Exploring the Boundaries of Topology-Hiding Computation

Marshall Ball[1] ⋆, Elette Boyle[2] ⋆⋆, Tal Malkin[3] ⋆⋆⋆, and Tal Moran[2] †

[1] Columbia University and IDC Herzliya. marshall@cs.columbia.edu
[2] IDC Herzliya, Israel. {elette.boyle,talm}@idc.ac.il
[3] Columbia University, USA. tal@cs.columbia.edu

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. In a line of recent works [Moran, Orlov & Richelson, TCC'15, Hirt et al. CRYPTO'16, Akavia & Moran EUROCRYPT'17, Akavia et al. CRYPTO'17], THC protocols for securely computing any function in the *semi-honest* setting have been constructed. In addition, it was shown by Moran et al. that in the fail-stop setting THC with negligible leakage on the topology is impossible.

In this paper, we further explore the feasibility boundaries of THC.

- We show that even against semi-honest adversaries, topology-hiding broadcast on a small (4-node) graph implies oblivious transfer; in contrast, trivial broadcast protocols exist unconditionally if topology can be revealed.
- We strengthen the lower bound of Moran *et al.*, identifying and extending a relation between the *amount* of leakage on the underlying graph topology that must be revealed in the fail-stop setting, as a function of the number of parties and communication round complexity: Any n-party protocol leaking δ bits for $\delta \in (0, 1]$ must have $\Omega(n/\delta)$ rounds.

We then present THC protocols providing close-to-optimal leakage rates, for *unrestricted graphs* on n nodes against a fail-stop adversary controlling a dishonest majority of the n players. These constitute the first general fail-stop THC protocols. Specifically, for this setting we show:

⋆ Supported in part by the Defense Advanced Research Project Agency (DARPA) and Army Research Office (ARO) under Contract #W911NF-15-C-0236, NSF grants #CNS1445424 and #CCF-1423306, ISF grant no. 1790/13, and the Check Point Institute for Information Security. Any opinions, findings and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency, Army Research Office, the National Science Foundation, or the U.S. Government.

⋆⋆ Supported in part by ISF grant 1861/16, AFOSR Award FA9550-17-1-0069, and ERC Grant no. 307952.

⋆⋆⋆ Supported in part by the Defense Advanced Research Project Agency (DARPA) and Army Research Office (ARO) under Contract #W911NF-15-C-0236, NSF grants #CNS1445424 and #CCF-1423306, and the Leona M. & Harry B. Helmsley Charitable Trust. Any opinions, findings and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency, Army Research Office, the National Science Foundation, or the U.S. Government.

† Supported in part by ISF grant no. 1790/13 and by the Bar-Ilan Cyber-center.

– A THC protocol that leaks at most one bit and requires $O(n^2)$ rounds.
– A THC protocol that leaks at most δ bits for arbitrarily small non-negligible δ, and requires $O(n^3/\delta)$ rounds.

These protocols also achieve full security (with no leakage) for the semi-honest setting. Our protocols are based on one-way functions and a (stateless) *secure hardware box* primitive. This provides a theoretical feasibility result, a heuristic solution in the plain model using general-purpose obfuscation candidates, and a potentially practical approach to THC via commodity hardware such as Intel SGX. Interestingly, even with such hardware, proving security requires sophisticated simulation techniques.

1 Introduction

Secure multiparty computation (MPC) is a fundamental research area in cryptography. Seminal results, initiated in the 1980s [56,33,8,18], and leading to a rich field of research which is still flourishing, proved that mutually distrustful parties can compute arbitrary functions of their input securely in many settings. Various adversarial models, computational assumptions, complexity measures, and execution environments have been studied in the literature. However, until recently, almost the entire MPC literature assumed the participants are connected via a *complete graph*, allowing any two players to communicate with each other.

Recently, Moran, Orlov and Richelson [52] initiated the study of *topology-hiding computation (THC)*. THC addresses settings where the network communication graph may be partial, and the network topology itself is sensitive information to keep hidden. Here, the goal is to allow parties who see only their immediate neighborhood, to securely compute arbitrary functions (that may depend on their secret inputs and/or on the secret underlying communication graph). In particular, the computation should not reveal any information about the graph topology beyond what is implied by the output.

Topology-hiding computation 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. Examples indicating interest in privacy of the network graph in these application domains include the project diaspora*[1], which aims to provide a distributed social network with privacy as an important goal; works such as [16,54] which try to understand the internal ISP network topology despite the ISP's wish to hide them; and works such as [45,25] that try to protect location privacy in sensor network routing, among others.

There are only a few existing THC constructions, and they focus mostly on the *semi-honest* adversarial setting, where the adversary follows the prescribed protocol. In particular, for the semi-honest setting, the work of Moran et al. [52] achieves THC for network graphs with a *logarithmic diameter* in the number of players, from the assumptions of oblivious transfer (OT) and PKE.[4] Hirt et al. [42] improve these results,

[4] Alternatively, the [52] results can be interpreted as results for arbitrary graphs, but where the adversary is limited in its corruption pattern, and not allowed to corrupt any k-neighborhoods where k depends on the graph.

relying on the DDH assumption, but still requiring the graph to have logarithmic diameter. Akavia and Moran [3] achieve THC for other classes of graphs, in particular graphs with small circumference. Recently, this was extended by Akavia et al. [2] to DDH-based THC for *general* graphs.

In the *fail-stop* setting, where an adversary may abort at any point but otherwise follows the protocol, the only known construction is one from [52], where they achieve THC for a very limited corruption and abort pattern: the adversary is not allowed to corrupt a full neighborhood (even a small one) of any honest party, and not allowed an abort pattern that disconnects the graph. This result is matched with a lower bound, proving that THC in the fail-stop model is *impossible* (the proof utilizes an adversary who disconnects the graph using aborts).

In this paper, we further explore the feasibility boundaries of THC. In the semi-honest model, we study the the minimal required computational assumptions for THC, and in the fail-stop model we study lower and upper bounds on the necessary leakage. All our upper bounds focus on THC for arbitrary graphs with arbitrary corruption patterns (including dishonest majority). The security notion in the fail-stop setting is one of "security with abort", in which the adversary is allowed to abort honest parties after receiving the protocol's output.

1.1 Our Results

We will often describe our results in terms of the special case of *Topology-Hiding Broadcast* (THB), where one party is broadcasting an input to all other parties. We note that all our results apply both to THB and to THC (for arbitrary functionalities). In general, THB can be used to achieve THC for arbitrary functions using standard techniques, and for our upper bound protocols in particular the protocols can be easily changed to directly give THC of any functionality instead of broadcast.

Lower Bounds. We first ask what is the minimal assumption required to achieve THB in the semi-honest model. Our answer is that at the very least, OT is required (and this holds even for small graphs). Specifically, we prove:

> *Theorem (informal): If there exists a 4-party protocol realizing topology-hiding broadcast against a semi-honest adversary, then there exists a protocol for Oblivious Transfer.*

Note that without the topology-hiding requirement, it is trivial to achieve broadcast *unconditionally* in the semi honest case, as well as the fail-stop case with security with abort. Indeed, the trivial protocol (sometimes referred to as "flooding") consists of propagating everything you received from your neighbors in the previous round, and then aborting if there is any inconsistency, for sufficiently many rounds (as many as the diameter of the graph).

We mentioned above the result of [52], who prove that THC in the fail-stop model is impossible to achieve, since any protocol in the fail-stop model must have some *non-negligible* leakage. We next refine their attack to characterize (and amplify) the *amount*

of leakage required, as a function of the number of parties n and communication rounds r of the protocol. We model the leakage of a protocol by means of a leakage oracle $\mathcal{L}$ evaluated on the parties' inputs (including graph topology) made available to the ideal-world simulator, and say that a protocol has $(\delta, \mathcal{L})$-leakage if the simulator only accesses $\mathcal{L}$ with probability δ over its randomness.[5]

In particular, we demonstrate the following:

> *Theorem (informal): For an arbitrary leakage oracle $\mathcal{L}$ (even one which completely reveals all inputs), the existence of r-round, n-party THB with $(\delta, \mathcal{L})$-leakage implies $\delta \in \Omega(n/r)$.*

The theorem holds even if all parties are given oracle access to an arbitrary functionality, as is the case with the secure hardware box assumption mentioned below. This improves over the bound of [52], which corresponds to $\delta \in \Omega(1/r)$ when analyzed in this fashion.

Upper Bounds. We start by noting that a modification of the construction in [52] gives a scheme achieving TH computation in the *semi-honest* setting, for *log-diameter graphs* from OT alone (rather than OT + PKE as in the original work). This matches our lower bound above, showing THC if and only if OT, in the case of low diameter graphs.

Our main upper bound result is a THC construction for arbitrary graph structures and corruptions, in the *fail-stop, dishonest majority* setting, and (since leakage is necessary), with *almost no leakage*.

We have two versions of our scheme. The first is a scheme in $O(n^2)$ rounds (where n is a bound on the number of parties), which leaks at most one bit about the graph topology (i.e., simulatable given a single-output-bit leakage oracle $\mathcal{L}$). The leaked bit is information about whether or not one given party has aborted at a given time in the computation. This information may depend on the graph topology.

We then extend the above to a randomized scheme with arbitrarily small inverse polynomial leakage δ, in $O(n^3/\delta)$ rounds; more specifically, $(\delta, \mathcal{L})$-leakage for single-bit-output oracle $\mathcal{L}$. Here the leakage from $\mathcal{L}$ also consists of information about whether or not one given party has aborted at a given time in the computation. However, roughly speaking, the protocol is designed so that this bit depends on the graph topology only if the adversary has chosen to obtain this information in a specific "lucky" round, chosen at random (and kept hidden during the protocol), and thus happens with low probability.

We also point out that a simpler version of our scheme achieves full security (with no leakage) in the semi-honest model (for arbitrary graphs and arbitrary corruption pattern). Moreover, we leverage our stronger assumption to achieve essentially optimal round complexity in the semi-honest model—the protocol runs in $O(\mathrm{diam}(G))$ rounds (where $\mathrm{diam}(G)$ is a bound on the diameter of the communication graph G) and can directly compute any functionality (any broadcast protocol must have at least $\mathrm{diam}(G)$ rounds, otherwise the information might not reach all of the nodes in the graph). In contrast, the only previous THC protocol for general graphs [2] requires $\Omega(n^3)$ rounds

[5] Interestingly, this formalization is not equivalent to (and slightly weaker than) $\mathcal{L}_\delta$-leakage for respective functionality $\mathcal{L}_\delta$ that provides the output of $\mathcal{L}$ only with probability δ; see the full version for details. (Note that ruling out a weaker notion means a stronger lower bound.)

for a single broadcast; computing more complex functionalities requires composing this with another layer of MPC on top.

Our schemes relies on the existence of one-way functions (OWF), as well as a *secure hardware box*, which is a stateless "black box", or oracle, with a fixed secret program, given to each participant before the protocol begins. We next discuss the meaning and implications of this underlying assumption, but first we summarize our main upper bound results:

> *Theorem (informal): If OWF exist and given a secure hardware box, for any n-node graph G and poly-time computable function f,*
> - *There exists an efficient topology-hiding computation protocol for f against poly-time fail-stop adversaries, which leaks at most one bit of information about G, and requires $O(n^2)$ rounds.*
> - *For any inverse polynomial δ, there exists an efficient topology-hiding computation protocol for f against poly-time fail-stop adversaries, which leaks at most δ bits of information about G, and requires $O(n^3/\delta)$ rounds.*

We remark that the first result gives an n-party, r-round protocol with $(O(n^2/r), \mathcal{L})$-leakage, in comparison to our lower bound that shows impossibility of $(o(n/r), \mathcal{L})$-leakage. Closing this gap is left as an intriguing open problem.

