Topology-Hiding Computation on All Graphs

Adi Akavia[1(✉)], Rio LaVigne[2(✉)], and Tal Moran[3(✉)]

[1] The Academic College of Tel-Aviv Jaffa, Tel Aviv-Yafo, Israel
akavia@mta.ac.il
[2] MIT, Cambridge, USA
rio@mit.edu
[3] IDC Herzliya, Herzliya, Israel
talm@idc.ac.il

Abstract. A distributed computation in which nodes are connected by a partial communication graph is called *topology-hiding* if it does not reveal information about the graph beyond what is revealed by the output of the function. Previous results have shown that topology-hiding computation protocols exist for graphs of constant degree and logarithmic diameter in the number of nodes [Moran-Orlov-Richelson, TCC'15; Hirt et al., Crypto'16] as well as for other graph families, such as cycles, trees, and low circumference graphs [Akavia-Moran, Eurocrypt'17], but the feasibility question for general graphs was open.

In this work we positively resolve the above open problem: we prove that topology-hiding MPC is feasible for *all* graphs under the Decisional Diffie-Hellman assumption.

Our techniques employ random-walks to generate paths covering the graph, upon which we apply the Akavia-Moran topology-hiding broadcast for chain-graphs (paths). To prevent topology information revealed by the random-walk, we design multiple random-walks that, together, are locally identical to receiving at each round a message from each neighbors and sending back processed messages in a randomly permuted order.

1 Introduction

The beautiful theory of secure multiparty computation (MPC) enables multiple parties to compute an arbitrary function of their inputs without revealing anything but the

A. Akavia—Work partly supported by the ERC under the EU's Seventh Framework Programme (FP/2007–2013) ERC Grant Agreement no. 307952.

R. LaVigne—This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 1122374. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of the National Science Foundation. Research also supported in part by NSF Grants CNS-1350619 and CNS-1414119, and by the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under contracts W911NF-15-C-0226 and W911NF-15-C-0236.

T. Moran—Supported by ISF grant no. 1790/13.

J. Katz and H. Shacham (Eds.): CRYPTO 2017, Part I, LNCS 10401, pp. 447–467, 2017.
DOI: 10.1007/978-3-319-63688-7_15

function's output [10,11,24]. In the original definitions and constructions of MPC, the participants were connected by a full communication graph (a broadcast channel and/or point-to-point channels between every pair of parties). In real-world settings, however, the actual communication graph between parties is usually not complete, and parties may be able to communicate directly with only a subset of the other parties. Moreover, in some cases the graph itself is sensitive information (e.g., if you communicate directly only with your friends in a social network).

A natural question is whether we can successfully perform a joint computation over a partial communication graph while revealing no (or very little) information about the graph itself. In the information-theoretic setting, in which a variant of this question was studied by Hinkelman and Jakoby [15], the answer is mostly negative. The situation is better in the computational setting. Moran et al. showed that topology-hiding computation *is* possible against static, semi-honest adversaries [21]; followed by constructions with improved efficiency that make only black-box use of underlying primitives [16]. However, all these protocol are restricted to communication graphs with *small diameter*. Specifically, these protocols address networks with diameter $D = O(\log n)$, logarithmic in the number of nodes n (where the diameter is the maximal distance between two nodes in the graph). Akavia and Moran [1] showed that topology hiding computation is feasible also for *large diameter* networks of certain forms, most notably, cycles, trees, and low circumference graphs.

However, there are natural network topologies not addressed by the above protocols [1,16,21]. They include, for example, wireless and ad-hoc sensor networks, as in [8,23]. The topology in these graphs is modeled by random geometric graphs [22], where, with high probability, the diameter and the circumference are simultaneously large [3,9]. These qualities exclude the use of all aforementioned protocols. So, the question remained:

Is topology hiding MPC feasible for every network topology?

1.1 Our Results

In this work we prove that topology hiding MPC is feasible for *every* network topology under the Decisional Diffie-Hellman (DDH) assumption, thus positively resolving the above open problem. The adversary is static and semi-honest as in the prior works [1,16,21].[1] Our protocol also fits a stronger definition of security than that from prior works: instead of allowing the adversary to know who his neighbors are, he only gets pseudonyms; importantly, an adversary cannot tell if two nodes he controls share an honest neighbor. This stronger definition is elaborated on in the full version of this paper.

Theorem 1 (Topology-hiding broadcast for all network topologies – informal). *There exists a topology-hiding protocol realizing the broadcast functionality on every network topology (under DDH assumption and provided the parties are given an upper-bound n on the number of nodes).*

The formal theorem is stated and proved in Sect. 3.3.

[1] Moran et al. [21] consider also a fail-stop adversary for proving an impossibility result.

As in [1,16,21], given a *topology-hiding broadcast* for a point-to-point channels network, we can execute on top of it any MPC protocol from the literature that is designed for networks with broadcast channels; the resulting protocol remains topology-hiding. Put together with the existence of secure MPC for all efficiently computable functionalities (assuming parties have access to a broadcast channel and that public key encryption exist) [10,11,24], we conclude that *topology-hiding MPC* for all efficiently computable functionality and all networks topologies (assuming a certain type of public key encryption exists).

1.2 High-Level Overview of Our Techniques

Our main innovation is the use of *random walks* on the network graph for specifying a path, and then viewing this path as a chain-graph and employing the topology-hiding broadcast for chains of Akavia and Moran [1].

A challenge we face is that the walk itself may reveal topology information. For example, a party can deduce the graph commute-time from the number of rounds before a returning visit by the walk. We therefore hide the random-walk by using multiple simultaneous random-walks (details below). The combination of all our random-walks obeys a simple communication structure: at every round each node receives an incoming message from each of its neighbors, randomly permutes the messages, and sends them back.

To give more details on our protocol, let us first recall the Akavia-Moran protocol for chain-graphs. The Akavia-Moran protocol proceeds in two phases, a forward and a backward phase. In the forward phase, messages are passed forward on the chain, where each node adds its own encryption layer and computes the OR of the received message with its bit using homomorphic multiplication (with proper re-randomizing). In the backward phase, the messages are passed backward along the same path, where each node deletes its encryption layer. At the end of the protocol, the starting node receives the plaintext value for the OR of all input bits. This protocol is augmented to run n instances simultaneously; each node initiates an execution of the protocol while playing the role of the first node. So, by the end of the protocol, each node has the OR of all bits, which will be equal to the broadcast bit. Intuitively, this achieves topology-hiding because at each step, every node receives an encrypted message and public key. An encryption of zero is indistinguishable from an encryption of 1, and so each node's view is indistinguishable from every other view.

We next elaborate on how we define our multiple random walks, focusing on two viewpoints: the viewpoint of a node, and the viewpoint of a message. We use the former to argue security, and the latter to argue correctness.

From the point of view of a node v with d neighbors, the random walks on the forward-phase are specified by choosing a sequence of independent random permutations $\pi_t : [d] \to [d]$, where in each forward-phase round t, the node forwards messages received from neighbor i to neighbor $\pi_t(i)$ (after appropriate processing of the message, as discussed above). The backward-phase follows the reverse path, sending incoming message from neighbor j to neighbor $i = \pi_t^{-1}(j)$, where t is the corresponding round in the forward-phase. Furthermore, recall that all messages are encrypted under semantically-secure encryption. This fixed communication pattern together with

the semantic security of the messages content leads to the topology-hiding property of our protocol.

