Network-Hiding Communication and Applications to Multi-Party Protocols

Martin Hirt[1], Ueli Maurer[1], Daniel Tschudi[1*], and Vassilis Zikas[2**]

[1] ETH Zurich
{hirt, maurer, tschudid}@inf.ethz.ch
[2] RPI
vzikas@cs.rpi.edu

Abstract. As distributed networks are heavily used in modern applications, new security challenges emerge. In a multi-party computation (in short, MPC) protocol over an incomplete network, such a challenge is to hide, to the extent possible, the topology of the underlying communication network. Such a topology-hiding (aka network hiding) property is in fact very relevant in applications where anonymity is needed.

To our knowledge, with the exception of two recent works by Chandran *et al.* [ITCS 2015] and by Moran *et al.* [TCC 2015], existing MPC protocols do not hide the topology of the underlying communication network. Moreover, the above two solutions are either not applicable to arbitrary networks (as is [ITCS 2015]) or, as in [TCC 2015], they make non-black-box and recursive use of cryptographic primitives resulting in an unrealistic communication and computation complexity even for simple, i.e., low degree and diameter, networks.

Our work suggests the first topology-hiding communication protocol for incomplete networks which makes black-box use of the underlying cryptographic assumption—in particular, a public-key encryption scheme—and tolerates any adversary who passively corrupts arbitrarily many network nodes. Our solutions are based on a new, enhanced variant of threshold homomorphic encryption, in short, TH-PKE, that requires no a-priori setup and allows to circulate an encrypted message over any (unknown) incomplete network and then decrypt it without revealing any network information to intermediate nodes. We show how to realize this enhanced TH-PKE from the DDH assumption. The black-box nature of our scheme, along with some optimization tricks that we employ, makes our communication protocol more efficient than existing solutions.

We then use our communication protocol to make any semi-honest secure MPC protocol topology-hiding with a reasonable—i.e., for simple networks, polynomial with small constants—communication and computation overhead. We further show how to construct anonymous broadcast without using expensive MPCs to setup the original pseudonyms.

1 Introduction

Secure communication is perhaps the central goal of cryptography. It allows a sender, Alice, to securely transmit a message to a receiver, Bob so that even if some eavesdropper, Eve, is intercepting their communication she can not figure out anything about the transmitted message. When Alice and Bob share a physical (but potentially tappable) communication channel, this task can be easily carried out by use of standard public-key cryptography techniques, e.g., Bob sends Alice his public key who uses it to encrypt her message and send it over the physical communication channel to Bob. But this idealized scenario occurs rarely in modern networks, such as the Internet, where Alice and Bob would most likely not share a physical channel and would, instead, have to communicate over some (potentially incomplete) network of routers. Without further restrictions, the above modification marginally complicates the problem as it can be directly solved by means of a private flooding scheme. In such a scheme, Alice encrypts her message, as before, and sends it to all her immediate neighbors, i.e., network routers with which she shares physical links, who then forward it to their immediate neighbors, and so on, until it reaches Bob. Clearly, if Alice has a path to Bob and the forwarding step is repeated as many times as the length of this path, the message will reach Bob. And the fact that the intermediate routers only see encryptions of the transmitted message means that they do not learn anything about the message.

* Research was supported by the Swiss National Science Foundation (SNF), project no. 200020-132794.

** Work done in part while the author was at ETH Zurich supported by the Swiss NSF Ambizione grant PZ00P2_142549, and while the author was visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467.

But modern distributed protocols often require much more than just privacy of the transmitted message. For example, ensuring anonymity in communication is a major goal of security as it, for example, protects against censorship or coercion. Similarly, as privacy awareness in social networks increases, users might not be willing to reveal information about the structure of their peer graph (i.e., their Facebook friends graph) to outsiders. Other applications might require to hide a communicating agent's location, as is the case in espionage or when using mobile agents to propagate information through some ad-hoc network, e.g., in vehicle-to-vehicle communication. All these applications require a routing scheme, that hides the topology of the underlying communication network. Evidently, using the simple private flooding strategy does not hide the topology of the underlying communication network as, for example, an eavesdropping router can easily determine its distance (and direction) to the sender by observing in which round (and from whom) it receives the first encryption.