On Secure Hardware Box Assumption. A secure hardware box is an oracle with a fixed, stateless secret program. This bears similarity to the notion of *tamper-proof hardware tokens*, introduced by Katz [46] to achieve UC secure MPC, and used in many followup works in various contexts, both with stateful and stateless tokens (cf. [15,37,20] and references within).

A hardware box is similar to a stateless token, but is incomparable in terms of the strength of the assumption. On one hand, a hardware box is worse, as we assume an honest setup of the box (by a party who does not need to know the topology of the graph, but needs to generate a secret key and embed the right program), while hardware tokens are typically allowed to be generated maliciously (although other notions of secure hardware generated honestly have been considered before, e.g. [43,19]). On the other hand, a hardware box is better, in that, unlike protocols utilizing tokens, it does not need to be passed around during the protocol, and the players do not need to embed their own program in the box: there is a single program that is written to all the boxes before the start of the protocol.

Unlike previous uses of secure hardware in the UC settings (where we know some setup assumption is necessary for security), we do not have reason to believe that strong setup (much less a hardware oracle) is necessary to achieve THC. However, we believe many of the core problems of designing a THC protocol remain even given a secure hardware oracle. For example, the lower bounds on leakage hold even in this setting. In particular, our hardware assumption does not make the solution trivial (in fact, in some senses the proofs become harder, since even a semi-honest adversary may query the oracle "maliciously"). Our hope is that the novel techniques we use in constructing the protocol, and in proving its correctness, will be useful in eventually constructing a

protocol in the standard model. We note that this paradigm is a common one in cryptography: protocols are first constructed using a helpful "hardware' oracle", and then ways are found to replace the hardware assumption with a more standard one. Examples include the ubiquitous "random oracle", but also hardware assumptions much more similar to ours, such as the Signature Card assumption first used to construct Proof-Carrying Data (PCD) schemes [19]. (Signature cards contain a fixed program and a secret key, and can be viewed as a specific instance of our secure hardware assumption.)

Thus, one way to think of our upper bound result is as a step towards a protocol in the standard model.

At the same time, our phrasing of the assumption as "secure hardware" is intentional, and physical hardware may turn out to be the most practical approach to actually implementing a THC protocol. Because our functionality is fixed, stateless, and identical for all parties, our secure hardware box can be instantiated by a wide range of physical implementations, including general-purpose "trusted execution environments", that are becoming widespread in commodity hardware (for example, both ARM (TrustZone) and Intel (SGX) have their own flavors implemented in their CPUs). We discuss a potentially practical approach to THC through the use of SGX secure hardware in the full version of this paper.

Future directions. Our work leaves open many interesting directions to further pursue, such as the following.

- Obtain better constructions for the case of *honest majority*.
- Obtain THC in the fail-stop model from standard cryptographic assumptions. In particular, can THC be achieved from OT alone, matching our lower bound?
- The results for THC in the fail-stop setting are in some ways reminiscent of the results for optimally fair coin tossing. In particular, in both cases there is an impossibility result if no leakage or bias is allowed, and there are lower bounds and upper bounds trading off the amount of leakage with the number of rounds (cf. [22,23,51,24]). It would be interesting to explore whether there is a formal connection between THC and fair coin tossing, and whether such a connection can yield tighter bounds for THC.
- THC with security against malicious adversaries is an obvious open problem, with no prior work addressing it (to the best of our knowledge). Could our results be extended to achieve security in the malicious settings? More generally, could a secure hardware box be useful towards maliciously secure THC?

1.2 Technical Overview

A starting point for our upper bound protocol is the same starting point underlying the previous THB constructions [52,42,3]: Consider the trivial flooding protocol that achieves broadcast with no topology hiding, by propagating the broadcasted bit to all the neighbors repeatedly until it reaches everyone. One problem with this protocol is that the messages received by a node leak a lot of information about the topology of the graph (e.g., the distance to the broadcaster). Previous works mitigate that by encrypting

the communication, and also requiring all nodes to send a bit in every round: the broadcaster sends its bit, and the other nodes send a 0; each node then ORs all the incoming bits, and forwards to its neighbors in the next round. However, this leaves the question of how the bit will be decrypted to obtain the final result. This is where previous works differ in their techniques to address this issue (using nested MPC, homomorphic encryption, ideas inspired by mix-nets or onion routing to allow gradual decryption, etc), and the different techniques imply different limitations on the allowed graph topology (or corruption patterns) that support the solution.

We also begin with the same starting point of trying to implement flooding. We then use the secure hardware box, which will contain a relevant secret key that allows it to process encrypted inputs (partial transcripts propagated from different parts of the graph) and produce an encrypted output to propagate further in the next round, as well as decrypting the output at the end. However, we have several new technical challenges that arise, both because of the fail-stop setting, and because of the existence of the box itself.

First, the fail-stop setting presents a significant challenge (indeed, provably necessitating some leakage). Intuitively, abort behavior by the adversary will influence the behavior of honest parties (e.g., if an honest party is isolated by aborts in their immediate neighborhood, they would not be able to communicate and will have to abort rather than output something; aborting behavior of honest parties can in turn provide information to the adversary about the graph topology). The hardware box will help in checking consistency of partial transcripts, and helping honest parties manage when and how they disclose their plan to output abort at the end of the protocol.

A second source of difficulty stems from having the secure hardware box at the disposal of the adversary. This allows to inject a malicious aspect to the adversary's behavior, even in a fail-stop (or even semi-honest) setting. Indeed, since each player has their own box, and the box is stateless, the adversary can run the boxes with arbitrary inputs, providing different partial transcripts, abort or non-abort behavior, etc, in order to try and learn information about the graph topology. This presents challenges that make the proof of security much more involved and quite subtle.

Overview of our solution. Recall the core source of information leakage is from the abort-or-not values of various parties, as a function of fail-stop aborts caused by the adversary. The first idea of our construction is to limit the amount of leakage to a *single* bit by ensuring that for any fail-stop abort strategy, the abort-or-not value of only a single party will be topology dependent. This is achieved by designating a special "threshold" round T_i for each party: if the party P_i learns of an abort somewhere in the graph before round T_i, he will output abort at the conclusion of the protocol, and if he only learns of an abort after this round he will output the correct bit value. By sufficiently separating these threshold rounds, and leveraging the fact that an abort will travel to all nodes within n rounds (independent of the graph topology), we can guarantee that any given abort structure will either reach before or after the threshold round of a *single* party in a manner dependent on the topology.

Note that in the above, if the threshold rounds T_i are known, then there exists an adversarial strategy which indeed leaks a full bit on the topology. To obtain arbitrarily

small leakage δ, we modify the above protocol by expanding the "zone" of each party into a collection of $O(n/\delta)$ possible threshold rounds. The value of the true threshold for each party is determined (pseudo-)randomly during protocol execution and is hidden from the parties themselves (who see only encrypted state vectors from their respective secure hardware boxes). Because of this, the probability that an adversary will be able to successfully launch a leakage attack on any single party's threshold round will drop by a factor of δ/n; because this attack can be amplified by attacking across several parties' zones, the overall winning probability becomes comparable to δ. Note that such an increase in rounds to gain smaller leakage is to be expected, based on our leakage lower bound.

The more subtle and complex portion of our solution comes in the simulation strategy, in particular for simulating the output of the hardware box on arbitrary local queries by adversarial parties. At a high level, the simulator will maintain a collection of graph structures corresponding to query sequences to the boxes (where outputs from previous box queries are part of input to a later box query), and will identify a specific set of conditions in which a query to the leakage oracle must be made. See below for a more detailed description.

Overview of simulation strategy. Simulation consists almost entirely of answering queries to the hardware box. As intermediate outputs from the box are encrypted, the chief difficulty lies in determining what output to give to queries corresponding to the *final round* of the protocol: either $\perp$ (abort); the broadcast bit; or something else, given partial leakage information about the graph and only the local neighborhood of the corrupted parties.

The simulator uses a data structure to keep track of the relationship between queries to the hardware box and outputs from previous queries. In the real world, this relationship is enforced by the unforgeability of the authenticated encryption scheme. The simulator can use this data structure to determine whether a query is "derived," in part, from 'honest' (simulated) messages, and additionally, what initializations were used for the non-honest parties connected to the node expecting output.

One of the major difficulties is that even a semi-honest adversary can locally query his hardware box in malicious ways: combining new initializations in novel ways with pieces of the honest transcript, or aborting in multiple different patterns. The bulk of the proof is devoted to showing that all of these cases can be simulated.

One key fact utilized in this process is that if the protocol gives any output at all, then all honest nodes must have encrypted states at round n (the maximum diameter of the graph) that contain a complete picture of the graph, inputs, session keys, etc. Therefore, the real hardware box will not give plaintext output if such an honest state is mixed with states in a manner that deviates from the real protocol evaluation significantly.

An added wrinkle is that the hardware box, by virtue of the model, is required to handle a variety of abort sequences. Moreover, the kind of output received after certain abort timing inherently leaks information about the topology. Yet, the simulator must decide output behavior without additional leakage queries. Here again the honest messages will essentially "lock" an adversary into aborts that are "consistent" with the

aborts in the real protocol evaluation. (For example, an adversary can "fast-forward" a node after the nodes output is guaranteed by pretending all of its neighbors aborted.)

The honest messages also aid in replay attacks as they allow the simulator to only consider connected groups of corrupt nodes. If two nodes are separated by honest nodes in the real world, then in their replay attack no new abort information will be transmitted from one to the other if the protocol is replayed in a locally consistent manner (modulo aborts). (If the protocol is not locally consistent no descendent of that query will yield plaintext output.)

Finally, if a query doesn't have any honest ancestors, the simulator can simulate output trivially as it knows all of the initialization information.

In short, the difficulties in the proof come from the fact that output depends on topology and abort structure, and a fail-stop adversary can use his box to essentially simulate malicious runs of the protocol after its completion to attempt to gain more leakage on the topology. However, the simulator can only query the leakage oracle at most once. Accordingly, the specific timing of its query in protocol evaluation is very delicate: if it is too early the adversary can abort other nodes to change output behavior in an unsimulatable manner, if it is too late then the adversary can fast-forward to get output in an unsimulatable manner. Moreover, output behavior must be known for all replay attacks where the simulator has incomplete initialization information (pieces of the honest transcript are used). As a consequence, we are forced to consider elaborate consistency conditions to bind the adversary to a specific evaluation (modulo aborts), and prove that these conditions achieve bind the adversary while still allowing him the freedom to *actively* attack the protocol using the hardware.

1.3 Related Work

We have already discussed above the prior works on topology-hiding computation in the computational setting [52,42,3], which are the most relevant to our work.

Topology-hiding computation was also considered earlier in the *information-theoretic setting*, by Hinkelmann and Jakoby [41]. They provide an impossibility result, proving that any information-theoretic THC protocol leaks information to an adversary (roughly, when two nodes who are not neighbors communicate across the graph, some party will be able to learn that it is on the path between them). They also provide an upper bound, achieving information theoretic THC that leaks a routing table of the network, but no other information about the graph.

There are several other lines of work that are related to communication over incomplete networks, but in different contexts, not with the goal of hiding the topology. For example, a line of work studied the feasibility of reliable communication over (known) incomplete networks (cf. [26,28,27,5,48,10,7,4,14]). More recent lines of work study secure computation with restricted interaction patterns in a few settings, motivated by improving efficiency, latency, scalability, usability, or security. Examples include [39,36,11,13,6,38]. 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. [11,13] use this idea in order to achieve communication locality).