From the point of view of a message, at each round of the forward-phase the message is sent to a uniformly random neighbor. Thus, the path the message goes through is a random-walk on the graph.[2] A sufficiently long random walk covers the entire graph with overwhelming probability. In this case, the output is the OR of the inputs bits of *all* graph nodes, and correctness is guaranteed.

1.3 Related Work

Topology Hiding in Computational Settings. Figure 1 compares our results to the previous results on topology hiding computation and specifies, for each protocol, the classes of graphs for which it is guaranteed to run in polynomial time.

The first result was a feasibility result and was the work of Moran et al. [21] from 2015. Their result was a broadcast protocol secure against static, semi-honest adversaries, and a protocol against failstop adversaries that do not disconnect the graph. However, their protocol is restricted to communication graphs with diameter logarithmic in the total number of parties.

Graphs families	[17,22]	[1]	[This Work]
Log diameter constant degree	+	–	+
Cycles, trees	–	+	+
Log circumference	–	+	+
Log diameter super-constant degree	–	–	+
Regular graphs	–	–	+
Arbitrary graphs	–	–	+

Fig. 1. Rows correspond to graph families; columns corresponds to prior works in the first two columns and to this work in last the column. A +/− mark for graph x and work y indicates that a topology hiding protocol is given/not-given in work y for graph x.

The main idea behind their protocol is a series of nested multiparty computations, in which each node is replaced by a secure computation in its local neighborhood that simulates that node. The drawback is that in order to get full security, this virtualization needs to extend to the entire graph, but the complexity of the MPC grows exponentially with the size of the neighborhood.

Our work is also a feasibility result, but instead builds on a protocol much more similar to the recent Akavia-Moran paper [1], which takes a different approach. They employ ideas from cryptographic voting literature, hiding the order of nodes in the cycle by "mixing" encrypted inputs before decrypting them and adding layers of public keys to the encryption at each step. In this work, we take this layer-adding approach and

[2] We remark that the multiple random-walks are not independent; we take this into account in our analysis.

apply it to random walks over all kinds of graphs instead of deterministically figuring out the path beforehand.

Other related works include a work by Hirt et al. [16], which describes a protocol that acheives better efficiency than [21], but as it uses similar tactics, is still restricted to network graphs with logarithmic diameter. Addressing a problem different from topology-hiding, the work by Chandran et al. [7] reduces communication complexity of secure MPC by allowing each party to communicate with a small (sublinear in the number of parties) number of its neighbors.

Topology Hiding in Information Theoretic Settings. Hinkelmann and Jakoby [15] considered the question of topology-hiding secure computation, but focused on the information theoretic setting. Their main result was negative: any MPC protocol in the information-theoretic setting inherently leaks information about the network graph to an adversary. However, they also show that the only information we need to leak is the routing table: if we leak the routing table beforehand, then one can construct an MPC protocol which leaks no further information.

Secure Multiparty Computation with General Interaction Patterns. Halevi et al. [13] presented a unified framework for studying secure MPC with arbitrary restricted interaction patterns, generalizing models for MPC with specific restricted interaction patterns [4,12,14]. Their goal is not topology hiding, however. Instead, they ask the question of when is it possible to prevent an adversary from learning the output to a function on several inputs. They started by observing that an adversary controlling the final players $P_i, \cdots, P_n$ in the interaction pattern can learn the output of the computed function on several inputs because the adversary can rewind and execute the protocol on any possible party values $x_i, \cdots, x_n$. This model allows complete knowledge on the underlying interaction pattern (or as in our case, the graph).

1.4 Organization of Paper

In Sect. 2 we describe our adversarial model and introduce some notation. In Sect. 2.5 we detail the special properties we require from the encryption scheme that we use in our cycle protocol, and show how it can be instantiated based on DDH. In Sect. 3, we explain our protocol for topology-hiding broadcast on general graphs and prove its completeness and security. Then, in Sect. 4, we go over a time and communication tradeoff, and explain how we can optimize our protocol with respect to certain classes of graphs. Finally, in Sect. 5, we conclude and discuss future work.

2 Preliminaries

2.1 Computation and Adversarial Models

We model a network by an undirected graph $G = (V, E)$ that is not fully connected. We consider a system with n parties denoted $P_1, \ldots, P_n$, where n is upper bounded by $\text{poly}(\kappa)$ and κ is the security parameter. We identify V with the set of parties $\{P_1, \ldots, P_n\}$.

We consider a static and computationally bounded (PPT) adversary that controls some subset of parties (any number of parties). That is, at the beginning of the protocol, the adversary corrupts a subset of the parties and may instruct them to deviate from the protocol according to the corruption model. Throughout this work, we consider only semi-honest adversaries. In addition, we assume that the adversary is rushing; that is, in each round the adversary sees the messages sent by the honest parties before sending the messages of the corrupted parties for this round. For general MPC definitions including in-depth descriptions of the adversarial models we consider, see [10].

2.2 Notation

In this section, we describe our common notation conventions for both graphs and for our protocol.

Graph Notation

Let $G = (V, E)$ be an undirected graph. For every $v \in V$, we define the neighbors of v as $\mathcal{N}(v) = \{w : (v, w) \in E\}$ and will refer to the degree of v as $d_v = |\mathcal{N}(v)|$.

Protocol Notation

Our protocol will rely on generating many public-secret key pairs, and ciphertexts at each round. In fact, each node will produce a public-secret key pair for each of its neighbors at every timestep. To keep track of all these, we introduce the following notation. Let $pk^{(t)}_{i\to d}$ represent the public key created by node i to be used for neighbor d at round t; $sk^{(t)}_{i\to d}$ is the corresponding secret key. Ciphertexts are labeled similarly: $c^{(t)}_{d\to i}$, is from neighbor d to node i.

2.3 UC Security

As in [21], we prove security in the UC model [5]. If a protocol is secure in the UC model, it can be composed with other protocols without compromising security, so we can use it as a subprotocol in other constructions. This is critical for constructing topology-hiding MPC based on broadcast—broadcast is used as a sub-protocol.

A downside of the UC model is that, against general adversaries, it requires setup. However, setup is not necessary against semi-honest adversaries that must play according to the rules of the protocol. Thus, we get a protocol that is secure in the plain model, without setup. For details about the UC framework, we refer the reader to [5].

2.4 Simulation-Based Topology Hiding Security

Here we will review the model for defining simulation-based topology hiding computation, as proposed by [21], in the UC framework.

The UC model usually assumes all parties can communicate directly with all other parties. To model the restricted communication setting, [21] define the $\mathcal{F}_{\text{graph}}$-hybrid model, which employs a special "graph party," P_{graph}. Figure 2 shows $\mathcal{F}_{\text{graph}}$'s functionality: at the start of the functionality, $\mathcal{F}_{\text{graph}}$ receives the network graph from P_{graph}, and then outputs, to each party, that party's neighbors. Then, $\mathcal{F}_{\text{graph}}$ acts as an "ideal

channel" for parties to communicate with their neighbors, restricting communications to those allowed by the graph.

Since the graph structure is an input to one of the parties in the computation, the standard security guarantees of the UC model ensure that the graph structure remains hidden (since the only information revealed about parties' inputs is what can be computed from the output). Note that the P_{graph} party serves only to specify the communication graph, and does not otherwise participate in the protocol.

Participants/Notation:
This functionality involves all the parties $P_1, \ldots, P_n$ 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 $\mathcal{N}(v)$ to each P_v.
Communication Phase:
Inputs: $\mathcal{F}_{\text{graph}}$ receives from a party P_v a destination/data pair (w, m) where $w \in \mathcal{N}(v)$ and m is the message P_v wants to send to P_w.
Output: $\mathcal{F}_{\text{graph}}$ gives output (v, m) to P_w indicating that P_v sent the message m to P_w.