1.1 Related Literature

The problem of routing through an incomplete network has received a lot of attention in communication networks with a vast amount of works aiming at optimizing communication complexity in various network types. In the following, however, we focus on the cryptographic literature which is more relevant to our goals—namely network hiding communication—and treatment.

Perhaps the main venue of work in which keeping the network hidden is a concern is the literature on anonymous communication, e.g., [Cha03, RR98, SGR97]. These works aim to hide the identity of the sender and receiver in a message transmission, in a way that protects these identities even against traffic analysis. In a different line of work initiated by Chaum [Cha81], so called *mix* servers are used as proxies which shuffle messages sent between various peers to disable an eavesdropper from following a message's path. This technique has been extensively studied and is the basis of several practical anonymization tools. An instance of the mix technique is the so called onion routing [SGR97, RR98], which is perhaps the most wide-spread anonymization technique. Roughly, it consists of the sender applying multiple encryptions in layers on his message, which are then "peeled-off" as the cipher-text travels through a network of onion routers towards its destination. An alternative anonymity technique by Chaum [Cha88] and implemented in various instances (e.g.,[Bd90, GJ04, GGOR14]) is known as *Dining Cryptographers networks*, in short DC-nets. Here the parties themselves are responsible for ensuring anonymity. The question of hiding the communication network was also recently addressed in the context of secure multi-party computation by Chandran *et al.* [CCG$^+$15]. This work aims to allow n parties to compute an arbitrary given function in the presence of an adaptive adversary, where each party communicates with a small (sublinear in the total number of parties) number of its neighbors. Towards this goal, [CCG$^+$15] assumes that parties are secretly given a set of neighbors that they can communicate with. Because the adversary is adaptive, it is crucial in their protocol that the communication does not reveal much information about the network topology, as such information would allow the adversary to potentially discover the neighbors of some honest party, corrupt them, and isolate this party, thereby breaking its security.[3] Another work which considers such an adaptive corruption setting is the work of King and Saia [KS10], which is tailored to the Byzantine agreement problem. We note in passing that the result of [CCG$^+$15, KS10] was preceded by several works which considered the problem of MPC over incomplete networks. However, these works do not aim to keep the network hidden as they either only consider a static adversary,[4] e.g., [BGT13], and/or they only achieve so called *almost everywhere computation* [GO08, KSSV06a, KSSV06b, CGO15] where the adversary is allowed to isolate a small number of honest parties.

Most related to the goals of our work is the recent work of Moran, Orlov, and Richelson [MOR15], which considers the problem of *topology-hiding secure multi-party computation* over an incomplete network in the computational setting (i.e., assuming secure public-key encryption) tolerating a semi-honest (passive) and static adversary. At a very high level, [MOR15] uses public-key encryption and (semi-honest) multi-party computation to implement a proof-of-concept network-hiding communication protocol, which emulates a complete network of secure channels. This emulated network is then used to execute an arbitrary multi-party protocol in which parties communicate over a complete communication network, e.g., [GMW87, Pas04]. In fact, as noted in [MOR15], relying on a computational assumption seems inevitable, as in the information-theoretic setting the work of Hinkelmann and Jakoby [HJ07]

[3] In fact, by a factor $\sqrt{n}$ increase on the number of neighbors of each party, [CCG$^+$15] can avoid the assumption of a trusted setup privately distributing the neighborhoods and achieve the same level of security while having the parties generate these neighborhoods themselves.

[4] A static adversary chooses all the parties to corrupt at the beginning of the protocol execution and therefore learning the network topology through the communication cannot help him isolate any honest party.

excludes fully topology-hiding communication.[5] Due to the similarity to our goal we include a detailed comparison of our results with [MOR15] in Section 1.3.

