11: The protocol $\Pi_{\text{fstop-bcast}}$ (in the $\mathcal{L}_{\text{MPC}}$−hybrid model).

Graph Entry Phase: same as in $\mathcal{L}_{\text{MPC}}$

Main Phase:

Input: $\mathcal{L}_{\text{maj}}$ receives inputs:

- $(\mathrm{pk}^v, \{\mathrm{sk}_v^w\}_{w \in \mathrm{N}[v]})$ − the data P_v receives from $\mathcal{L}_{\text{maj}}$
- $\{\hat{y}_v^1, \ldots, \hat{y}_j^{2m-1}\}$ − encryptions $\hat{y}_v^j = \mathrm{Enc}_{\mathrm{pk}^v}(b_v^j)$ of values $b_v^j \in \{0,1\} \cup \{\perp\}$

from each P_v.

Computation:

- $\mathcal{L}_{\text{maj}}$ reconstructs the secret key $\mathrm{sk}^v = \bigoplus_{w \in \mathrm{N}[v]} \mathrm{sk}_w^v$ for each $v \in V$.
- For each $v \in V$, $\mathcal{L}_{\text{maj}}$ decrypts the bits $b_v^j = \mathrm{Dec}_{\mathrm{sk}^v}(\hat{y}_v^j)$ for $j = 1, \ldots, 2m-1$.
- $\mathcal{L}_{\text{maj}}$ computes $b_v = \mathrm{MAJ}_j(b_v^j)$, where MAJ is the majority function.

Output: $\mathcal{L}_{\text{maj}}$ gives P_v the output b_v.

Figure 12: The functionality $\mathcal{L}_{\text{maj}}$.

6 Discussion and Open Questions

Malicious Model. The most basic question we leave open in this work is the situation in the malicious model. Clearly, our impossibility results for the fail-stop model also apply here. Our positive results do not carry over, however. This is because a malicious adversary can "pretend" to be connected to an entire graph; in this fake graph, the adversary can corrupt *any* size neighborhood, violating our security assumptions.

General Graphs in the Semi-Honest Model. A second natural question that arises from this work is whether the restriction to graphs of logarithmic diameter is a necessary one, even in the semi-honest model. Does there exist a protocol for topology-hiding secure computation in arbitrary graphs?

Hiding the Identities of Neighbors. Another open problem we leave is whether topology-hiding security can be realized while hiding from P_v the identities of his neighbors. This would involve changing the $\mathcal{F}_{\text{graph}}$ functionality to give as output a local identity for each of the parties in $\mathcal{P}_v$'s neighborhood, and this identity would differ in other closed neighborhoods. One interesting application would be that adversaries in the same local neighborhood would not learn that they are distance 2 from each other. Even our semi-honest protocol revealed this information as parties at distance 2 had to communicate in local MPCs.

References

[1] http://www.its.dot.gov/research/v2v.htm.

[2] http://www.nytimes.com/2010/05/12/nyregion/12about.html.

[3] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols (extended abstract). In *STOC*, pages 503–513. ACM, 1990.

[4] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10. ACM, 1988.

[5] E. Boyle, S. Goldwasser, and S. Tessaro. Communication locality in secure multi-party computation - how to run sublinear algorithms in a distributed setting. In *TCC*, pages 356–376, 2013.

[6] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `http://eprint.iacr.org/2000/067`.

[7] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145. IEEE Computer Society, 2001.

[8] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. Cryptology ePrint Archive, Report 2002/140, 2002. `http://eprint.iacr.org/2002/140`.

[9] N. Chandran, J. A. Garay, and R. Ostrovsky. Edge fault tolerance on sparse networks. In A. Czumaj, K. Mehlhorn, A. M. Pitts, and R. Wattenhofer, editors, *ICALP (2)*, volume 7392 of *Lecture Notes in Computer Science*, pages 452–463. Springer, 2012.

[10] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[11] I. Damgård, Y. Ishai, and M. Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In H. Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 445–465. Springer, 2010.

[12] I. Damgård and J. B. Nielsen. Scalable and unconditionally secure multiparty computation. In A. Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 572–590. Springer, 2007.

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

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

[15] M. K. Franklin and M. Yung. Communication complexity of secure computation (extended abstract). In *STOC*, pages 699–710. ACM, 1992.

[16] J. Garay and R. Ostrovsky. Almost-everywhere secure computation. In N. P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 307–323. Springer, 2008.

[17] O. Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, New York, NY, USA, 2004.

[18] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.

[19] M. Hinkelmann and A. Jakoby. Communications in unknown networks: Preserving the secret of topology. *Theor. Comput. Sci.*, 384(2-3):184–200, 2007.

[20] R. Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In L. Babai, editor, *STOC*, pages 232–241. ACM, 2004.

[21] R. Pass. Bounded-Concurrent Secure Multi-Party Computation with a Dishonest Majority. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, STOC '04, pages 232–241, 2004.

[22] R. Pass and A. Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *FOCS*, pages 404–413. IEEE Computer Society, 2003.

[23] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *STOC*, pages 73–85. ACM, 1989.

[24] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, 1998.

[25] M. K. Reiter and A. D. Rubin. Anonymous web transactions with crowds. *Commun. ACM*, 42(2):32–38, 1999.

[26] A. C.-C. Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164. IEEE, 1982.