An early use of a hidden communication graph which is selected as a subgraph of an available known larger graph, is in the context of anonymous communication and

protection against traffic analysis. Particularly noteworthy are the mix-net and onion routing techniques ([17,55,53] and many follow up works), which also inspired some of the recent THC techniques.

There is a long line of work related to the use of secure hardware in cryptography, in various flavors with or without assuming honest generation, state, complete tamper proofness, etc. This could be dated back to the notion of oblivious RAM ([34] and many subsequent works). Katz [46] introduced the notion of a hardware token in the context of UC-secure computation, and this notion has been used in many followup works (e.g. [15,37,20] and many others). Variations on the hardware token, where the hardware is generated honestly by a trusted setup, include signature cards [43], trusted smartcards [40], and so called non-local-boxes [9]. The latter are similar to global hardware boxes that are generated honestly and take inputs and output from multiple parties (in contrast to our notion of a hardware box, which is local). Other variations and relaxations include tamper-evident seals [50], one time programs [35], and various works allowing some limited tampering ([44,31] and subsequent works). Finally, there is a line of works using other physical tools to perform cryptographic tasks securely, including [29,30,32]

2 Preliminaries

2.1 Secure Hardware

We model our secure hardware box as an ideal oracle, parameterized by a stateless program Π. The oracle query $O(\Pi)(x)$ returns the value $\Pi(x)$. Our definition is much simpler than the standard secure hardware token definitions, since all parties have access to the same program, and it is stateless—there is no need for a more complex functionality that keeps track of the "physical location" of the token or its internal state.

2.2 Topology Hiding Computation

The work of [52] put forth two formal notions of topology hiding: a simulation-based definition, and a weaker indistinguishability-based definition. In this work, we primarily focus on the simulation-based definition, given below. However, some of our lower bounds apply also to the indistinguishability-based notion.

The definition of [52] works in the $\mathcal{F}_{\text{graph}}$-hybrid model, for $\mathcal{F}_{\text{graph}}$ functionality (shown in Figure 1) that 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.

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

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. (If w is not a neighbor of v, $\mathcal{F}_{\text{graph}}$ does nothing.)
Output: $\mathcal{F}_{\text{graph}}$ gives output (v, m) to P_w indicating that P_v sent the message m to P_v.

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 1 (Topology Hiding (Simulation-Based)). *We say that a protocol Π securely realizes a functionality $\mathcal{F}$* hiding topology *if it securely 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).

2.3 Extended Definitions of THC

We extend the simulation definition of Topology-Hiding Computation beyond the semi-honest model, capturing fail-stop corruptions, and formalizing a measure of leakage of a protocol.

Topology Hiding with Leakage

We consider a weakened notion of topology hiding with partial information leakage. This is modeled by giving the ideal-world simulator access to a reactive functionality leakage oracle $\mathcal{L}$, where the type/amount of leakage revealed by the protocol is captured by the choice of the leakage oracle $\mathcal{L}$. For example, we will say a protocol "leaks a

single bit" about the topology if it is topology hiding for some oracle $\mathcal{L}$ which outputs at most 1 bit throughout the simulation.[6]

Definition 2 (Topology Hiding with $\mathcal{L}$-Leakage). *We say that a protocol Π securely realizes a functionality $\mathcal{F}$* hiding topology with $\mathcal{L}$-leakage *if it realizes $(\mathcal{F}_{graphInfo}||\mathcal{F}||\mathcal{L})$ in the $\mathcal{F}_{graph}$-hybrid model, where $\mathcal{L}$ is treated as an ideal (possibly reactive) functionality which outputs only to corrupt parties.*

Note that the above functionality $(\mathcal{F}_{\text{graphInfo}}||\mathcal{F}||\mathcal{L})$ is not a "well-formed" functionality in the sense of [12], as the output of the functionality depends on the set of corrupt parties. However, this is limited to additional information given to corrupt parties, which does not run into the simple impossibilities mentioned in [12] (indeed, it is easier to securely realize than $(\mathcal{F}_{\text{graphInfo}}||\mathcal{F})$). The definition also extends directly to topology hiding within different adversarial models, by replacing $\mathcal{F}_{\text{graph}}$ with the corresponding functionality (such as $\mathcal{F}_{\text{graph-failstop}}$ for fail-stop adversaries; see below).

It will sometimes be convenient when analyzing lower bounds and considering fractional bits of leakage to consider the following restricted notion of $(\delta, \mathcal{L})$-leakage, for probability $\delta \in [0, 1]$. Loosely, a $(\delta, \mathcal{L})$-leakage simulator is restricted to only utilizing the leakage oracle $\mathcal{L}$ with probability δ over the choice of its random coins. Note that this notion is closely related to $\mathcal{L}_\delta$-leakage for the oracle $\mathcal{L}_\delta$ which internally tosses coins and decides with probability δ to respond with the output of $\mathcal{L}$. Interestingly, however, the two notions are *not* equivalent: in the full version of this paper we show that there exist choices of $\mathcal{F}$, $\delta \in [0, 1]$, oracle $\mathcal{L}$, and protocols Π for which Π is a $(\delta, \mathcal{L})$-leakage secure protocol, but not $\mathcal{L}_\delta$-leakage secure. For our purposes, $(\delta, \mathcal{L})$-leakage will be more convenient.

Definition 3 (Topology Hiding with $(\delta, \mathcal{L})$-Leakage). *Let $\delta \in [0, 1]$ and $\mathcal{L}$ a leakage oracle functionality. We say that a protocol Π securely realizes a functionality $\mathcal{F}$* hiding topology with $(\delta, \mathcal{L})$-leakage *if it realizes $(\mathcal{F}_{graphInfo}||\mathcal{F}||\mathcal{L})$ in the $\mathcal{F}_{graph}$-hybrid model with the following property: For any adversarial environment $\mathcal{Z}$, it holds with probability $(1 - \delta)$ over the random coins of the simulator $\mathcal{S}$, that $\mathcal{S}$ does not make any call to $\mathcal{L}$.*

In the full version of this paper we show that this notion of $(\delta, \mathcal{L})$-leakage provides a natural form of composability.

Topology Hiding in the Fail-Stop Model

We now define security for the case that the adversary must follow the protocol (as in the semi-honest case), but may fail nodes. Consider the functionality $\mathcal{F}_{\text{graph-failstop}}$ given in fig. 2, which serves as the analog of $\mathcal{F}_{\text{graph}}$ in the semi-honest model.

As the initialization phase (and ideal-world-counterpart) of $\mathcal{F}_{\text{graph-failstop}}$ is identical to that of $\mathcal{F}_{\text{graph}}$, we denote it the same: $\mathcal{F}_{\text{graphInfo}}$. As before, the communication phase consists of repeated invocation of $\mathcal{F}_{\text{graph-failstop}}$. The fail input in the communication phase represents failing a node, as such, it should only be invoked adversarially (not part of normal protocol operation).

[6] Note that this is related to, but a different setting than leakage-resilient protocols, where the model considers leakage information to the adversary in the *real-world* execution.

Fig. 2: The functionality $\mathcal{F}_{\text{graph-failstop}}$.

Participants/Notation:
This functionality involves all the parties $P_1, \ldots, P_m$ and a special graph party P_{graph}.
Initialization Phase:
Inputs: $\mathcal{F}_{\text{graph-failstop}}$ waits to receive the graph $G = (V, E)$ from P_{graph}.
Outputs: $\mathcal{F}_{\text{graph-failstop}}$ outputs $\mathrm{N}_G[v]$ to each P_v.
Communication Phase:
Inputs: $\mathcal{F}_{\text{graph-failstop}}$ may receive one of the following from a party P_v:
1. a destination/data pair (w, m) where $w \in \mathrm{N}(v)$ and m is the message P_v wants to send to P_w. (If w is not a neighbor of v, or if either v or w has previously sent (Fail), $\mathcal{F}_{\text{graph-failstop}}$ ignores this message.)
2. Fail

Output: If the input is of the form:
1. (destination/data pair (w, m)) $\mathcal{F}_{\text{graph-failstop}}$ gives output (v, m) to P_w indicating that P_v sent the message m to P_v.
2. (Fail) $\mathcal{F}_{\text{graph-failstop}}$ gives output Fail to all neighbors of P_v.

Topology-hiding security-with-abort As is the case for standard (non-topology-hiding) MPC, when we allow active adversaries we relax the security definition to security-with-abort. However, there are wrinkles specific to the topology-hiding setting that make our security-with-abort definition slightly different.

In the standard extension of simulation-based security to security-with-abort, we add a special **abort** command to the ideal functionality; when invoked by the ideal-world simulator, all the honest parties' outputs are replaced by $\perp$. When the communication graph is complete, this extra functionality is trivial to add to any protocol: an honest party will output $\perp$ if it receives an **abort** message from any party (since honest parties will never send **abort**, this allows the adversary to abort any honest party, but does not otherwise change the protocol).

In the topology-hiding setting, this extra functionality—by itself—might already be too strong to realize, since, depending on when the abort occurs, the "signal" might not have time to reach all honest parties. (In fact, this is essentially the crux of the fail-stop impossibility result of [52] and of our leakage lower bound in section 3.2).

Thus, when we define security-with-abort for topology-hiding computation, we augment the ideal functionality with a slightly more complex **abort** command: it now receives list of parties as input (the "abort vector"); only the outputs of those parties will be replaced with $\perp$, while the rest of the parties will output as usual.

Note that in the UC model, the environment sees the outputs of *all* parties, including the honest parties. Hence, to securely realize a functionality-with-abort, the simulator must ensure that the simulation transcript, together with the honest parties' output, is indistinguishable in the real and ideal worlds. In the topology-hiding case, this means that the set of aborting parties must also be indistinguishable. Since whether or not a

party aborts during protocol execution depends on the topology of the graph, in order to determine the abort vector the simulator may require the aid of the leakage oracle (in our case, this is actually the only use of the leakage oracle).

Definition 4 (Fail-stop Topology Hiding). *We say that a protocol Π securely realizes a functionality $\mathcal{F}$* hiding topology against fail-stop adversaries *if it realizes $(\mathcal{F}_{graphInfo}\|\mathcal{F})$* with abort *in the $\mathcal{F}_{graph\text{-}failstop}$-hybrid model.*

Recall that general topology hiding computation against fail-stop adversaries is impossible [52]; we thus consider the notion of topology hiding against fail-stop *with $(\delta, \mathcal{L})$-leakage*.

Definition 5 (Fail-stop Topology Hiding with Leakage). *We say that a protocol Π securely realizes a functionality $\mathcal{F}$* hiding topology against fail-stop adversaries with $(\delta, \mathcal{L})$-leakage *if it realizes $(\mathcal{F}_{graphInfo}\|\mathcal{F}\|\mathcal{L})$ with abort in the $\mathcal{F}_{graph\text{-}failstop}$-hybrid model, with the following property: For any adversarial environment $\mathcal{Z}$, it holds with probability $(1-\delta)$ over the random coins of the simulator $\mathcal{S}$, that $\mathcal{S}$ does not make any call to $\mathcal{L}$.*

3 Lower Bounds

We begin by exploring lower bounds on the feasibility of topology-hiding computation protocols. In this direction, we present two results.