1.2 Our Contributions

In this work we present the first network-hiding communication protocol which makes black-box use of public-key encryption and, for networks with moderate degree and diameter, has a moderate communication and computation complexity. Our protocol allows the parties to communicate over an incomplete network of point-to-point channels in a way which computationally hides both the transmitted message and the neighborhood of honest parties from an adversary *passively* corrupting arbitrary many parties. We remark that as pointed out in [CCG+15], when the communication graph is to be kept hidden, the adversary cannot be eavesdropping on communication channels, and in particular cannot be informed when a message is transmitted over some channel. We resolve this issue by assuming, along the lines of [MOR15], a special network functionality (cf. Section 2).

A bit more concretely, the high-level idea of our construction is to enhance the naïve private flooding-protocol by using homomorphic public-key encryption (in short, PKE). The starting point of our approach is the observation—underlying also the construction from [MOR15]—that the flooding protocol would be topology-hiding if the parties could not read intermediate messages. But instead of using, as in [MOR15], expensive nested MPCs for ensuring this fact (see below for a high-level description of [MOR15]) we use a version of threshold PKE with additional network hiding properties. We also show how to implement our enhanced threshold PKE definition assuming hardness of the Decisional Diffie-Hellmann (DDH) problem.

To demonstrate our ideas, imagine there was a world in which parties (corresponding to all intermediate routers) could encrypt with a homomorphic public-key encryption scheme where the private (decryption) key is known to nobody, but instead parties have access to a decryption oracle. Provided that the associated PKE-scheme is semantically secure, parties can enhance the flooding protocol as follows: Alice encrypts its message and starts the flooding; in each step of the flooding protocol, the intermediate party—which, recall, is supposed to forward the received ciphertext—first re-randomizes the ciphertext and then forwards it. Once the message arrives to Bob, he invokes the decryption oracle to open its final ciphertext. We observe that in this case the adversary does no longer learn anything from intermediate messages, the protocol is thus topology-hiding.

There are two major challenges with the above approach. First, if intermediate parties are silent until a message reaches them during the flooding, then the adversary observing this fact can use it to deduce information about the network. E.g., if a neighbor p_i of a corrupted party has not sent anything by the second round of the flooding protocol, then the adversary can deduce that p_i is not a neighbor of Alice. Secondly, we need a way to implement the decryption oracle. Observe that using a off-the-shelf threshold decryption scheme and have decryption shares exchanged by means of flooding would trivially destroy the topology-hiding property; and the same is the case if we would use an MPC protocol for this purpose, unless the MPC were itself topology-hiding. In the following we discuss how we solve each of the protocols, separately.

The first issue—information leakage from silent parties—can be solved by having every party send messages in every round. As simple as this idea might seem, it has several difficulties. For starters, the messages that are injected by intermediary parties should be indistinguishable from encryptions, as otherwise adding this noise makes no difference. But now, there is a new issue that the intermediate parties cannot tell which of the indistinguishable messages they receive contains the initial message sent by Alice. The naive solution to this would be to have parties re-randomize everything they receive and add their own noise-message. But this would impose an exponential, in the graph diameter, factor both in the message and communication complexity. Our solution, instead, is to use the homomorphic properties of the encryption scheme and build an efficient process which allows every party to compute an encryption of the OR of the messages it receives from its neighbors. Thus, to transfer a bit b, Alice encrypts b and starts flooding, whereas every party encrypts a zero-bit and starts flooding simultaneously. In each following round of the flooding scheme, every party homomorphically computes the OR of the messages it receives and continues flooding with only this encryption. Bob keeps computing the OR of the encryptions he receives, and once sufficiently many rounds have passed, the decryption is invoked to

[5] To our understanding the result of [HJ07] does not apply to the case where a strong information-theoretic setup, e.g., sufficiently long correlated randomness, is available to the parties. Extending this results to that setting is an interesting open problem.

have him obtain Alice's bit. Note that we only treat the case of semi-honest parties here, thus no party will input an encryption of a one-bit into this smart flooding scheme which would destroy its correctness.