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

The initialization phase of $\mathcal{F}_{\text{graph}}$ provides local information about the graph to every corrupted party, and so both ideal-world and real-world adversaries get access to this information. This information is independent of the functionality we are trying to implement, but always present. So we will isolate it in the functionality $\mathcal{F}_{\text{graphInfo}}$ which contains only the initialization phase of $\mathcal{F}_{\text{graph}}$, and then, for any functionality $\mathcal{F}$, we compose $\mathcal{F}$ with $\mathcal{F}_{\text{graphInfo}}$, writing $(\mathcal{F}_{\text{graphInfo}} \| \mathcal{F})$ as the "composed functionality." Now we can define topology-hiding MPC in the UC framework:

Definition 2. *We say that a protocol* Π *securely realizes a functionality* $\mathcal{F}$ hiding topology *if it UC-realizes* $(\mathcal{F}_{graphInfo} \| \mathcal{F})$ *in the* $\mathcal{F}_{graph}$*-hybrid model.*

This definition also captures functionalities that depend on the structure of the graph, like shortest path or determining the length of the longest cycle.

Broadcast Functionality, $\mathcal{F}_{\text{Broadcast}}$

In accordance with this definition, we need to define an ideal functionality of broadcast, denoted $\mathcal{F}_{\text{Broadcast}}$, shown in Fig. 3. We will prove that a simulator only with knowledge

Participants/Notation:
This functionality involves all the parties $P_1, \ldots, P_n$.
Inputs: The broadcasting party P_i receives a bit $b \in \{0, 1\}$.
Outputs: All parties $P_1, \ldots, P_n$ receive output b.

Fig. 3. The functionality $\mathcal{F}_{\text{Broadcast}}$.

of the output of $\mathcal{F}_{\text{Broadcast}}$ and knowledge of the local topology of the adversarially chosen nodes Q can produce a transcript to nodes in Q indistinguishable from running our protocol.

2.5 Privately Key-Commutative and Rerandomizable Encryption

As in [1], we require a public key encryption scheme with the properties of being *homomorphic* (with respect to OR in our case), *privately key-commutative*, and *re-randomizable*. In this section we first formally define the properties we require, and then show how they can be achieved based on the Decisional Diffie-Hellman assumption.

We call an encryption scheme satisfying the latter two properties, i.e., privately key-commutative and re-randomizable, a *PKCR-enc*;

Required Properties

Let $\mathsf{KeyGen} : \{0,1\}^* \mapsto \mathcal{PK} \times \mathcal{SK}, \mathsf{Enc} : \mathcal{PK} \times \mathcal{M} \times \{0,1\}^* \mapsto C, \mathsf{Dec} : \mathcal{SK} \times C \mapsto \mathcal{M}$ be the encryption scheme's key generation, encryption and decryption functions, respectively, where $\mathcal{PK}$ is the space of public keys, $\mathcal{SK}$ the space of secret keys, $\mathcal{M}$ the space of plaintext messages and C the space of ciphertexts.

We will use the shorthand $[m]_k$ to denote an encryption of the message m under public-key k. We assume that for every secret key $sk \in \mathcal{SK}$ there is associated a single public key $pk \in \mathcal{PK}$ such that (pk, sk) are in the range of KeyGen. We slightly abuse notation and denote the public key corresponding to sk by $pk(sk)$.

Privately Key-Commutative

The set of public keys $\mathcal{PK}$ form an abelian (commutative) group. We denote the group operation $\circledast$. Given any $k_1, k_2 \in \mathcal{PK}$, there exists an efficient algorithm to compute $k_1 \circledast k_2$. We denote the inverse of k by k^{-1} (i.e., $k^{-1} \circledast k$ is the identity element of the group). Given a secret key sk, there must be an efficient algorithm to compute the inverse of its public key $(pk(sk))^{-1}$.

There exist a pair of algorithms $\mathsf{AddLayer} : C \times \mathcal{SK} \mapsto C$ and $\mathsf{DelLayer} : C \times \mathcal{SK} \mapsto C$ that satisfy:

1. For every public key $k \in \mathcal{PK}$, every message $m \in \mathcal{M}$ and every ciphertext $c = [m]_k$,
$$\mathsf{AddLayer}(c, sk) = [m]_{k \circledast pk(sk)} .$$
2. For every public key $k \in \mathcal{PK}$, every message $m \in \mathcal{M}$ and every ciphertext $c = [m]_k$,
$$\mathsf{DelLayer}(c, sk) = [m]_{k \circledast (pk(sk))^{-1}} .$$

We call this *privately* key-commutative since adding and deleting layers both require knowledge of the secret key.

Note that since the group $\mathcal{PK}$ is commutative, adding and deleting layers can be done in any order.

Rerandomizable

We require that there exists a ciphertexts "re-randomizing" algorithm $\mathsf{Rand} : C \times \mathcal{PK} \times \{0,1\}^* \mapsto C$ satisfying the following:

1. *Randomization:* For every message $m \in \mathcal{M}$, every public key $pk \in \mathcal{PK}$ and ciphertext $c = [m]_{pk}$, the distributions $(m, pk, c, \mathsf{Rand}(c, pk, U^*))$ and $(m, pk, c, \mathsf{Enc}_{pk}(m; U^*))$ are computationally indistinguishable.
2. *Neutrality:* For every ciphertext $c \in C$, every secret key $sk \in \mathcal{SK}$ and every $r \in \{0,1\}^*$,
$$\mathsf{Dec}_{sk}(c) = \mathsf{Dec}_{sk}(\mathsf{Rand}(c, pk(sk), r)).$$

Furthermore, we require that public-keys are "re-randomizable" in the sense that the product $k \circledast k'$ of an arbitrary public key k with a public-key k' generated using KeyGen is computationally indistinguishable from a fresh public-key generated by KeyGen.

Homomorphism

We require that the message space $\mathcal{M}$ forms a group with operation denoted $\cdot$, and require that the encryption scheme is homomorphic with respect this operation $\cdot$ in the sense that there exists an efficient algorithm $\mathsf{hMult} : C \times C \mapsto C$ that, given two ciphertexts $c = [m]_{pk}$ and $c' = [m']_{pk}$, returns a ciphertext $c'' \leftarrow \mathsf{hMult}(c, c')$ s.t. $\mathsf{Dec}_{sk}(c'') = m \cdot m'$ (for sk the secret-key associated with public-key pk).

Notice that with this operation, we can homomorphically raise any ciphertext to any power via repeated squaring. We will call this operation hPower.

Homomorphic OR

This feature is built up from the re-randomizing and the homomorphism features. One of the necessary parts of our protocol for broadcast functionality is to have a homomorphic OR. We need this operation not to reveal if it is ORing two 1's or one 1 at decryption. So, following [1], first we define an encryption of 0 to be an encryption of the identity element in $\mathcal{M}$ and an encryption of 1 to be an encryption of any other element. Then, we define HomOR so that it re-randomizes encryptions of 0 and 1 by raising ciphertexts to a random power with hPower.

---

function $\mathsf{HomOR}(c, c', pk, r = (r, r'))$ // r is randomness.
  $\hat{c} \leftarrow \mathsf{hPower}(c, r, pk)$ and $\hat{c}' \leftarrow \mathsf{hPower}(c', r', pk)$
  return $\mathsf{Rand}(\mathsf{hMult}(\hat{c}, \hat{c}''), pk)$
end function

Claim. Let $\mathcal{M}$ have prime order p, where $1/p$ is negligible in the security parameter, and $M, M' \in \{0,1\}$ be messages with corresponding ciphertexts c and c' under public key pk. The distribution $(c, c', pk, M, M', \mathsf{Enc}(M \vee M', pk; U^*))$ is computationally indistinguishable from $(c, c', pk, M, M', \mathsf{HomOR}(c, c', pk; U^*))$.