First, we demonstrate that topology hiding is inherently a non-trivial cryptographic notion, in the sense that even for semi-honest adversaries and the simple goal of broadcast (achievable trivially when topology hiding is not a concern), topology-hiding protocols imply the existence of oblivious transfer.

We then shift to the fail-stop model, and provide a lower bound on the *amount* of leakage that must be revealed by any protocol achieving broadcast, as a function of the number of rounds and number of parties. This refines the lower bound of [52], which shows only that non-negligible leakage must occur.

Both results rely only on the correctness guarantee of the broadcast protocol in the "legal" setting, where a single broadcaster sends a valid message. We make no assumptions as to what occurs in the protocol if parties supply an invalid set of inputs. (In particular, this behavior will not be need to be encountered within our lower bounds.)

More formally, our lower bounds apply to THC protocols achieving any functionality $\mathcal{F}$ that satisfies the following *single-broadcaster-correctness* property:

Definition 6 (Single-Broadcaster Correctness). *An ideal n-party functionality $\mathcal{F} : \{0,1,\perp\}^n \to \{0,1,\perp\}^n$ will be said to satisfy* single-broadcaster correctness *if for any input vector $(b_1,\ldots,b_n) \in \{0,1,\perp\}^n$ in which a* single *input $b := b_i$ is non-$\perp$, the functionality $\mathcal{F}$ outputs b to all parties within the connected component of P_i (and no output to all other parties).*

3.1 Semi-Honest Topology-Hiding Broadcast Implies OT

Consider the task of broadcast on a given communication graph. If parties are semi-honest, and no topology hiding is required, then such a protocol is trivial: In each round, every party simply passes the broadcast value to each of his neighbors; within n rounds, all parties are guaranteed to learn the value. However, such a protocol leaks information about the graph structure. For instance, the round in which a party receives the broadcast bit is precisely the distance of this party to the broadcaster. It is not clear at first glance whether this approach could be adapted unconditionally, or perhaps enhanced by tools such as symmetric-key encryption, in order to hide the topology.

We demonstrate that such an approach will not be possible. Namely, we show that even semi-honest topology-hiding broadcast (THB) implies the existence of oblivious transfer. This holds even for the weaker notion of *indistinguishability-based* (IND-CTA) topology-hiding security [52]) which directly implies the same lower bound for the simulation-based definition. As described above, our bound applies to protocols for any functionality which satisfies single-broadcaster correctness.

Theorem 1 (THB implies OT). *If there exists an n-party protocol for $n \geq 4$ achieving IND-CTA topology hiding against a semi-honest adversary, for any functionality $\mathcal{F}$ with single-broadcaster correctness, then there exists a protocol for oblivious transfer.*

We note that, because both the following protocol and proof are black box with respect to the IND-CTA topology-hiding broadcast protocol, the proof holds in the presence of secure hardware.

Proof. We present a protocol for semi-honest secure 2-party computation of the OR functionality given such a semi-honest topology-hiding broadcast protocol for $n = 4$ parties. This implies existence of oblivious transfer [21,49,47].

First, observe that in the semi-honest setting, topology-hiding broadcast of messages of any length (even of a single bit) directly implies topology-hiding broadcast of arbitrary-length messages, by sequential repetition.

In a secure OR computation protocol, two parties A, B begin with inputs $x_A, x_B \in \{0,1\}$, and must output $(x_A \vee x_B)$. In our construction, each party A, B will emulate *two* parties in an execution of the 4-party topology-hiding broadcast protocol $\mathcal{P}_{\text{sh-broadcast}}$ for messages of length λ: namely, A emulates P_0^A, P_1^A, and B emulates P_0^B, P_1^B, where P_0^A, P_0^B are connected as neighbors and P_1^A, P_1^B are similarly neighbors. Each of the parties A, B will emulate an edge between its own pair of parties if and only if its protocol input bit $x_A, x_B \in \{0,1\}$ is 1. More formally, the secure 2-party OR protocol is given in Figure 3.

We now demonstrate a simulator for the secure 2-party computation protocol. The simulator receives as input the security parameter 1^λ, the corrupted party C's input x_C (where $C \in \{A, B\}$), the final output $b \in \{0,1\}$, equal to the OR of x_C with the (secret) honest party input bit, and auxiliary input z. As its output, $\mathcal{S}_{\mathcal{A}}(1^\lambda, x_C, b, z)$ simulates an execution of $\mathcal{P}_{\text{OR}}$ interacting with the adversary $\mathcal{A}$ while emulates the role of the uncorrupted party $C' \neq C \in \{A, B\}$, *but using input b in the place of the (unknown) input $x_{C'}$.*

Denote by $\mathsf{view}_{\mathcal{A}}^{\mathcal{P}_{\text{OR}}}(1^\lambda, (x_A, x_B), z)$ the (real) view of the adversary $\mathcal{A}$ within the protocol $\mathcal{P}_{\text{OR}}$ on inputs x_A, x_B, when given auxiliary input z.

Fig. 3: Secure 2-Party OR Protocol $\mathcal{P}_{\text{OR}}$ from Semi-Honest THB

Inputs: Parties A, B have inputs $x_A, x_B \in \{0,1\}$.
Outputs: Parties A, B output $(x_A \vee x_B)$.
Protocol $\mathcal{P}_{\text{OR}}$:

1. A chooses a string $R \leftarrow \{0,1\}^\lambda$ at random.
2. A, B honestly emulate an execution of $\mathcal{P}_{\text{sh-broadcast}}$, as follows:
   - A emulates parties P_0^A, P_1^A with respective neighborhoods $\{P_0^B\}, \{P_1^B\}$ if $x_A = 0$ and with neighborhoods $\{P_1^A, P_0^B\}, \{P_0^A, P_1^B\}$ if $x_A = 1$.
   - B emulates parties P_0^B, P_1^B with respective neighborhoods $\{P_0^A\}, \{P_1^A\}$ if $x_B = 1$ and with neighborhoods $\{P_1^B, P_0^A\}, \{P_0^B, P_1^A\}$ if $x_B = 1$.

   Party P_0^A runs as the designated broadcaster, with input R. All other parties run with input $\perp$, indicating they are not broadcasting.
3. For each $X \in \{A, B\}$, $b \in \{0,1\}$, denote by out_b^X the output of emulated party P_b^X at the conclusion of $\mathcal{P}_{\text{sh-broadcast}}$. A, B output as follows:
   - A outputs 1 iff $\mathsf{out}_0^A = \mathsf{out}_1^A = R$.
   - B outputs 1 iff $\mathsf{out}_0^B = \mathsf{out}_1^B$.

Claim. For every $x_A, x_B \in \{0,1\}$, non-uniform polynomial-time adversary $\mathcal{A}$, and auxiliary input z, it holds that

$$(x_A, x_B, b, \mathsf{view}_{\mathcal{A}}^{\mathcal{P}_{\text{OR}}}(1^\lambda, (x_A, x_B), z)) \stackrel{c}{\cong} (x_A, x_B, b, \mathcal{S}_{\mathcal{A}}(1^\lambda, x_C, b, z)).$$

Proof. First observe that output correctness of $\mathcal{P}_{\text{OR}}$ holds, as follows. By single-broadcaster correctness of $\mathcal{P}_{\text{sh-broadcast}}$ (note that indeed there is a single broadcaster), all parties in the connected component of the broadcaster P_0^A within the emulated execution will output the string R. In particular, this includes P_0^B: i.e., $\mathsf{out}_0^B = R$. In contrast, any party outside the connected component of P_0^A will have a view in the emulated THB protocol that is information theoretically independent of the choice of R, and thus will output R with negligible probability. This means out_1^A and out_1^B will equal R precisely when there exists an edge between x_D^0 and x_D^1 for at least one $D \in \{A, B\}$: that is, iff $(x_A \vee x_B) = 1$.

In the case of $b = 0$, the simulation is perfect. In the case of $b = 1$, indistinguishability of the above real-world and ideal-world distributions follows directly by the indistinguishability under chosen topology attack (IND-CTA) security of $\mathcal{P}_{\text{sh-broadcast}}$. Namely, the simulation corresponds to execution of $\mathcal{P}_{\text{sh-broadcast}}$ on the graph G with an edge between the two uncorrupted parties $P_0^{C'}, P_1^{C'}$, whereas depending on the value of the honest input $x_{C'}$, the real distribution is an execution on either this graph G or the graph G' with this edge removed. A successful distinguisher thus breaks IND-CTA for the challenge graphs G, G'.

3.2 Lower Bound on Information Leakage in Fail-Stop Model

The work of [52] demonstrated that non-negligible leakage on the graph topology must occur in any broadcast protocol in the presence of fail-stop corruptions. In what follows,

we extend this lower bound, quantifying and amplifying the amount of information revealed.

Roughly, we prove that any protocol realizing broadcast with abort must leak $\Omega(n/R)$ bits of information on the graph topology, where n is the number of parties, and R is the number of rounds of interaction of the protocol.[7] More formally, we demonstrate an attack that successfully distinguishes between two different honest party graph structures with advantage $\Omega(n/R)$. This, in particular, rules out the existence of $(\delta, \mathcal{L})$-leakage topology hiding for $\delta \in o(n/R)$, for any leakage oracle $\mathcal{L}$. We compare this to our protocol construction in Section 4.2, which *achieves* $(\delta, \mathcal{L})$-leakage in this model for a single-output-bit $\mathcal{L}$ and $\delta \in O(n^2/R)$. We leave open the intriguing question of closing this gap.

The proof follows an enhanced version of the attack approach of [52], requiring the adversary to control only 4 parties, and perform only 2 fail-stop aborts. At a high level, parties are arranged in a chain with a broadcaster at one end, 2 aborting parties in the middle, and an additional corrupted party who is either on the same side or opposite side of the chain as the broadcaster. In the attack, one of the 2 middle parties aborts in round i, and the second aborts in round $i + d$ as soon as the first's abort message reaches him. Parties on one end of the chain thus see a single abort at round i, whereas parties on the other end see only an abort at round $i + d$. In [52] it is shown that the view of a party given an abort in round i versus $i + 1$ can be distinguished with advantage $\Omega(1/R)$, where R is the number of rounds.

We improve over [52] by separating the two aborting parties by a distance of $\Theta(n)$, instead of distance 1. Roughly, the corrupted party's view in the two positions will be consistent with either an abort in round i or round $i + \Theta(n)$ of the protocol, (versus i and $i + 1$ in [52]), which can be shown to yield distinguishing advantage $\Theta(n)$ better than in [52].

As in Section 3.1, our attack does not leverage any behavior outside the scope of a single broadcaster, and thus applies to any functionality $\mathcal{F}$ satisfying single-broadcaster correctness. Further, the proof only requires that that the protocol is correct and that information is required to travel over the network topology: that is, each node can only transmit information to adjacent nodes in any given given round. Therefore the theorem holds in the presence of secure hardware (which is only held locally and cannot be jointly accessed by different parties).

Theorem 2. *Let $\mathcal{L}$ be an arbitrary leakage oracle. Then no R-round n-party protocol can securely emulate broadcast (with abort) in the fail-stop model while hiding topology with $(\delta, \mathcal{L})$-leakage for any $\delta \in o(n/R)$.*

Proof. Let $\mathcal{P}$ be an arbitrary protocol which achieves broadcast with abort as above. We demonstrate a pair of graphs G_0, G_1 and an attack strategy $\mathcal{A}$ such that $\mathcal{A}$ can distinguish with advantage $\Omega(n/R)$ the executions of $\mathcal{P}$ within G_0 versus G_1. We then prove this suffices to imply the theorem.