To solve the second issue—i.e., implement the decryption oracle in a topology hiding manner—we introduce a new variant of threshold homomorphic public-key encryption (TH-PKE) with enhanced functionality, which we call *multi-homomorphic threshold encryption with reversible randomization.* Roughly speaking, our new TH-PKE assumes a strongly correlated setup, in which secret (sub)keys are nested in a way which is consistent with the network topology and which allows parties to decrypt messages in a topology hiding manner. We provide a security definition for the new primitive and describe a topology-hiding protocol for establishing the necessary setup using no setup-assumption whatsoever. And we also describe how to instantiate our schemes under the DDH assumptions. We believe that both the general definition of this augmented TH-PKE and the concrete instantiation could be of independent interest and can be used for anonymizing communication.

Applications Building on our topology hiding network and utilizing the functionality of our topology hiding homomorphic OR protocol we present the following applications:

– Anonymous broadcast: We consider a variant of anonymous broadcast where parties can broadcast messages under a pseudonym. The presented protocol allows to realize anonymous broadcast directly from the topology hiding homomorphic OR protocol without using expensive MPC to setup the pseudonyms.
– Topology hiding MPC: Having a topology-hiding network, we can execute on top of it any MPC protocol from the literature that is designed for point-to-point channels which will render it topology hiding.

1.3 Comparison with [MOR15]

The work by Moran *et al.* [MOR15] provides the first, to the best of our knowledge, work that solves this problem for general graphs in the computational setting. Our goals are closely related to theirs. In fact, our security definition of topology-hiding communication and, more general, computation is a refinement of their simulation-based definition of topology-hiding MPC. But our techniques are very different. In light of this similarity in goals, in the following we include a more detailed comparison to our work.

More concretely, the solution of [MOR15] also follows the approach of enhancing the naïve flooding protocol to make it topology hiding. The key idea is to use nested MPCs, recursively, to protect sensitive information during the execution of the flooding protocol. Roughly, in the basic topology-hiding communication protocol of [MOR15], each party P_i is replaced by a virtual-party $\hat{P}_i$, which is emulated by its immediate neighbors by invoking locally (i.e., in the neighborhood) an off-the-shelf MPC protocol. The complete network of point-to-point channels required by the MPC protocol is emulated by use of a PKE-scheme over the star network centered around P_i, i.e., by naïve flooding where P_i is used as the routing node. The above ensures that P_i cannot analyze the messages that are routed through him, as they are actually handled by its corresponding virtual party $\hat{P}_i$. However, there is now a new problem to be solved, namely, how do virtual parties use the underlying (incomplete) communication network to flood messages in a topology hiding manner? This is solved as follows: To enable secure communication between adjacent virtual-parties a PKE-scheme is used (once more). Here each virtual-party generates a key-pair and sends the encryption key to the adjacent virtual-parties using real parties as intermediates. This basic protocol is topology-hidingly secure as long as the adversary does not corrupt an entire neighborhood. But this is of course not enough for arbitrarily many corruptions to be tolerated. Thus, to ensure that the overall flooding protocol is also topology hiding, each virtual party is replaced, again by means of MPC, by a "doubly virtual" party $\hat{\hat{P}}$. This will ensure that only adversaries corrupting all the parties that emulate $\hat{\hat{P}}$ can break the topology hiding property. To extend the set of tolerable adversaries, the doubly virtual parties are again emulated, and this process is continued until we reach an emulated party that is emulated by all parties in the network. This requires in the worst case a number of nested MPCs in the order of the network diameter.

In the following we provide a comparison of the solution of [MOR15] with ours demonstrating the advantages of our solution both in terms of simplicity and efficiency. In all fairness, we should remark that the solution of [MOR15] was explicitly proposed as a proof-of-concept solution. The major advantage of our work over [MOR15] is that our communication protocol makes no use of generic MPC, and makes black-box use of the underlying PKE. This not only yields a substantial efficiency improvement, in terms of both communication and computation, but it also yields a more intuitive solution to the problem, as it uses the natural primitive to make communication private, namely encryption, instead of MPC.