Proof. We will go through three cases for values of M and M': first, when $M = M' = 0$; second when $M = 1$ and $M' = 0$; and third when $M = 1$ and $M' = 1$. The case $M = 0$ and $M' = 1$ is handled by the second case.

– Consider when $M = M' = 0$. Note that $1_{\mathcal{M}}$ is the group element in $\mathcal{M}$ that encodes 0, so an encryption of 0 is represented by an encryption of the identity element,

$m = m' = 1_{\mathcal{M}}$, of $\mathcal{M}$. Consider c_0 and c'_0 both encryptions of $1_{\mathcal{M}}$. After hPower, both $\hat{c}_0$ and $\hat{c}'_0$ are still encryptions of $1_{\mathcal{M}}$. hMult then produces an encryption of $1_{\mathcal{M}} \cdot 1_{\mathcal{M}} = 1_{\mathcal{M}}$, and Rand makes that ciphertext indistinguishable to a fresh encryption of $1_{\mathcal{M}}$. We have proved our first case.

- Next, let c_0 be an encryption of 0 and c'_1 be an encryption of 1. In this case, 0 is represented again by $1_{\mathcal{M}}$, but c'_1 is represented by some $m' \xleftarrow{\$} \mathcal{M}$ (with all but negligible probability $m' \neq 1$). After hPower, $\hat{c}_0$ still encrypts $1_{\mathcal{M}}$, but $\hat{c}'_1$ encrypts $\hat{m} = m'^{r'}$ for some $r' \xleftarrow{\$} \mathbb{Z}_p$. hMult yeilds an encryption of $\hat{m}$ and Rand makes a ciphertext computationally indistinguishable from a fresh encryption of $\hat{m}$. Since $\mathcal{M}$ has prime order p and $r' \xleftarrow{\$} \mathbb{Z}_p$, as long as $m' \neq 1$, m'^r is uniformly distributed over $\mathcal{M}$, and so computationally has a distribution indistinguishable to a fresh encryption of the boolean message 1.
- Finally, let c_1 and c'_1 both be encryptions of 1: c_1 encrypts $m \xleftarrow{\$} \mathcal{M}$ and c'_1 encrypts $m' \xleftarrow{\$} \mathcal{M}$. We will go through the same steps to have at the end, a ciphertext computationally indistinguishable[3] from a fresh encryption of $m^r \cdot m'^{r'}$ for $r, r' \xleftarrow{\$} \mathbb{Z}_p$. Again because the order of $\mathcal{M}$ is prime, $m^r \cdot m'^{r'}$ is uniformly distributed over $\mathbb{Z}_p$, and so the resulting ciphertext looks like a fresh encryption of 1. □

This claim means that we cannot tell how many times 1 or 0 has been OR'd together during an or-and-forward type of protocol. This will be critical in our proof of security.

Instantiation of OR-Homomorphic PKCR-enc Under DDH

We use standard ElGamal, augmented by the additional required functions. The KeyGen, Dec and Enc functions are the standard ElGamal functions, except that to obtain a one-to-one mapping between public keys and secret keys, we fix the group G and the generator g, and different public keys vary only in the element $h = g^x$. Below, g is always the group generator. The Rand function is also the standard rerandomization function for ElGamal:

function RAND($c = (c_1, c_2), pk, r$)
  return $(c_1 \cdot g^r, pk^r \cdot c_2)$
end function

We use the shorthand notation of writing Rand(c, pk) when the random coins r are chosen independently at random during the execution of Rand. We note that the distribution of public-keys outputted by KeyGen is uniform, and thus the requirement for "public-key rerandomization" indeed holds. ElGamal public keys are already defined over an abelian group, and the operation is efficient. For adding and removing layers, we define:

[3] In our definition of a PKCR encryption scheme, Rand is only required to be computationally randomizing, which carries over in our distribution of homomorphically-OR'd ciphertexts. However, ElGamal's re-randomization function is distributed statistically close to a fresh ciphertext, and so our construction will end up having HomOR be identically distributed to a fresh encryption of the OR of the bits.

function AddLayer$(c = (c_1, c_2), sk)$
  return $(c_1, c_2 \cdot c_1^{sk})$
end function
function DelLayer$(c = (c_1, c_2), sk)$
  return $(c_1, c_2 / c_1^{sk})$
end function

Every ciphertext $[m]_{pk}$ has the form $(g^r, pk^r \cdot m)$ for some element $r \in \mathbb{Z}_{ord(g)}$. So

$$\mathsf{AddLayer}\left([m]_{pk}, sk'\right) = (g^r, pk^r \cdot m \cdot g^{r \cdot sk'}) = (g^r, pk^r \cdot (pk')^r \cdot m) = (g^r, (pk \cdot pk')^r \cdot m) = [m]_{pk \cdot pk'}.$$

It is easy to verify that the corresponding requirement is satisfied for DelLayer as well.

ElGamal message space already defined over an abelian group with homomorphic multiplication, specifically:

function hMult$(c = (c_1, c_2), c' = (c'_1, c'_2))$
  return $c'' = (c_1 \cdot c'_1, c_2 \cdot c'_2)$
end function

Recalling that the input ciphertext have the form $c = (g^r, pk^r \cdot m)$ and $c' = (g^{r'}, pk^{r'} \cdot m')$ for messages $m, m' \in \mathbb{Z}_{ord(g)}$, it is easy to verify that decrypting the ciphertext $c'' = (g^{r+r'}, pk^{r+r'} \cdot m \cdot m')$ returned from hMult yields the product message $\mathsf{Dec}_{sk}(c'') = m \cdot m'$.

Finally, to obtain a negligible error probability in our broadcast protocols, we take G a prime order group of size satisfying that $1/|G|$ is negligible in the security parameter κ. With this property and valid Rand and hMult operations, we get hPower and hence HomOR with ElGamal.

3 Topology Hiding Broadcast Protocol for General Graphs

In this section, we describe how our protocol works and prove that it is complete and secure.

The protocol (see Protocol 1) is composed of two phases: an aggregate (forward) phase and a decrypt (backward) phase. In the aggregate phase messages traverse a random walk on the graph where each of the passed-through nodes adds a fresh encryption layer and homomorphically ORs the passed message with its bit. In the decrypt phase, the random-walk is traced back where each node deletes the encryption layer it previously added. At the end of the backward phase, the node obtains the plaintext value of the OR of all input bits. The protocol executes simultaneous random walks, locally defined at each node v with d neighbors by a sequence of permutations $\pi_t \colon [d] \to [d]$ for each round t, so that at round t of the forward phase messages received from neighbor i are forwarded to neighbor $\pi_t(i)$, and at the backward phase messages received from neighbor j are sent back to neighbor $\pi_t^{-1}(j)$.