Both graphs G_0, G_1 are line graphs on n nodes. In graph G_0, the parties appear in order (i.e., the neighbors of P_i are P_{i-1} and P_{i+1}). In graph G_1, the parties appear

[7] For simplicity we consider a fixed number of rounds R; however, the techniques can be extended to probabilistic R as well.

in order, except with the following change: The location of parties P_3, P_4, P_5 (in nodes $3, 4, 5$ of G_0) are now in nodes $n-2, n-1$, and n, respectively; in turn, parties P_{n-2}, P_{n-1}, and P_n (in nodes $n-2, n-1$, and n of G_0) are now in nodes $3, 4, 5$.

The adversary $\mathcal{A}$ will corrupt: party P_1 (always at position 1 in both graphs), which we will denote as B the broadcaster; party P_4 (who is in position 4 of G_0 and position $n-1$ in G_1), which we will denote as D the "detective" party; and parties P_7 and P_{n-4} (in fixed positions $7, n-4$) who we will denote as A_1 and A_2 the aborting parties. For simplicity of notation, in the following analysis, we will denote the two nodes $4, n-1$ in which D can be located as v, v'. We will further denote the distance $(n-4)-7$ between the aborting parties A_1 and A_2 as m; note that $m \in \Theta(n)$.

Note that the neighbors of all corrupted parties are the same across G_0 and G_1 (this is the purpose of moving the uncorrupted parties P_3, P_5 in addition to P_3, as well as maintaining a gap between the collections of relevant corrupted parties).

We define two events:

$$\begin{aligned} E_i &:= \text{Event that the first abort occurs in round } i\text{, by either } A_1 \text{ or } A_2 \\ H_{v,b} &:= \text{Event that the party at node } v \text{ outputs the correct broadcast bit } b \\ &\qquad \text{(note that this depends on the protocol and the graph on which it is run)} \end{aligned}$$

By (single-broadcaster) correctness of $\mathcal{P}$, it must hold for every broadcast bit b that $\Pr[H_{v,b}|E_R] = 1$: that is, if node A_1 or A_2 aborts in the final round, then the news of the abort will not reach node v, in which case the corresponding party must output in the same (correct) fashion as if no abort occurred.

By an information argument, it must be the case for v' that for some choice of $b \in \{0,1\}$, it holds that $\Pr[H_{v'_\ell,b}|E_1] \leq 1/2$. Recall that v'_ℓ lies on the opposite end of the aborting parties compared to the broadcasting node B.

Combining the above two statements, it holds that $\exists b \in \{0,1\}$ such that we have $\Pr[H_{v,b}|E_R] - \Pr[H_{v',b}|E_1] \geq 1/2$.

By telescoping and the pigeonhole principle, there must exist some m-step of rounds between R and 1 which contains at least (m/R) of this mass:

$$\exists b \in \{0,1\}, \exists j^* \in \left[\left\lfloor \frac{R}{m} \right\rfloor\right] \text{ s.t. } \begin{cases} \Pr[H_{v_\ell,b}|E_{j^*m}] - \Pr[H_{v'_\ell,b}|E_{(j^*-1)m}] \geq \frac{m}{2R}, \text{ or} \\ \Pr[H_{v'_\ell,b}|E_{j^*m}] - \Pr[H_{v_\ell,b}|E_{(j^*-1)m}] \geq \frac{m}{2R}. \end{cases} \tag{1}$$

We now leverage these facts to describe an attack.

The Attack. Consider a non-uniform adversary $\mathcal{A}$ hardcoded with: the $b \in \{0,1\}$ and $j^* \in [\lfloor R/m \rfloor]$ from Equation (1), and whether we are in the top or the bottom of the two cases (in which the roles of v and v' are reversed). Suppose temporarily that we are the first case. $\mathcal{A}$ proceeds as follows:

1. Corrupt the set of parties $\{B, A_1, A_2, D\}$ from the set of n parties.
2. Execute an honest execution of protocol $\mathcal{P}$ up to round $(j^*-1)m$. In the execution, party B is initialized with input broadcast bit b, and all other emulated parties with input $\perp$ (i.e., not broadcasting).
3. At round $(j^*-1)m$, abort party A_2. Continue honestly simulating all other corrupted parties for m rounds.

4. At round j^*m, abort party A_1.
5. Continue honestly simulating all other corrupted parties until the conclusion of the protocol. Denote by out_D the protocol output of the "detective" party D.
6. $\mathcal{A}$ outputs $\mathsf{out}_D \oplus b$.

If we are instead in the setting of case 2 in Equation (1), then the attack is identical, except that the roles of A_1 and A_2 in Step 2 are swapped.

Claim. $\mathcal{A}$ distinguishes between the execution of $\mathcal{P}$ on graphs G_0 and G_1 with advantage $\Omega(n/R)$.

Proof. This argument follows a similar structure as that of [52]. Suppose wlog we are in Case 1 of Equation (1). (Case 2 is handled in an identical symmetric manner.) Recall that the aborting parties A_1, A_2 are distance m from one another. This means that A_1 aborts at some round $(j^* - 1)m$, then the view of parties to his left (in particular, the party at node $v = 4$) is as in the event $E_{(j^*-1)m}$. However, information of this abort must take at least m rounds in order to reach any parties to the *right* of A_2; thus, since A_2 aborts already at this round j^*m, A_1's abort is never seen by parties to the right of A_2 (in particular, the party at node $v' = m - 1$), who will have view consistent with event E_{j^*m}.

If the execution took place on G_0, then D is at node v, otherwise it is at node v'. Thus, the advantage of the adversary $\mathcal{A}$ is precisely $\Pr[H_{v_\ell,b}|E_{j^*m}] - \Pr[H_{v'_\ell,b}|E_{(j^*-1)m}] \geq \frac{m}{2R}$.

Now, suppose that $\mathcal{P}$ securely realizes $\mathcal{F}_{BC}^{sh}$ hiding topology with $(\delta, \mathcal{L})$-leakage, for some leakage oracle $\mathcal{L}$. Consider the distribution $\mathcal{S}'$ generated by running the $(\delta, \mathcal{L})$-leakage simulator $\mathcal{S}$, but aborting and outputting $\perp$ in the event that the randomness of $\mathcal{S}$ indicates it will query the leakage oracle. By construction, the statistical distance between $\mathcal{S}'$ and the properly simulated distribution $\mathcal{S}$ with access to leakage $\mathcal{L}$ for any fixed choice of real inputs (i.e., honest graph) is bounded by δ. In particular, $\mathcal{S}'$ is within δ statistical distance from both $\mathcal{S}^{\mathcal{L}(G_0)}$ and $\mathcal{S}^{\mathcal{L}(G_1)}$ (denoting oracle access to leakage on the respective graphs G_0, G_1). By the assumed $(\delta, \mathcal{L})$-leakage simulation security of the protocol, for both $b = 0, 1$ it holds that $\mathcal{S}^{\mathcal{L}(G_b)}$ is computationally indistinguishable from the adversarial view of execution of $\mathcal{P}$ on G_b. Combining these steps, we see that no efficient adversary can distinguish between the executions of $\mathcal{P}$ on graphs G_0 and G_1 with advantage non-negligibly better than δ.

Therefore, combined with Claim 3.2 it follows that δ must be bounded below by $\delta \in \Omega(n/R)$.

4 Upper Bounds

In this section, we observe that oblivious transfer implies semi-honest topology-hiding computation on small diameter graphs, and then present two constructions of topology-hiding broadcast with security against fail-stop adversaries from secure hardware.

The construction of semi-honest THC for graphs with small diameter follows is a modified variant of the protocol given in [52].

The first fail-stop secure topology-hiding broadcast protocol leaks at most one bit in the presence of aborts, by exploiting a stratified structure where the protocol is broken

into epochs corresponding to the parties playing. If at the end of an epoch the communication network is still intact, the corresponding party will receive output at the end of the protocol. If the network is not intact, the party will not receive the broadcast bit. Aborting during a given an epoch may leak a bit about the distance from some aborting party to the one corresponding to the epoch. But by the next epoch all parties (or rather, their secure hardware) will be aware that the network has been disrupted and no future epochs will yield output to their corresponding parties.

The second protocol is a simple modification of the first which extends each epoch into many smaller eras. The era that actually determines the party's output is randomly (and secretly) chosen by the secure hardware. So, unless the first abort occurs in this era (leaking a single bit), all parties (namely, their secure hardware) will reach consensus about the network being disrupted and what their output is, independently of the network topology. Thus a bit is only leaked with probability that degrades inversely with the number of eras.

4.1 OT Implies Semi-Honest THC (for Small-Diameter Graphs)

Semi-Honest THC for small-diameter graphs is, in fact, *equivalent* to OT. This follows from a minor modification to the MPC-based protocol of [52]. Recall, the high-level approach of [52] is a recursive construction:

- At the base level, nodes run the (insecure) OR-and-forward protocol, except that every node has a key pair for a PKE scheme, and every message to node i from one of its neighbors is encrypted under pk_i.
- The recursion step is to replace every node by an MPC protocol in its local neighborhood (the node and all its immediate neighbors), such that its internal state is revealed only if the entire neighborhood colludes.
  The communication pattern for each of these MPCs is a star. Since leaf nodes can't communicate directly, they must pass messages through the center. In order to simulate private channels the leaf nodes first exchange PKE public keys (we are in the semi-honest model, so man-in-the-middle attacks are not relevant) and then use the PKE scheme to encrypt messages between them.
  Note that the MPC simulates the node's next-message function. All nodes receive as input a secret-share of the state from the previous round, and output a secret share of the updated state. In addition, the input of the central node contains the list of messages received from its neighbors in the previous round, and its output contains the list of messages to send to its neighbors in this round. At the end of the MPC execution, the central node sends the messages to its neighbors (who will then use them as part of their input in MPC executions in the next round).
  This structure is the reason for requiring the messages to a node at the base level to be encrypted—the MPC doesn't hide the messages themselves from the central node, hence privacy would be lost if they were unencrypted.

We will replace the PKE scheme with a key-agreement protocol and a symmetric encryption scheme. Since the existence of OT implies both of these primitives, the resulting protocol can be build from OT (we note that, unlike the construction of OT from

THB, this construction is *not* black-box in the OT primitive—the recursion step will require non-black-box access to the OT).

For the base step, instead of using a PKE scheme, every node will perform a key-agreement protocol with all of its neighbors. Henceforth, messages from p_i to a direct neighbor p_j will be encrypted under their shared key (using the symmetric encryption scheme). This ensures that to an adversary that does not have access to the state of either p_i or p_j messages between the two are indistinguishable from random.

For the recursion step, we do the same thing except with the leaf nodes. That is, every pair of leaf nodes p_i, p_j will execute the key agreement protocol, using the central node to pass messages. Henceforth, the private channel between them in the MPC protocol will be simulated by encrypting messages using their shared key and passing the messages via the central node.

4.2 Constructions for Fail-Stop Adversaries

Both fail-stop protocols presented in this section achieve a standard notion of broadcast. The broadcast functionalities considered previously were defined with respect to the network topology, particularly its connected components. Forthwith, we will assume the network (before failures) is fully connected.

Definition 7 **($\mathcal{F}_{BC}$).** *The ideal n-party broadcast functionality $\mathcal{F}_{BC}$ is defined by the following output behavior on input $b_i \in \{0, 1, \perp\}$ from every party P_i: if a exactly one $b := b_i$ is non-$\perp$, then $\mathcal{F}_{BC}$ outputs b to all parties; otherwise, all outputs are $\perp$.*

Protocol with one bit of leakage

We present a topology-hiding broadcast protocol secure against fail-stop adversaries making static corruptions given one bit of leakage. We assume a secure hardware box and one-way functions. We also note that the protocol presented here is secure against semi-honest adversaries without any leakage.