More concretely, the player-virtualization protocol from [MOR15] makes non-black-box use of public-key encryption, i.e., the circuit which is computed via MPC is a public-key encryption/decryption circuit. This is typically a huge circuit which imposes an unrealistic slowdown both on the computation complexity and on the round and/or communication complexity.[6] And this is just at the first level of recursion; the computation of the second level, computes a circuit, which computes the circuit, which computes PK encryptions/decryptions, and so on. Due to the lack of concrete suggestions of instantiation of the PKE and MPC used in [MOR15] we were unable to compute exact estimates on the running time and communication complexity of the suggested protocols. Notwithstanding it should be clear that even for the simple case in which the network has constant degree and logarithmic diameter—for which their communication protocol in [MOR15] achieves a polynomial complexity—and even for the best MPC instantiation the actual constants are huge.

Instead, our solutions make black-box use of the underlying PKE scheme and are, therefore, not only more communication and computation efficient, but also easier to analyze. In fact, in our results we include concrete upper bounds on the communication complexity[7] of all our protocols. Indicatively, for a network with diameter D and maximum degree d our network-hiding broadcast protocol communicates at most $(d+1)^D \cdot n \cdot \lambda$ bits within just $5 \cdot D$ rounds, where λ is linear (with small constant, less than 5)[8] in the security parameter κ of the underlying PKE scheme. We note that many natural network graphs, such as social networks or the internet have a small diameter.[9]

1.4 Preliminaries and Notation

We consider an MPC-like setting where n parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ wish to communicate in a synchronous manner over some incomplete network of secure channels. When the communication is intended to be from P_i, the *sender*, to P_j, the *receiver*, we will refer to the parties in $\mathcal{P} \setminus \{P_i, P_j\}$ as the *intermediate parties.* We will assume a passive and non-adaptive (aka static) computationally bounded adversary who corrupts an arbitrary subset $\overline{H} \subseteq \mathcal{P}$ of parties. Parties in $\overline{H}$ are called *dishonest* or *corrupted* while parties in $H = \mathcal{P} \setminus \overline{H}$ are called *honest.* We use simulation based security to prove our results. For simplicity our proofs are in Canetti's modular composition framework [Can98] but all our results translate immediately to the universal composition UC framework [Can00]. (Recall that we consider semi-honest static security.) In fact, to make this transition smoother, we describe our hybrids in the form of UC functionalities. For compactness, for any functionalities $\mathcal{F}$ and $\mathcal{G}$, we will denote by $\{\mathcal{F}, \mathcal{G}\}$ the composite functionality that gives parallel access to $\mathcal{F}$ and $\mathcal{G}$.

Throughout this work, we assume an, at times implicit, security parameter κ and write $\text{neg}(\kappa)$ to refer to a negligible function of κ. (See [Gol01] for a formal definition of negligible functions.) For an algorithm A we write $(y_1, \ldots, y_k) \leftarrow A(x_1, \ldots, x_k)$ to denote that $(y_1, \ldots, y_k)$ are outputs of A given inputs $(x_1, \ldots, x_k)$. For a probabilistic algorithm B we write $(y_1, \ldots, y_k) \leftarrow B(x_1, \ldots, x_k; r)$ where r is the chosen randomness. If we write $(y_1, \ldots, y_k) \twoheadleftarrow B(x_1, \ldots, x_k)$ instead, we assume that the randomness has been chosen uniformly.

1.5 Organization of the Paper

The remainder of the paper is organized as follows. In Section 2 we give our definition of topology-hiding security. In Section 3 we present a construction which allows to realize topology-hiding communication. The construction is based on multi-homomorphic threshold encryption with reversible randomization (RR-MHT-PKE) which is introduced in Section 3.1. Next, in Section 3.2 we describe a topology-hiding threshold encryption protocol based on RR-MHT-PKE. This protocol is used in Section 3.3 to topology-hidingly realize the Boolean-OR functionality. This allows to give a toplogy-hiding construction of broadcast and secure channels in Section 3.4. Finally, in Section 4 we present topology-hiding MPC and topology-hiding anonymous broadcast as applications of the protocols from the previous section.