3.1 Proof of Completeness

The main idea of the protocol is that we take a random walk around the graph, or-ing bits as we go, and hopefully by the time we start walking backwards along that path

Protocol 1. Topology-hiding broadcast for general graphs. Inputs parameters: n is the number of nodes; $2^{-\tau}$ the failure probability; d_i the degree of node i; and b_i the input bit of node i. See Sect. 2.2 for an explanation of notation.

procedure BROADCAST$((n, \tau, d_i, b_i))$
  // The number of steps we take in our random walk will be T.
  $T \leftarrow \tau \cdot 8n^3$
  Generate $T \cdot d_i$ key pairs: for $t \in [T]$ and $d \in [d_i]$, generate pair $(pk^{(t)}_{i\to d}, sk^{(t)}_{i\to d}) \leftarrow \mathsf{KeyGen}(1^\kappa)$.
  Generate $T-1$ random permutations on d_i elements $\{\pi_1, \cdots, \pi_{T-1}\}$. Let π_T be the identity permutation.
  // **Aggregate phase**.
  For all $d \in [d_i]$, send to neighbor d the ciphertext $[b_i]_{pk^{(1)}_{i\to d}}$ and the public key $pk^{(1)}_{i\to d}$.
  for $t = 1$ to $T-1$ **do**
    for Neighbors $d \in [d_i]$ **do**
      Wait to receive ciphertext $c^{(t)}_{d\to i}$ and public key $k^{(t)}_{d\to i}$.
      Let $d' \leftarrow \pi_t(d)$.
      Compute $k^{(t+1)}_{i\to d'} = k^{(t)}_{d\to i} \circledast pk^{(t+1)}_{i\to d'}$.
      Compute $\hat{c}^{(t+1)}_{i\to d} \leftarrow \mathsf{AddLayer}\left(c^{(t)}_{d\to i}, sk^{(t+1)}_{i\to d'}\right)$ and $[b_i]_{k^{(t+1)}_{i\to d'}}$.
      Compute $c^{(t+1)}_{i\to d'} \leftarrow \mathsf{HomOR}\left([b_i]_{k^{(t+1)}_{i\to d'}}, \hat{c}^{(t+1)}_{i\to d'}\right)$.
      Send $c^{(t+1)}_{i\to d'}$ and $k^{(t+1)}_{i\to d'}$ to neighbor d'.
    end for
  end for
  Wait to receive $c^{(T)}_{d\to i}$ and $k^{(T)}_{d\to i}$ from each neighbor $d \in [d_i]$.
  Compute $[b_i]_{k^{(T)}_{d\to i}}$ and let $e^{(T)}_{d\to i} \leftarrow \mathsf{HomOR}\left(c^{(T)}_{d\to i}, [b_i]_{k^{(T)}_{d\to i}}\right)$
  // **Decrypt phase**.
  for $t = T$ to 1 **do**
    For each $d \in [d_i]$, send $e^{(t)}_{i\to d'}$ to $d' = \pi_t^{-1}(d)$. // Passing back.
    for $d \in [d_i]$ **do**
      Wait to receive $e^{(t)}_{d\to i}$ from neighbor d.
      Compute $d' \leftarrow \pi_t^{-1}(d)$.
      $e^{(t-1)}_{i\to d'} \leftarrow \mathsf{DelLayer}\left(e^{(t)}_{d\to i}, sk^{(t)}_{i\to d'}\right)$ // If $t = 1$, DelLayer decrypts..
    end for
  end for
  // **Produce output bit**.
  $b \leftarrow \bigvee_{d\in[d_i]} e^{(0)}_{i\to d}$.
  Output b.
end procedure

we have reached all of the nodes. We will rely on the following definition and theorem from Mitzenmacher and Upfal's book (see Chap. 5) [20].

Definition 3 (Cover time). *The cover time of a graph $G = (V, E)$ is the maximum over all vertices $v \in V$ of the expected time to visit all of the nodes in the graph by a random walk starting from v.*

Theorem 4 (Cover time bound). *The cover time of any connected, undirected graph $G = (u, v)$ is bounded above by $4nm \leq 4n^3$.*

Corollary 5. *Let $\mathcal{W}(u, \tau)$ be a random variable whose value is the set of nodes covered by a random walk starting from u and taking $\tau \cdot (8n^3)$ steps. We have*

$$\Pr_{\mathcal{W}}[\mathcal{W}(u, \tau) = V] \geq 1 - \frac{1}{2^\tau}.$$

Proof. First, consider a random walk that takes t steps to traverse a graph. Theorem 4 tells us that we expect $t \leq 4n^3$, and so by a Markov bound, we have

$$\Pr\left[t \geq 2 \cdot (4n^3)\right] \leq \frac{1}{2}$$

Translating this into our notation, for any node $u \in G$, $\Pr[\mathcal{W}(u, 1) = V] \geq \frac{1}{2}$.

We can represent $\mathcal{W}(u, \tau)$ as a union of τ random walks, each of length $8n^3$: $\mathcal{W}(u_1 = u, 1) \cup \mathcal{W}(u_2, 1) \cup \cdots \cup \mathcal{W}(u_\tau, 1)$, where u_i is the node we have reached at step $i \cdot 8n^3$ (technically, u_i is a random variable, but the specific node at which we start each walk will not matter). $\mathcal{W}(u, \tau)$ will succeed in covering all nodes in G if any $\mathcal{W}(u_i, 1)$ covers all nodes.

So, we will bound the probability that all $\mathcal{W}(u_i, 1) \neq V$. Note that each $\mathcal{W}(u_i, 1)$ is independent of all other walks except for the node it starts on, but our upper bound is independent of the starting node. This means

$$\Pr\left[\mathcal{W}(u_i, 1) \neq V,\ \forall i \in [\tau]\right] = \prod_{i \in [\tau]} \Pr\left[\mathcal{W}(u_i, 1) \neq V\right] \leq \frac{1}{2^\tau}.$$

Therefore,

$$\Pr\left[\mathcal{W}(u, \tau) = V\right] = 1 - \Pr\left[\mathcal{W}(u, \tau) \neq V\right] \geq 1 - \Pr\left[\mathcal{W}(u, 1) \neq V\right]^\tau \geq 1 - \frac{1}{2^\tau}.$$

□

Theorem 6 (Completeness). *At the end of Protocol 1, which takes $2 \cdot \tau \cdot 8n^3$ rounds, every node will have $b = \bigvee_{i \in [n]} b_i$ with probability at least $1 - n/2^\tau$.*

Proof. First, we will prove that by the end of our protocol, every node along the walk OR's its bit and the resulting bit is decrypted. Then, we will prove that with all but probability $n/2^\tau$, every node has some walk that gets the output bit, meaning that with high probability, b at the end of the protocol is the output bit received by each node.

So, consider a single node, u_0, with bit b_0. Recall that $T = \tau \cdot 8n^3$. We will follow one walk that starts at u_0 with bit b_0. In the protocol, u_0's neighbors are ordered 1 to d_{u_0} and referred to by their number. Since this ordering is arbitrary, we will let u_1 identify the neighbor chosen by the protocol to send the encryption of b_0 in the first round, and, generalizing this notation, u_i will identify the ith node in the walk. For the sake of notation, pk_i will denote the public key generated by node u_i at step $i + 1$ for node u_{i+1} (so $pk_i = pk^{(i+1)}_{u_i \to u_{i+1}}$), and k_i will be the aggregate key-product at step i (so $k_i = pk_0 \circledast \ldots \circledast pk_i$).