In what follows we consider parties to correspond to their node in the set of all network nodes, $[n]$.

Our protocol has two major phases:

1. **Graph Collection:** Collect a description of the graph, inputs, and aliases. This phase runs for a number of rounds proportional to the network diameter. Any abort seen during this phase by any party will cause that part to abort in the final round.
2. **Consistency Checking and Abort Segregation:** Checking consistency and outputting. This phase has a number of subphases corresponding to the number of parties, n.
   Each subphase runs for a number of rounds proportional to the size of the network. During subphase i, party i will no longer abort if the first abort it sees takes place during that subphase or later. However, an abort seen by any party in $\{i + 1, i + 2, \ldots, n\}$ will still cause that party to abort in the final round. The intuition is that if any party aborts in a subphase, all non-aborted parties are guaranteed to know that an abort has occurred by the end of next subphase.

By subphase n, if no abort has been seen, all honest parties will output correctly, regardless of subsequent abort behaviors.

The hardware box, aside from initialization and final output, will take as input and output authenticated-encryptions of the player's current "state." The plaintext state of a party P_u with session key k_u after round i, denoted s_u^i, contains the following information:

- The party's alias: id_u, a random λ-length string chosen at outset.
- The round number: i_u.
- Current "knowledge" of the graph: G_u^i.
- Current "knowledge" of inputs: $\mathbf{m}^{u,i} = (m_1^{u,i}, \ldots, m_n^{u,i})$.
- Current "knowledge" of session keys: $\mathbf{sk}^{u,i} = (\mathrm{sk}_1^{u,i}, \ldots, \mathrm{sk}_n^{u,i})$.
- First round an abort was seen: a_u^i.
- Indicator of whether or not a neighbor has aborted in a previous round: $\mathbf{b}_u^i := (b_{v_1}^i, \ldots, b_{v_d}^i)$. (This information is not strictly necesssary, but convenient when proving security.)
- abort flag: α.

We now define the C_{fs} functionality that is embedded in the hardware box. As stated above, we take network locations (typically denoted u or v) to be elements of $[n]$. State information is represented as vectors over the alphabet $\Sigma = \{0, 1, ?, \perp\}$. We take $\boldsymbol{e}_i(x)$ to be the vector of all ?s with $x \in \{0, 1, \perp\}$ in the ith position (the length of the vector should be clear from context, unless otherwise specified). If the vector is in fact an $m \times n$-matrix, we take $\boldsymbol{e}_i(\boldsymbol{x})$ to denote the vector with $?^n$ in all rows, except i which contains $\boldsymbol{x} \in \Sigma^n$. The network (graph) is represented as an adjacency matrix, with ?s denoting what is unknown and $\perp$s representing errors, or inconsistencies. In particular, the closed neighborhood of u, $N[u]$, is the $n \times n$ matrix with the adjacency vector of u in the uth row and ?s elsewhere. Let $H : 2^{\Sigma^m} \to \Sigma^m$ denote the component-wise "accumulation" operator where the ith output symbol, for $i \in [m]$, is as defined as follows:

$$H_i \boldsymbol{X} := \begin{cases} 1 \text{ if } \exists \boldsymbol{x} \in \boldsymbol{X} \in [k] : \boldsymbol{x}_i = 1, \text{ and } \forall \boldsymbol{y} \in \boldsymbol{X} : \boldsymbol{y}_i \in \{1, ?\} \\ 0 \text{ if } \exists \boldsymbol{x} \in \boldsymbol{X} : \boldsymbol{x}_i = 0, \text{ and } \forall \boldsymbol{y} \in \boldsymbol{X} : \boldsymbol{x}_i \in \{0, ?\} \\ ? \text{ if } \forall \boldsymbol{x} \in \boldsymbol{X} : \boldsymbol{x}_i = ? \\ \perp \text{ otherwise} \end{cases}$$

Finally, let $R = n(n + 2) + 1$ (final number of rounds).

Fig. 4: The Functionality C_{fs} (Part 1: continued in figure 5).

Notation:

Let $(\mathrm{G}, \mathrm{E}, \mathrm{D})$ denote a symmetric key authenticated encryption scheme. Let $\text{PRF} \leftarrow \{\text{PRF}_k\}_k$ denote a pseudorandom function with security parameter λ, chosen randomly from some such family during setup. Additionally, let $\text{msk} \leftarrow \mathrm{G}(1^\lambda)$, again chosen during setup.

We take u to denote the location in the network associated with the party, P_u, using this instance of C_{fs}.

As above, we use the following to denote state information:

$$s_v^i = (v; \text{sk}_v; i; G_v^i; \mathbf{m}^{v,i} = m_1^{v,i}, \ldots, m_n^{v,i}; \mathbf{sk}^{v,i} = \text{sk}_1^{v,i}, \ldots, \text{sk}_n^{v,i}; a_v^i; \mathbf{b}_v^i := b_{v_1}^i, \ldots, b_{v_d}^i, \alpha).$$

Input:

- (Initialization input) $x = (m, u, N[u], \text{sk}_u)$, where m should be the broadcast message if u is broadcaster, and $\perp$ otherwise. $N[u]$ denotes the closed neighborhood of u. sk is a random session key.
- (Round input) d authenticated encryptions of the form:

$$x = \left(x_u, x_{v_1}, \ldots, x_{v_d}, \text{sk}_u\right)$$

where $v_1, \ldots, v_d$ are in $N(u)$, additionally in numerical order, and ignored if $\deg(u) < d$.

For all $v \in N[v]$, x_v should either an encrypted state, $\mathrm{E}_{\text{msk}}(s_v^i)$, or Abort. Moreover, we

Let $\mathcal{L}$ denote the the class of efficient leakage functions that leak one bit about the topology of the network.

Theorem 3. *The protocol $\mathcal{F}_{\text{fs-broadcast}}$ topology hiding realizes broadcast with $\mathcal{L}$ leakage with respect to static corruptions.*

Remark 1. By observing that the leakage oracle is only used by the simulator in the event of an abort, the protocol $\mathcal{F}_{\text{fs-broadcast}}$ is secure against probabilistic polynomial time semi-honest adversaries without leakage.

Correctness. Assuming there are no aborts, correctness follows by induction on the rounds. By inspection, local consistency checks will pass under honest evaluation if there are no aborts. Clearly, at the end of round i, the encrypted broadcast message will have reached all parties whose distance is at most i from the broadcaster. So by round $n + 1$, all parties will have the message. Similarly, by round $n + 1$, all local descriptions of the network and aliases will have reached all parties. Moreover, no abort flags will be triggered. Thus, the global consistency checks will pass in the final round, R, and all parties will receive the broadcast message.

By inspection, it is easy to see that if evaluation is semi-honest with possible aborts each party will either output the unique non-$\perp$ input, or abort.

Fig. 5: The Functionality C_{fs} (Part 2)

Computation:

If input is of "Initialization" format: I.e., $x = (m, u, N[u], \text{sk}_u)$.
Let $\boldsymbol{s}_u^1 = (u; \text{sk}; 1; G_u^1 = N[u]; \boldsymbol{e}_u(m_u); \boldsymbol{e}_u(\mathbf{sk}_u); \perp; 0, \ldots, 0; 0)$, and output: $\text{E}_{\text{msk}}(s_u^1; \text{PRF}(x))$. If $N[u]$ is not a closed neighborhood (i.e. if there is more than one node with more than one edge), output $\perp$.

If input is of "Round" format: I.e., $x = (x_u, x_{v_1}, \ldots, x_{v_d}, \text{sk})$ where $x_v = \text{E}_{\text{msk}}(\hat{s}_v)$ or Abort, and $\hat{s}_v = (\hat{v}; \hat{\text{sk}}_v; \hat{i}_v; G_v; \hat{\mathbf{m}}^v; \hat{a}_v; \hat{\mathbf{b}}_v; \hat{\alpha}_v)$, for $v \in N[u]$

Perform local consistency checks:

- If $x_u =$ Abort, output $\perp$.
- Authenticate/decrypt all non-Abort inputs, to get $\hat{\boldsymbol{s}}_u, \hat{\boldsymbol{s}}_{v_1}, \ldots, \hat{\boldsymbol{s}}_{v_\ell}$. If authentication or encryptions fails, halt and output $\perp$.
- If round counters $(\hat{i}_u, \hat{i}_{v_1}, \ldots, \hat{i}_{v_\ell})$ are not all equal, halt and output $\perp$.
- For each network location $v \in N[u]$, where $N[u]$ is extracted from $\boldsymbol{s}_u$, if v does not correspond to plaintext state input (in correct position) or Abort, set $\alpha' = 1$.
  Additionally, if some $v \notin N[u]$ is associated with any input state, output $\perp$.
- If there exists $x_v \neq$ Abort but $\hat{b}_v^u = 1$, output $\perp$.
- If the plain text session key input sk does not match the session key extracted from $\hat{\boldsymbol{s}}_u$, output $\perp$.

Perform global consistency checks and accumulation:

- Initialize abort flag to be the same as in $\hat{\boldsymbol{s}}_u$: $\alpha' = \hat{\alpha}_u$.
- If sender keys don't match "stored" keys, in other words $\hat{\mathbf{sk}}_{\hat{v}} = (\hat{\mathbf{sk}}^{\hat{u}})_{\hat{v}}$ for all input non-Abort-ing locations $\hat{v}$, then set $\alpha' := 1$ (encrypted abort).
- $i' := \hat{i}_u + 1$,
- Generate $G' := H\{\hat{G}^v : v \in N[u]\}$, If any component of G' is $\perp$, let $\alpha' := 1$.
- $\mathbf{m}' := H\{\hat{\mathbf{m}}^v : v \in N[u]\}$, If any component of $\mathbf{m}'$ is $\perp$, let $\alpha' := 1$.
- $\mathbf{sk}' := H\{\hat{\mathbf{sk}}^v : v \in N[u]\}$. If any component of $\mathbf{sk}'$ is $\perp$, let $\alpha' := 1$.
- For all $v \in \hat{N}(u)$, if $x_v =$ Abort then set $b'_v := 1$. Otherwise, set $b'_v = \hat{b}_v^u$. Let $\mathbf{b}' = (b'_{v_1}, \ldots, b'_{v_d})$.
- If if $\hat{a}_u \neq \perp$ and either $\hat{a}_v^i \neq \perp$ or $x_v =$ Abort for any $v \in \hat{N}[u]$,then set $a' := \hat{i}_u$. Otherwise, $a' := \hat{a}_u$.

Check to enforce commitment:
If $\hat{i}_u = n + 1$ and $\hat{G}_u$ contains any '?', set $\alpha' := 1$.

Output:

- if $\hat{i}_u < R$, then output: $\text{E}_{\text{msk}}(\boldsymbol{s}'; \text{PRF}(x))$,
  where $\boldsymbol{s}' := (\hat{u}_u; \hat{\text{sk}}_u; i'; G'; \mathbf{m}'; \mathbf{sk}'; a'; \mathbf{b}', \alpha')$.
- Else, if $\hat{i}_u = R$: If there exists more than one v such that $(\hat{\mathbf{m}}^u)_v \neq \perp$, output $\perp$. If $a' \leq (\hat{u} + 1) \cdot n$ or $\alpha' = 1$, output $\perp$. Otherwise, output the message corresponding to the unique non-$\perp$ location: $(\hat{\mathbf{m}}^u)_{v^*}$.
- Else, output $\perp$.