[6] Of course the latter can be traded off by choosing to use either a communication heavy or a round heavy protocol.

[7] We note that the computation complexity of our protocols is similar to their communication complexity.

[8] This can be contrasted with the complexity $O(d)^D \cdot n \cdot \lambda$ obtained by [MOR15].

[9] Backstrom *et al.* [UKBM11] showed that a sub-graph of the Facebook social network consisting of 99.6% of all users had a diameter of 6. In this particular case the broadcast protocol would communicate at most $n^7 \cdot \lambda$ bits within 30 rounds.

2 Topology Hiding Security Definition

In this section we provide the formal simulation-based definition of topology-hiding computation. Our definition is an adaptation of the original simulation-based definition of Moran et al. [MOR15]. More concretely, the topology-hiding property requires that parties learn no information on the underlying communication network other than the description of their local neighborhood, i.e., the identities of their neighbors. To capture this property, we assume that the parties (in the real world) have access to a *network* functionality $\mathcal{N}$ which has knowledge of every party P_i's neighborhood (i.e., the set of point-to-point channels connected to P_i) and allows P_i to communicate (only) to its neighbors.

Clearly, a protocol execution over such a network $\mathcal{N}$ allows an adversary using it knowledge of the neighborhood of corrupted parties; thus the simulator needs to also be able to provide this information to its environment. To give this power to the simulator, [MOR15] augments the ideal functionality with an extra component which allows the simulator access to this information. In this work we use $\mathcal{N}$ itself in the ideal world to provide this information to the simulator. Note that this does not affect the security statements, as the trivial $\mathcal{N}$-dummy protocol $\phi^{\mathcal{N}}$ securely realizes $\mathcal{N}$. [10]

A conceptual point in which our model of topology-hiding computation deviates from the formulation of Moran *et al.* has to do with respect to how the communication graph is chosen. At first thought, one might think that parameterizing the network functionality with the communication graph does the trick. This is, however, not the case because the parameters of hybrid-functionalities are known to the protocol which invokes them and are therefore also known to the adversary. The only information which is not known to the adversary are inputs of corrupted parties and internal randomness of the functionality; thus, as a second attempt, one might try to have the network functionality sample the communication graph from a given distribution.[11] Unfortunately this also fails to capture the topology-hiding property in full, as we would like to make sure that the adversary (or simulator) gets no information on any *given* (hidden) graph.

Motivated by the above, [MOR15] defines topology-hiding computation using the following trick: they assume an extra incorruptible party, whose only role is to provide the network graph as input to the network functionality. Because this network-choosing party is (by assumption) honest, the simulator cannot see its input and needs to work having only the knowledge that $\mathcal{N}$ allows him to obtain, i.e., the neighborhood of corrupted parties.

In this work we take a slightly different, but equivalent in its effect, approach to avoid the above hack of including a special purpose honest party. We assume that each party provides its desired neighborhood to $\mathcal{N}$ as (a special part of) its input. Since the inputs are explicitly chosen by the environment, we are effectively achieving the same topology-hiding property as [MOR15] but without the extra special-purpose honest party.

In the remainder of this section we provide a formal specification of our network functionality (also referred to as network resource) and our formal security definition of topology-hiding computation.

The Network The network topology is captured by means of an undirected graph $G = (V, E)$ with vertex-set $V = \mathcal{P}$ and edge-set $E \subseteq \mathcal{P} \times \mathcal{P}$. An edge $(P_v, P_u) \in E$ indicates that P_u is in the neighborhood of P_v, which, intuitively, means that P_u and P_v can communicate over a bilateral secure channel. For a party P_v denote by $\mathbf{N}_G(v)$ its neighborhood in G. We will refer to $\mathbf{N}_G[v] = \{P_v\} \cup \mathbf{N}_G(v)$ as P_v's closed neighborhood. Furthermore let $\mathbf{N}_G[v]^k$ be all nodes in G which have distance k or less to P_v. (Clearly $P_v \in \mathbf{N}_G[v]^k$.)