- On the first step, u_0 encrypts b_0 with pk_0 into c_1 and sends it and public key pk_0 to one of its neighbors, u_1. We will follow c_1 on its random walk through T nodes.
- At step $i \in [T-1]$, c_i was just sent to u_i from u_{i-1} and is encrypted under the product $k_{i-1} = pk_0 \circledast pk_1 \circledast \cdots \circledast pk_{i-1}$, also sent to u_i. u_i computes the new public key $pk_0 \circledast \cdots \circledast pk_i = k_i$, adding its own public key to the product, encrypts b_i under k_i, and re-encrypts c_i under k_i via $\mathsf{AddLayer}$. Then, using the homomorphic OR, u_i computes c_{i+1} encrypted under k_i. u_i sends c_{i+1} and k_i to $u_{i+1} = \pi_i^{(u_i)}(u_{i-1})$.
- At step T, node u_T receives c_T, which is the encryption of $b_0 \vee b_1 \vee \cdots b_{T-1}$ under key $pk_0 \circledast \cdots \circledast pk_{T-1} = k_{T-1}$. u_T encrypts and then OR's his own bit to get ciphertext $e_T = \mathsf{HomOR}(c_T, [b_T]_{k_{T-1}})$. u_T sends e_T back to u_{T-1}.
- Now, on its way back in the decrypt phase, for each step $i \in [T-1]$, u_i has just received e_i from node u_{i+1} encrypted under $pk_1 \circledast \cdots \circledast pk_i = k_i$. u_i deletes the key layer pk_i to get k_{i-1} and then using $\mathsf{DelLayer}$, removes that key from encrypting e_i to get e_{i-1}. u_i sends e_{i-1} and k_{i-1} to $u_{i-1} = (\pi_i^{(u_i)})^{-1}(u_{i+1})$.
- Finally, node u_0 receives e_0 encrypted only under public key pk_0 on step 1. u_0 deletes that layer pk_0, revealing $e_0 = b_0 \vee \cdots \vee b_T$.

Now notice that each of these "messages" sent from every node to every neighbor takes a random walk on the graph when viewed on their own (these are correlated random walks when viewed as a whole, but independently, they can be analyzed as random walks). Let $\mathcal{W}_u$ represent some walk of the message starting at node u—even though u starts $\deg(u)$ different walks, we will only consider one walk per node.

By Corollary 5, for each $\mathcal{W}_u$, their set of traversed nodes covers the graph with probability $1 - \frac{1}{2^\tau}$ where $\tau = T/(8n^3)$. A union bound yields

$$\Pr[\exists u \in V,\ \mathcal{W}_u \neq V] \leq \sum_{u \in V} \Pr[\mathcal{W}_u \neq V] \leq n \cdot \frac{1}{2^n} \leq \frac{n}{2^\tau}.$$

This means that all walks cover the graph with at least the following probability

$$\Pr[\forall u \in V,\ \mathcal{W}_u = V] \geq 1 - \frac{n}{2^\tau},$$

and every walk will traverse the entire graph with all but negligible probability in our parameter τ. □

3.2 Proof of Soundness

We now turn to analyzing the security of our protocol, with respect to the topology-hiding security from Definition 2.

Theorem 7. *If the underlying encryption OR-homomorhpic PKCR scheme is CPA-secure, then Protocol 1 realizes the functionality of $\mathcal{F}_{Broadcast}$ in a topology-hiding way against a statically corrupting, semi-honest adversary.*

Proof. First, we will describe an ideal-world simulator $\mathcal{S}$: $\mathcal{S}$ lives in a world where all honest parties are dummy parties and has no information on the topology of the graph other than what a potential adversary knows. More formally, $\mathcal{S}$ works as follows

1. Let Q be the set of parties corrupted by $\mathcal{A}$. $\mathcal{A}$ is a static adversary, so Q and the inputs of parties in Q must be fixed by the start of the protocol.
2. $\mathcal{S}$ sends the input for all parties in Q to the broadcast function $\mathcal{F}_{broadcast}$. $\mathcal{F}_{broadcast}$ outputs bit b_{out} and sends it to $\mathcal{S}$. Note $\mathcal{S}$ only requires knowledge of Q's inputs and the output of $\mathcal{F}_{\text{Broadcast}}$.
3. $\mathcal{S}$ gets the local neighborhood for each $P \in Q$: $\mathcal{S}$ knows how many neighbors each P has and if that neighbor is also in Q, but doesn't need to know anything else about the topology[4].
4. Consider every party $P \in Q$ such $\mathcal{N}(P) \not\subset Q$. $\mathcal{S}$ will need to simulate these neighbors not in Q.
   - **Simulating messages from honest parties in Aggregate phase.** For every $Q \in \mathcal{N}(P)$ and $Q \notin Q$, $\mathcal{S}$ simulates Q as follows. At the start of the algorithm, $\mathcal{S}$ creates T key pairs:

     $$(pk^{(1)}_{Q\to P}, sk^{(1)}_{Q\to P}), \cdots, (pk^{(T)}_{Q\to P}, sk^{(T)}_{Q\to P}) \leftarrow \mathsf{Gen}(1^\kappa)$$

     At step $t = i$ in the for loop on line 8, $\mathcal{S}$ simulates Q sending a message to P by sending $([0]_{pk^{(i)}_{Q\to P}}, pk^{(i)}_{Q\to P})$. $\mathcal{S}$ receives the pair $(c^{(i)}_{P\to Q}, k^{(i)}_{P\to Q})$ from P at this step.
   - **Simulating messages from honest parties in the Decrypt phase.** Again, for every $P \in Q$, $Q \in \mathcal{N}(P)$ and $Q \notin Q$, $\mathcal{S}$ simulates Q. At $t = i$ in the for loop on line 21, $\mathcal{S}$ sends $[b_{out}]_{k^{(i)}_{Q\to P}}$ to P. $\mathcal{S}$ receives $e^{(i)}_{P\to Q}$ from P.

We will prove that any PPT adversary cannot distinguish whether he is interacting with the simulator $\mathcal{S}$ or with the real network except with negligible probability.

1. Hybrid 1. $\mathcal{S}$ simulates the real world exactly and has information on the entire topology of the graph, each party's input, and can simulate each random walk identically to how the walk would take place in the real world (Fig. 4, top).
2. Hybrid 2. $\mathcal{S}$ replaces the real keys with simulated public keys, but still knows everything about the graph (as in Hybrid 1). Formally, for every honest Q that is a neighbor to $P \in Q$, $\mathcal{S}$ generates

   $$(pk^{(1)}_{Q\to P}, sk^{(1)}_{Q\to P}), \cdots, (pk^{(T)}_{Q\to P}, sk^{(T)}_{Q\to P}) \leftarrow \mathsf{Gen}(1^\kappa)$$

   and instead of adding a layer to the encrypted $[b]_{pk^*}$ that P has at step t, as done in line 12 and 13, $\mathcal{S}$ computes $b' \leftarrow b_Q \vee b$ and sends $[b']_{pk^{(t)}_{Q\to P}}$ to P during the aggregate phase; it is the same message encrypted in Hybrid 1, but it is now encrypted under an unlayered, fresh public key. In the decrypt phase, each honest Q neighbor to P will get back the bit we get from the sequence of OR's encrypted under that new public key as well; the way all nodes in Q peel off layers of keys guarantees this.
3. Hybrid 3. $\mathcal{S}$ now simulates the ideal functionality during the aggregate phase, sending encryptions of 0. Formally, during the aggregate phase, every honest Q that is

[4] Recall that from Definition 2, $\mathcal{F}_{\text{graphInfo}}$ reveals if nodes in Q have neighbors in common, however all $\mathcal{S}$ needs to know is which neighbors are also in Q; $\mathcal{S}$ does not use all of the available graph information (in the full version of the paper, we describe a stronger definition capturing this quality).