Fig. 6: The protocol $\mathcal{F}_{\text{fs-broadcast}}$ in $\mathcal{F}_{\text{graph}}$-hybrid model.

Notation: Let $O(C_{\text{fs}})$ denote a secure hardware box (oracle) running C_{fs}.

Input: Each party P_x receives $O(C_{\text{fs}})$. If P_x is broadcaster, P_x also has input $m = m_x$, the message to be broadcast. If P_x is not broadcaster, m_x is the all zero vector.

Protocol for Broadcast:

- (Initialization) P_u chooses a random id $\leftarrow \{0,1\}^\lambda$. P_u computes $c_u^1 = O(C_{\text{fs}})(m_u; u; N(x); \text{sk}_u)$. P_u sends $(v; c_u^1)$ to $\mathcal{F}_{\text{graph}}$ for all $v \in N(u)$. Each $v \in Nu$ receives $(u; c_u^1)$. P_u receives $(v; c_v^1)$ for each $v \in N(u)$.
- (Round $i = 2$ to $R - 1$) P_u computes $c_u^i = O(C_{\text{fs}})(c_u^{i-1}, c_{v_1}^{i-1}, \ldots, c_{v_d}^{i-1})$. P_u sends $(v; c_u^i)$ to $\mathcal{F}_{\text{graph}}$ for all $v \in N(u)$. Each $v \in N(u)$ receives $(u; c_u^i)$. P_u receives $(v; c_v^i)$ for each $v \in N(u)$.
- (Round R) P_u computes $\hat{m} = O(C_{\text{fs}})(c_u^R, c_{v_1}^R, \ldots, c_{v_d}^R; \text{sk}_u)$, and outputs $\hat{m}$.

Security. We start with a rough overview of simulation and why it works, with the full proof of security given in the full version.

Crucially, the authenticated encryption of internal states makes it infeasible for an adversary to either forge states, or glean any information about their contents. As the simulator may have incomplete information about the graph topology, this allows it to send fake states and simply output consistently with the real protocol. Moreover, the unforgeability gives the simulator full knowledge of any initial information used when querying the box, and, importantly, how these queries relate to one another (especially whether or not they are consistent).

The difficulty in the proof is in dealing with "replay" attacks, where the adversary combines information from the honest nodes in malicious ways with other initializations. The session keys aid in this by rendering it infeasible for the adversary to replay as an uncorrupted party with a modified local topology. Additionally, the collection phase implies that honest nodes have complete information about topology and initialization information used in execution by round n. Thus, when this information is later combined with initializations that do not match the execution exactly, the only plaintext output such a malicious adversary will receive is $\perp$. Together this means the simulator only has to provide output when query structure matches execution almost exactly (up to somewhat local aborts). The upshot being that the simulator can provide output identical to a real execution, even though it has incomplete knowledge of the network topology.

One of the dangers in simulation is if an adversary corrupts all the nodes within a distance r of a given node, it has enough information to "fast forward" the node to get its program outputs for the next $r + 1$-rounds. Additionally, after its threshold round has occurred, an adversary can abort all the neighbors of a node, and iterate the remaining rounds by itself to get output. However, we show that the simulator will always have enough information to fool such adversaries.

- For each party, P_u, generate $(n+2)n$ random encryptions of 0. These will constitute the messages sent by honest parties to corrupted parties, and the output of the oracle for queries consistent with semi-honest evaluation.
- The simulated oracle will remember all queries from the adversary in a data structure, the outputs given, and how they relate to one another (as we explain in Def. **??**). The idea is that "valid" inputs will return more encryptions of 0 until the last "round." Then, the simulator will use the data structure to determine the appropriate output given the initialization queries used in conjunction with the single bit of leakage (supposing there was an Abort in the execution, described below). We describe the simulation of the hardware box program in more detail below in Figure **??**.

  Any query which isn't an initialization input or a concatenation of previous queries will immediately return $\perp$. Likewise, any combination of previous queries that correspond to locally inconsistent topologies or round numbers. Moreover, after n rounds any queries that yield an inconsistent topology (the combined the base initializations of all queries, and the previous queries they depend on, does not yield a single consistent graph) will be recognized in the real world. Thus the simulator need only give output in the final round if all queries, including their ancestors, correspond to consistent graph initialization.

  If the first abort occurred in round i, the simulator will query to determine if the real encrypted state of party $j = \lfloor i/n \rfloor + 2$ contained information "witnessed" the abort after round $(j+2)n$. If so, the queries corresponding to the final round of execution for parties $P_1, P_2, \ldots, P_j$ will return the broadcast message, and Abort to all other parties. If P_j "witnessed" the abort on or before round $(j+2)n$, then the query corresponding to P_j's final input to the black box program will return Abort as well (all other outputs for "final" queries are unchanged from the previous case). The simulator uses the single bit of leakage to determine if an abort reach P_j in time.
- When the adversary corrupts a party once the protocol is underway, first choose a random sk. Then, fix the oracle to yield pre-determined ciphertexts corresponding to the honest initialization and pre-determined messages from its neighborhood.

We refer the reader to the full version of this paper for a complete description of the simulator and hybrids.

Protocol with arbitrarily low leakage

This protocol is only a slight modification of the previous one. To achieve δ leakage, each party is not associated with a single subphase, but instead a sequence of $n\lceil 1/\delta \rceil$ subphases of a *zone*. At the outset, parties provide randomness (which can be drawn from the session keys), which will assist in selecting one of these subphases to be the true one. Thus, the probability of an aborting adversary successfully hitting any sub-subphase with its first abort is dependent on the graph structure is $< 1/\delta$.

The state is identical to the previous, with one additional parameter, t, encoding the threshold round.

The protocol here is the same as before, except now $R = n(n^2\lceil 1/\delta \rceil + 2) + 1$.

We now define $C_{\text{rand-fs}}$ functionality. We take notation to be consistent with the previous construction, C_{fs}, where not otherwise specified.

Fig. 7: The Functionality $C_{\text{rand-fs}}$ (Part 1: continued in figure 8).

Notation:

Let (G, E, D) denote a symmetric key authenticated encryption scheme. Let PRF denote a pseudorandom function with security parameter λ, chosen randomly from some such family during setup. Additionally, let $\text{msk} \leftarrow \text{G}(1^\lambda)$, again chosen during setup.

Let $\text{Threshold}_{\delta,n}$ be a pseudorandom function such that $\text{Threshold}_{\delta,n}(u; \text{sk}_u)$ that outputs a a pseudorandom integer in $\{3n + (u-1)n^2\lceil 1/\delta\rceil, 4n + (u-1)n^2\lceil 1/\delta\rceil, \ldots, 2n + un^2\lceil 1/\delta\rceil\}$.

As above, we use the following to denote state information:

$$s_v^i = (v; \text{sk}_v; i; G_v^i; \mathbf{m}^{v,i} = m_1^{v,i}, \ldots, m_n^{v,i}; \mathbf{sk}^{v,i} = \text{sk}_1^{v,i}, \ldots, \text{sk}_n^{v,i}; a_v^i; \mathbf{b}_v^i := b_{v_1}^i, \ldots, b_{v_d}^i; \alpha_v; t_v).$$

Input:

- (Initialization input) $x = (m, u, N[u], \text{sk}_u)$, where m should be the broadcast message if u is broadcaster, and $\perp$ otherwise. $N[u]$ denotes the closed neighborhood of u, a binary vector representing all adjacencies (or lack thereof). sk is a random session key.
- (Round input) d authenticated encryptions of the form:

$$x = \left(x_u, x_{v_1}, \ldots, x_{v_d}, \text{sk}_u\right)$$

where $v_1, \ldots, v_d$ are in $N(u)$ and ignored if $\deg(u) < d$.

For all $v \in N[v]$, x_v is either an encrypted state, $\text{E}_{\text{msk}}(s_v^i)$, or Abort.

Theorem 4. *For any $\delta = 1/poly(\lambda, n)$, the protocol $\mathcal{F}_{fs\text{-}broadcast}$, when C_{fs} is replaced with $C_{rand\text{-}fs}(\delta)$, topology hiding realizes broadcast with $(\delta, \mathcal{L})$ leakage with respect to static corruptions.*

Correctness. The proof here is nearly identical to the preceding one.

Security. Here the simulator is nearly identical to the previous, except it chooses each location's random threshold itself, and only queries the leakage oracle if when the first real abort occurs in chosen block. Here, it queries more-or-less identically to before. For all other nodes it outputs according to whether the threshold has already occurred or not. The leakage oracle itself will represent an identical functionality to the previous case.

Because the distribution of thresholds is computationally indistinguishable in simulated case from the real one, an adversary will be unable to distinguish. As many of the lemmas from the previous construction hold here, we will simply bound the probability that the leakage oracle is called (and hence, the leakage itself).

Fig. 8: The Functionality $C_{\text{rand-fs}}$ (Part 2)

Computation:

If input is of "Initialization" format: I.e., $x = (m, u, N[u], \text{sk}_u)$.
Let $\boldsymbol{s}_u^1 = (u; \text{sk}; 1; G_u^1 = N[u]; \boldsymbol{e}_u(m_u); \boldsymbol{e}_u(\mathbf{sk}_u); \perp; 0, \ldots, 0; 0; \text{Threshold}(u; \text{sk}))$, and output: $\text{E}_{\text{msk}}(s_u^1; \text{PRF}(x))$. If $N[u]$ is not a closed neighborhood (i.e. if there is more than one node with more than one edge), output $\perp$.

If input is of "Round" format: I.e., $x = (x_u, x_{v_1}, \ldots, x_{v_d})$ where $x_v = \text{E}_{\text{msk}}(\hat{s}_v^i)$ or Abort, and $\hat{s}_v^i = (\hat{v}; \hat{\text{sk}}_v; \hat{i}_v; G_v; \hat{\mathbf{m}}^v; \hat{a}_v; \hat{\mathbf{b}}_v; \hat{\alpha}_v; \hat{t}_v)$, for $v \in N[u]$.

Perform local consistency checks:

- If $x_u = \text{Abort}$, output $\perp$.
- Authenticate/decrypt all non-Abort inputs, to get $\hat{\boldsymbol{s}}_u, \hat{\boldsymbol{s}}_{v_1}, \ldots, \hat{\boldsymbol{s}}_{v_\ell}$. If authentication or encryptions fails, halt and output $\perp$.
- If round counters $(\hat{i}_u, \hat{i}_{v_1}, \ldots, \hat{i}_{v_\ell})$ are not all equal, halt and output $\perp$.
- For each network location $v \in N[u]$, where $N[u]$ is extracted from $\boldsymbol{s}_u$, if v does not correspond to plaintext state input (in correct position) or Abort, output $\perp$.
  Additionally, if some $v \notin N[u]$ is associated with any input state, output $\perp$.
- If there exists $x_v \neq \text{Abort}$ but $\hat{b}_v^u = 1$, output $\perp$.
- If the plain text local session key input sk does not match the session key extracted from $\hat{\boldsymbol{s}}_u$, output $\perp$.