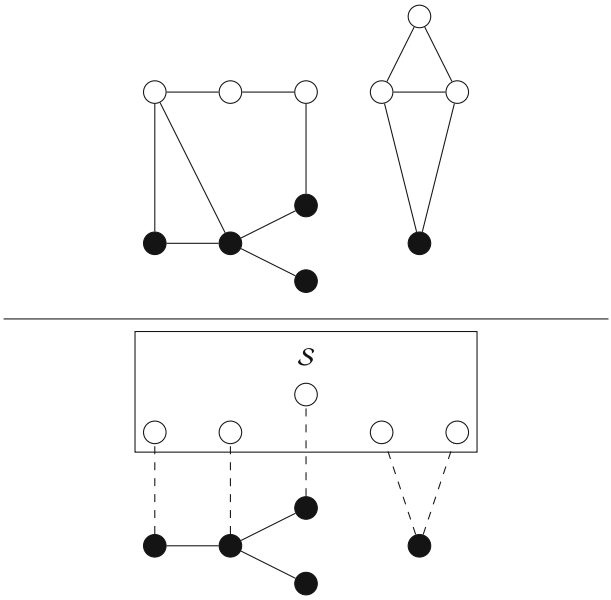

Fig. 4. An example of how the simulator works. The top shows how the graph normally looks. The bottom shows the graph topology that the simulator generates. Black nodes are nodes in Q. Notice that the simulator doesn't require knowing if nodes in Q have a common neighbor—neighbors can identify themselves with pseudonyms or edge-labels.

a neighbor to $P \in Q$ $\mathcal{S}$ sends $[0]_{pk^{(t)}_{Q\to P}}$ to P instead of sending $[b']_{pk^{(t)}_{Q\to P}}$. Nothing changes during the decrypt phase; the simulator still sends the resulting bit from each walk back and is not yet simulating the ideal functionality.

4. Hybrid 4. $\mathcal{S}$ finally simulates the ideal functionality at the during the decrypt phase, sending encryptions of b_{out}, the output of $\mathcal{F}_{\text{Broadcast}}$, under the simulated public keys. This is instead of simulating random walks through the graph and ORing only specific bits together. Notice that this hybrid is equivalent to our original description of $\mathcal{S}$ and requires no knowledge of other parties' values or of the graph topology other than local information about Q (as specified by the $\mathcal{F}_{\text{graphInfo}}$ functionality).

Now, let's say we have an adversary $\mathcal{A}$ that can distinguish between the real world and the simulator. This means $\mathcal{A}$ can distinguish between Hybrids 1 and 4. So, $\mathcal{A}$ can distinguish, with non-negligible probability, between two consecutive hybrids. We will argue that given the security of our public key scheme and the high probability of success of the algorithm, that this should be impossible.

1. First, we claim no adversary can distinguish between Hybrid 1 and 2. The difference between these Hybrids is distinguishing between AddLayer and computing a fresh encryption key. In Hybrid 1, we compute a public key sequence, multiplying public key k by a freshly generated public key. In Hybrid 2, we just use a fresh public key. Recall that the public keys in our scheme form a group. Since the key sequence

$k \circledast pk_{new}$ has a new public key that has not been included anywhere else in the transcript, $k \circledast pk_{new}$ can be thought of as choosing a new public key independently at random from k. This is the same distribution as just choosing a new public key: $\{k \circledast pk_{new}\} \equiv \{pk_{new}\}$. Therefore, any tuple of multiplied keys and fresh keys are indistinguishable from each other. So, no adversary $\mathcal{A}$ can distinguish between Hybrids 1 and 2.

2. Now we will show that no PPT adversary can distinguish between Hybrids 2 and 3. The only difference between these two hybrids is that $\mathcal{A}$ sees encryptions of the broadcast bit as it is being transmitted as opposed to seeing only encryptions of 0 from the simulator. Note that the simulator chooses a key independent of any key chosen by parties in Q in each of the aggregate rounds, and so the bit is encrypted under a key that $\mathcal{A}$ does not know. This means that if $\mathcal{A}$ can distinguish between these two hybrids, then $\mathcal{A}$ can break semantic security of the scheme, distinguishing between encryptions of 0 and 1.
3. For this last case, we will show that there should not exist a PPT adversary $\mathcal{A}$ that can distinguish between Hybrids 3 and 4.
   There are two differences between Hybrids 3 and 4. The first is that, during the decrypt phase, we send $b_{out} = \bigvee_{i \in [n]} b_i$, the OR of all of the node's bits, instead of $b_{\curvearrowright W} = \bigvee_{u \in \curvearrowright W} b_u$, the OR of all node's bits in a specific length-T walk.
   Corollary 5 tells us that a walk $\mathcal{W}$ taken during the course of the algorithm covers the graph with probability $1 - 1/2^{\tau}$. There are two walks starting at each edge in the graph, which is at most $2n^2$ walks. So, the probability that $b_{out} \neq b_{\curvearrowright W}$ at most $2n^2/2^{\tau}$, which is negligible in τ, and therefore is undetectable.
   The second difference is that our simulated encryption of b_{out} is generated by making a fresh encryption of b_{out}. But, if $b_{out} = b_{\curvearrowright W}$ (which it will with overwhelming probability), by the claim in Sect. 2.5, the encryption generated by ORing many times in the graph is computationally indistinguishable to a fresh encryption of b_{out}. Therefore, computationally, it is impossible to distinguish between Hybrids 3 and 4. □

3.3 Proof of Main Theorem

In this section, we put the pieces together: we formally state and prove Theorem 1 using Protocol 1.

Theorem 8 (Topology-hiding broadcast for all network topologies). *If there exists an OR-homomorphic PKCR, then for any network topology graph G on n nodes, there exists a polynomial-time protocol Π that is a topology-hiding realization of broadcast functionality $\mathcal{F}_{broadcast}$.*

Proof. Will will show that Protocol 1 is the topology-hiding realization of $\mathcal{F}_{broadcast}$. Since we assume existence of an OR-homomorphic PKCR, we are able to run our protocol. The rest of this proof is simply combining the results of Theorems 6 and 7. Now, for a security parameter κ, we let $\tau = \kappa + \log(n)$.

To show Protocol 1 is complete, Theorem 6 states that for our parameter τ, Protocol 1 outputs the correct bit for every node with probability at least $1 - n/2^{\tau} = 1 - 1/2^{\kappa}$.

This means, our protocol is correct with overwhelming probability with respect to the security parameter κ.

To show our protocol is sound, Theorem 7 states that for our input parameter τ, an adversary can distinguish a simulated transcript from a real transcript with probability negligible in τ. Since τ is strictly greater than κ, our protocol is secure with respect to κ as well. Therefore, Protocol 1 is sound against all PPT adversaries: they have only a negligible chance with respect to κ of distinguishing the simulation versus a real instantiation of the protocol. □

Corollary 9. *Under the DDH assumption, there exists polynomial-time executable, topology-hiding broadcast for any graph G.*

Proof. ElGamal, which is secure under the DDH assumption, is an OR-homomorphic PKCR by Sect. 2.5. So, applying Theorem 8, we get that there exists a protocol which is a topology-hiding realization of $\mathcal{F}_{broadcast}$. □

Because we now have topology-hiding broadcast on any graph, we can use the existence of secure MPC for all efficiently computable functionalities $\mathcal{F}$, we get topology-hiding MPC for all efficiently computable $\mathcal{F}$ (assuming we have an OR-homomorphic PKCR, or DDH).

4 Complexity and Optimizations

In this section we give an upper bound on the communication complexity of Protocol 1 and discuss optimizations for graph families where tighter cover time bounds are known.

In the following n, m are upper bounds on the number of nodes and edges; B an upper bound on the cover time; and τ an input parameter controlling the probability of incorrect output to be at most $n/2^{\tau}$. We point out that while in Protocol 1 we set the number of rounds to be $T = 2\tau B$ for $B = 4n^3$; our completeness and soundness proofs hold for every upper bound B on the cover time.

4.1 Communication Complexity

We show that the communication complexity is $\Theta(B\tau m)$ group elements, where B is an upper bound on the cover time of the graph (for our protocol on general graphs, we have $B = 4n^3$). We measure the communication complexity in terms of the overall number of group elements transmitted throughout the protocol (where the group elements are for the ciphertext and public-key pairs of the underlying DDH-based encryption scheme, and their size is polynomial in the security parameter).

Claim (Communication complexity). The communication complexity of Protocol 1 with $T = 2\tau B$ is $\Theta(B\tau m)$ group elements.

Proof. The random-walks in Protocol 1 are of length $T = 2B\tau$, yielding $2T$ total rounds of communication including both the forward and backwards phases. At each round, every node v sends out $\deg(v)$ messages. Summing over all $v \in V$, all of the nodes

communicate $2m$ messages every round – one for each direction of each edge (for m denoting the number of edges in the network graph). By the end of the protocol, the total communication is $4Tm = \Theta(B\tau m)$. □

We conclude the communication complexity of Protocol 1 on input n, τ is $\Theta(\tau n^5)$ group elements.

Corollary 10. *On input n, τ, the communication complexity of Protocol 1 is $\Theta(\tau n^5)$ group elements.*

Proof. For a graph with at most n nodes, $B = 4n^3$ is an upper bound on the cover time (see Theorem 4), and $m = n^2$ is an upper bound on the number of edges. Assigning those B, m in the bound from Sect. 4.1, the proof follows: $\Theta(B\tau m) = \Theta(\tau \cdot n^3 \cdot n^2) = \Theta(\tau n^5)$. □

4.2 Better Bounds on Cover Time for Some Graphs

Now that we have seen how the cover time bound B controls both the communication and the round complexity, we will look at how to get a better bound than $O(n^3)$.

Cover time has been studied for various kinds of graphs, and so if we leak the kind of graph we are in (e.g. expanders), then we can use a better upper bound on the cover time, shown in Fig. 5.

For example on expander graphs (arising for example in natural applications on random regular graphs), it is known that the cover times $C_G = O(n \log n)$, much less than $O(n^3)$ [6]. This means that for expanders, we can run in $C_G = O(n \log n)$ round complexity, and $O(C_G \tau m) = O(\tau mn \log n)$ communication complexity. Even assigning the worst case bound $m \leq n^2$, we get round and communication complexity $O(n \log n)$ and $O(\tau n^3 \log n)$ respectively—much better than the general case that has $O(\tau n^5)$ communication complexity.

Type of Graph	Cover time
Arbitrary graph [21]	$O(n^3)$
Expanders [6]	$O(n \log n)$
Regular Graphs [18]	$O(n^2)$

Fig. 5. Cover times for specific graphs.

5 Conclusion and Future Work

This work showed that topology-hiding computation is feasible for *every* network topology (in the computational setting, assuming DDH), using random walks. This resolution completes a line of works on the feasibility of topology hiding computation against a static semi-honest adversary [1,16,21]. Yet, it leaves open the feasibility question against a malicious or adaptive adversary. Another intriguing question is whether our

random walks could be derandomized, perhaps using *universal-traversal* [2,19] that is a deterministic walk guaranteed to cover all d-regular n-nodes graph, with explicit constructions known under some restrictions such as consistent labeling [17].

References

1. Akavia, A., Moran, T.: Topology hiding computation beyond logarithmic diametere. In: Eurocrypt (2017, to appear)
2. Aleliunas, R., Karp, R.M., Lipton, R.J., Lovász, L., Rackoff, C.: Random walks, universal traversal sequences, and the complexity of maze problems. In: 20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29–31 October 1979, pp. 218–223 (1979)
3. Balogh, J., Bollobás, B., Krivelevich, M., Müller, T., Walters, M.: Hamilton cycles in random geometric graphs. Ann. Appl. Probab. **21**(3), 1053–1072 (2011)
4. Beimel, A., Gabizon, A., Ishai, Y., Kushilevitz, E., Meldgaard, S., Paskin-Cherniavsky, A.: Non-interactive secure multiparty computation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 387–404. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44381-1_22
5. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: FOCS, pp. 136–145. IEEE Computer Society (2001)
6. Chandra, A.K., Raghavan, P., Ruzzo, W.L., Smolensky, R.: The electrical resistance of a graph captures its commute and cover times. In: Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing, STOC 1989, pp. 574–586. ACM, New York (1989)
7. Chandran, N., Chongchitmate, W., Garay, J.A., Goldwasser, S., Ostrovsky, R., Zikas, V.: 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 2015, pp. 153–162. ACM, New York (2015)
8. Estrin, D., Govindan, R., Heidemann, J., Kumar, S.: Next century challenges: scalable coordination in sensor networks. In: Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 263–270. ACM (1999)
9. Friedrich, T., Sauerwald, T., Stauffer, A.: Diameter and broadcast time of random geometric graphs in arbitrary dimensions. Algorithmica **67**(1), 65–88 (2013)
10. Goldreich, O.: Foundations of Cryptography: Basic Applications, vol. 2. Cambridge University Press, New York (2004)
11. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, STOC 1987, pp. 218–229. ACM, New York (1987)
12. Goldwasser, S., et al.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (2014). doi:10.1007/978-3-642-55220-5_32
13. Halevi, S., Ishai, Y., Jain, A., Kushilevitz, E., Rabin, T.: Secure multiparty computation with general interaction patterns. In: Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, ITCS 2016, pp. 157–168. ACM, New York (2016)
14. Halevi, S., Lindell, Y., Pinkas, B.: Secure computation on the web: computing without simultaneous interaction. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 132–150. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22792-9_8
15. Hinkelmann, M., Jakoby, A.: Communications in unknown networks: preserving the secret of topology. Theor. Comput. Sci. **384**(2–3), 184–200 (2007). Structural Information and Communication Complexity (SIROCCO 2005)

16. Hirt, M., Maurer, U., Tschudi, D., Zikas, V.: Network-hiding communication and applications to multi-party protocols. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 335–365. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53008-5_12
17. Hoory, S., Wigderson, A.: Universal traversal sequences for expander graphs. Inf. Process. Lett. **46**(2), 67–69 (1993)
18. Kahn, J.D., Linial, N., Nisan, N., Saks, M.E.: On the cover time of random walks on graphs. J. Theor. Probab. **2**(1), 121–128 (1989)
19. Lovász, L.: Random walks on graphs: a survey. In: Miklós, D., Sós, V.T., Szőnyi, T. (eds.) Combinatorics, Paul Erdős is Eighty, vol. 2, pp. 353–398. János Bolyai Mathematical Society, Budapest (1996)
20. Mitzenmacher, M., Upfal, E.: Probability and Computing - Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, Cambridge (2005)
21. Moran, T., Orlov, I., Richelson, S.: Topology-hiding computation. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9014, pp. 169–198. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46494-6_8
22. Penrose, M.: Random Geometric Graphs. Oxford University Press, Oxford (2003). no. 5
23. Pottie, G.J., Kaiser, W.J.: Wireless integrated network sensors. Commun. ACM **43**(5), 51–58 (2000)
24. Yao, A.C.-C.: How to generate and exchange secrets. In: Proceedings of the 27th Annual Symposium on Foundations of Computer Science, SFCS 1986, pp. 162–167. IEEE Computer Society, Washington, D.C. (1986)