Perform global consistency checks and accumulation:

- Initialize abort flag to be the same as in $\hat{\boldsymbol{s}}_u$: $\alpha' = \hat{\alpha}_u$.
- If sender keys don't match "stored" keys, in other words $\hat{\mathbf{sk}}_{\hat{v}} = (\hat{\mathbf{sk}}^{\hat{u}})_{\hat{v}}$ for all input non-Abort-ing locations $\hat{v}$, then set $\alpha' := 1$ (encrypted abort).
- $i' := \hat{i}_u + 1$,
- Generate $G' := H\{\hat{G}^v : v \in N[u]\}$, If any component of G' is $\perp$, let $\alpha' := 1$.
- $\mathbf{m}' := H\{\hat{\mathbf{m}}^v : v \in N[u]\}$, If any component of $\mathbf{m}'$ is $\perp$, let $\alpha' := 1$.
- $\mathbf{sk}' := H\{\hat{\mathbf{sk}}^v : v \in N[u]\}$. If any component of $\mathbf{sk}'$ is $\perp$, let $\alpha' := 1$.
- For all $v \in \hat{N}(u)$, if $x_v = \text{Abort}$ then set $b'_v := 1$. Otherwise, set $b'_v = \hat{b}_v^u$, for all $v \in N(u)$. Let $\mathbf{b}' = (b'_{v_1}, \ldots, b'_{v_d})$.
- If $\hat{a}_u \neq \perp$ and either $\hat{a}_v \neq \perp$ or $x_v = \text{Abort}$ for any $v \in \hat{N}[u]$,then set $a' := \hat{i}_u$. Otherwise, $a' := \hat{a}_u$.

Check to enforce commitment:
If $\hat{i}_u = n + 1$ and $\hat{G}_u$ contains any '?', set $\alpha' = 1$.

Output:

- if $\hat{i}_u < R$, then output: $\text{E}_{\text{msk}}(\boldsymbol{s}'; \text{PRF}(x))$,
  where $\boldsymbol{s}' := (\hat{u}_u; \hat{\text{sk}}_u; i'; G'; \mathbf{m}'; \mathbf{sk}'; a'; \mathbf{b}'; \alpha'; \hat{t}_u)$.
- Else, if $\hat{i}_u = R$: If there does not exist exactly one v such that $(\hat{\mathbf{m}}^u)_v \neq \perp$, output $\perp$. If $a' \leq t$ or $\alpha' = 1$, output $\perp$. Otherwise, output the message corresponding to the unique non-$\perp$ location: $(\hat{\mathbf{m}}^u)_{v^*}$.
- Else, output $\perp$.

The Simulator. As before the first the simulator first generates a non-aborting execution for each corrupt component, which will form the basis of the messages from honest parties. Additionally, in this case, the simulator selects a threshold block uniformly for each party's zone. Having chosen thresholds and an execution, simulation of $C_{\text{rand-fs}}$ proceeds identically to the previous protocol (Figure **??**).

Lemma 1. *For any probabilistic poly-time adversary, the simulator only needs to query the leakage oracle with probability at most δ.*

Proof. Recall that the simulator only needs to query the leakage oracle with respect to at most one party.

For any fixed network location, we will bound the probability that the simulator needs to call the leakage oracle for that location by δ/n. Then by a union bound, the probability that the simulator needs to call the leakage oracle for *any* node.

Recall that by lemma **??**, any non-aborting query graph induced by a threshold node must match the non-aborting execution exactly. As a consequence to get non-aborting output, the adversary must have run the protocol up to at least $r + 1$ rounds before the node's threshold. Thus, anything that happens before such a time will give no information about the threshold round (beyond whether it has or has not occurred yet). Additionally, by the time an adversary can learn whether the threshold follows its current block, it will be to late to execute a non-simulatable abort (outside of the corruption radius of the node).

If the first abort occurs in the i-th block of a location's zone, then the probability the adversary hits the chosen block is:

$$\underbrace{\left(1 - \frac{i-1}{n/\delta}\right)}_{\text{prob. threshold hasn't occured}} \cdot \underbrace{\left(\frac{1}{n/\delta - (i-1)}\right)}_{\text{cond. prob. of hitting relevant block}} = \frac{1}{n/\delta} = \frac{\delta}{n}$$

References

1. diaspora*: The online social world where you are in control.
2. A. Akavia, R. LaVigne, and T. Moran. Topology-hiding computation on all graphs. In J. Katz and H. Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 447–467. Springer, 2017.
3. A. Akavia and T. Moran. Topology-hiding computation beyond logarithmic diameter. In *To Appear: Advances in Cryptology - EUROCRYPT 2017*, 2017.
4. A. Beimel. On private computation in incomplete networks. *Distributed Computing*, 19(3):237–252, 2007.
5. A. Beimel and M. K. Franklin. Reliable communication over partially authenticated networks. *Theor. Comput. Sci.*, 220(1):185–210, 1999.
6. A. Beimel, A. Gabizon, Y. Ishai, E. Kushilevitz, S. Meldgaard, and A. Paskin-Cherniavsky. Non-interactive secure multiparty computation. In J. A. Garay and R. Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 387–404. Springer, 2014.

7. A. Beimel and L. Malka. Efficient reliable communication over partially authenticated networks. *Distributed Computing*, 18(1):1–19, 2005.
8. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Noncryptographic Fault-Tolerant Distributed Computations. In *Proc. 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 1988.
9. H. Bhurman, M. Christandl, F. Unger, S. Wehner, and A. Winter. Implications of superstrong nonlocality for cryptography. *Proceedings of the Royal Society A*, 462(2071):1919 – 1932.
10. M. Bläser, A. Jakoby, M. Liskiewicz, and B. Manthey. Private computation: k-connected versus 1-connected networks. *J. Cryptology*, 19(3):341–357, 2006.
11. 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 2015*, Lecture Notes in Computer Science, pages 356–376. Springer, 2013.
12. R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In J. H. Reif, editor, *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 494–503. ACM, 2002.
13. 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 2015 Conference on Innovations in Theoretical Computer Science*, ITCS '15, pages 153–162, New York, NY, USA, 2015. ACM.
14. N. Chandran, J. A. Garay, and R. Ostrovsky. Edge fault tolerance on sparse networks. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, pages 452–463, 2012.
15. N. Chandran, V. Goyal, and A. Sahai. New constructions for UC secure computation using tamper-proof hardware. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, pages 545–562, 2008.
16. H. Chang, R. Govindan, S. Jamin, S. . J. Shenker, and W. Willinger. Towards capturing representative AS-level Internet topolog ies. *Computer Networks*, 44(6), April 2004.
17. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.
18. D. Chaum, C. Crepeau, and I. Damgard. Multiparty Unconditionally Secure Protocols. In *Proc. 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 11–19, 1988.
19. A. Chiesa and E. Tromer. Proof-carrying data and hearsay arguments from signature cards. In A. C. Yao, editor, *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 310–331. Tsinghua University Press, 2010.
20. S. G. Choi, J. Katz, D. Schröder, A. Yerukhimovich, and H. Zhou. (efficient) universally composable oblivious transfer using a minimal number of stateless tokens. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 638–662, 2014.
21. B. Chor and E. Kushilevitz. A zero-one law for boolean privacy. *SIAM J. Discrete Math.*, 4(1):36–47, 1991.
22. R. Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *STOC*, pages 364–369, 1986.
23. R. Cleve and R. Impagliazzo. Martingales, collective coin flipping and discrete control processes˙ Unpublished, 1993.
24. D. Dachman-Soled, Y. Lindell, M. Mahmoody, and T. Malkin. On the black-box complexity of optimally-fair coin tossing. In *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, pages 450–467, 2011.

25. J. Deng, R. Han, and S. Mishra. Decorrelating wireless sensor network traffic to inhibit traffic analysis attacks. *Pervasive and Mobile Computing*, 2(2):159–186, 2006.
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. *J. ACM*, 40(1):17–47, 1993.
28. C. Dwork, D. Peleg, N. Pippenger, and E. Upfal. Fault tolerance in networks of bounded degree. *SIAM J. Comput.*, 17(5):975–988, 1988.
29. B. Fisch, D. Freund, and M. Naor. Physical zero-knowledge proofs of physical properties. In J. A. Garay and R. Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 313–336. Springer, 2014.
30. B. A. Fisch, D. Freund, and M. Naor. Secure physical computation using disposable circuits. In Y. Dodis and J. B. Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 182–198. Springer, 2015.
31. R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, and T. Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In M. Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 258–277. Springer, 2004.
32. A. Glaser, B. Barak, and R. Goldston. A zero-knowledge protocol for nuclear warhead verification. *Nature*, 510:497–502, 2004.
33. 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.
34. O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.
35. S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. One-time programs. In D. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2008.
36. 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, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 575–591, 2013.
37. V. Goyal, Y. Ishai, A. Sahai, R. Venkatesan, and A. Wadia. Founding cryptography on tamper-proof hardware tokens. In *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, pages 308–326, 2010.
38. S. Halevi, Y. Ishai, A. Jain, E. Kushilevitz, and T. Rabin. Secure multiparty computation with general interaction patterns. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, ITCS '16, pages 157–168, New York, NY, USA, 2016. ACM.
39. S. Halevi, Y. Lindell, and B. Pinkas. Secure computation on the web: Computing without simultaneous interaction. In P. Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 132–150. Springer, 2011.
40. C. Hazay and Y. Lindell. Constructions of truly practical secure protocols using standard-smartcards. In P. Ning, P. F. Syverson, and S. Jha, editors, *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, pages 491–500. ACM, 2008.

41. M. Hinkelmann and A. Jakoby. Communications in unknown networks: Preserving the secret of topology. *Theoretical Computer Science*, 384(2–3):184–200, 2007. Structural Information and Communication Complexity (SIROCCO 2005).
42. M. Hirt, U. Maurer, D. Tschudi, and V. Zikas. Network-hiding communication and applications to multi-party protocols. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 335–365, 2016.
43. D. Hofheinz, J. Muller-Quade, and D. Unruh. Universallly composalbe zero-knowledge arguments and commitments from signature cards. In *5th Central European Conference on Cryptology*, 2005.
44. Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In D. Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.
45. P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk. Enhancing source-location privacy in sensor network routing. In *25th International Conference on Distributed Computing Systems (ICDCS 2005), 6-10 June 2005, Columbus, OH, USA*, pages 599–608, 2005.
46. J. Katz. Universally composable multi-party computation using tamper-proof hardware. In *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, pages 115–128, 2007.
47. J. Kilian. A general completeness theorem for two-party games. In C. Koutsougeras and J. S. Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 553–560. ACM, 1991.
48. M. V. N. A. Kumar, P. R. Goundan, K. Srinathan, and C. P. Rangan. On perfectly secure cmmunication over arbitrary networks. In *PODC*, pages 193–202, 2002.
49. E. Kushilevitz. Privacy and communication complexity. *SIAM J. Discrete Math.*, 5(2):273–284, 1992.
50. T. Moran and M. Naor. Basing cryptographic protocols on tamper-evident seals. *Theor. Comput. Sci.*, 411(10):1283–1310, 2010.
51. T. Moran, M. Naor, and G. Segev. An optimally fair coin toss. *J. Cryptology*, 29(3):491–513, 2016.
52. T. Moran, I. Orlov, and S. Richelson. Topology-hiding computation. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015*, volume 9014 of *Lecture Notes in Computer Science*, pages 169–198. Springer, 2015.
53. M. K. Reiter and A. D. Rubin. Anonymous web transactions with crowds. *Commun. ACM*, 42(2):32–38, 1999.
54. N. T. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *Proceedings of SIGCOMM 2002*, 2002.
55. P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *1997 IEEE Symposium on Security and Privacy, May 4-7, 1997, Oakland, CA, USA*, pages 44–54, 1997.
56. A. C. Yao. Protocols for secure computations. In *Proc. 23rd IEEE Symposium on Foundations of Computer Scien ce (FOCS)*, pages 160–164, 1982.