The network functionality allows two types of access: (1) any party $P_v \in \mathcal{P}$ can submit its neighbors $\mathbf{N}_G[v]$, and (2) every party can submit a vector $\vec{m}$ of messages, one for each of its neighbors, which are then delivered in a batch form to their intended recipients. In order to be able to make statements for restricted classes of graphs, e.g., expanders, we parameterize the network functionality by a family $\mathcal{G}$ of setups and require that $\mathcal{N}_{\mathcal{G}}$ only allows (the environment on behalf of) the honest parties to chose their neighborhood from this class. Note, that the adversary is not bound to choose a neighborhood from a graph in $\mathcal{G}$, i.e., any valid neighborhood is accepted for corrupted parties. This is not an issue in the semi-honest setting considered in this work as a semi-honest adversary will submit whatever input the environment hands it. Thus, for the semi-honest case it suffices that the functionality becomes unavailable (halts) upon receiving an invalid neighborhood from the adversary (or from some honest party). [12] In the

[10] In any case, our protocol will not output anything other than the output of the functionality, hence the simulator will only use $\mathcal{N}$ to learn the corrupted parties neighborhood.

[11] Intuitively, this would correspond to the hidden graph model of [CCG+15].

[12] Note that the environment knows/chooses all the inputs and therefore knows whether or not the submitted neighborhoods are allowed by the graph class.

full version of this paperwe also describe a network functionality that adequately captures the guarantees needed to prevent a malicious adversary from using the check of whether or not the neighborhood he submits results in an invalid-graph message from $\mathcal{N}_{\mathcal{G}}$ to obtain information on the neighborhood of honest parties.

In the description of $\mathcal{N}_{\mathcal{G}}$ we use the following notation: For a graph G with vertex set V, and for any $V' \subseteq V$, we denote by $G|_{V'}$ the restriction of G to the vertices in V', i.e., the graph that results by removing from G all vertices in $V \setminus V'$ and their associated edges.

Functionality $\mathcal{N}_{\mathcal{G}}$

The network initializes a topology graph $G = (V, E) := (\mathcal{P}, \emptyset)$.

Info Step:

1. Every party $P_i \in \mathcal{P}$ (and the adversary on behalf of corrupted parties) sends (input) $(\texttt{MyNeigborhood}, \mathbf{N}_G[i])$ to $\mathcal{N}$; if $\mathbf{N}_G[i]$ is a valid neighborhood for P_i, i.e., $\mathbf{N}_G[i] \subseteq \{(P_i, P_j) \mid P_j \in \mathcal{P}\}$, then $\mathcal{N}_{\mathcal{G}}$ updates $E := E \cup \mathbf{N}_G[i]$.
2. If there exist no $G' \in \mathcal{G}$ such that $G' = G$ then $\mathcal{N}_{\mathcal{G}}$ sets $E := \emptyset$ and halts. (Every future input is answered by outputting a special symbol $(\texttt{BadNetwork})$ to the sender of this input.)

Communication Step:

1. For each $P_i \in \mathcal{P}$ let $\mathbf{N}_G(i) = \{P_{i_1}, \ldots, P_{i_{\nu_i}}\}$.
2. Every $P_i \in \mathcal{P}$ sends $\mathcal{N}_{\mathcal{G}}$ input $(\texttt{send}, \vec{m}_i)$, where $\vec{m}_i = (m_{i,i_1}, \ldots, m_{i,i_{\nu_i}})$; if P_i does not submit a vector $\vec{m}_i$ of the right size or format, then $\mathcal{N}_{\mathcal{G}}$ adopts $\vec{m}_i = (\perp, \ldots, \perp)$.
3. Every P_i receives (output) $\vec{m}^i = (m_{i_1,i}, \ldots, m_{i_{\nu_i},i})$ from $\mathcal{N}_{\mathcal{G}}$.

An important feature of the above functionality is that the communication pattern (i.e., which parties send or receive messages) does not reveal to the adversary any information other than the neighborhood of corrupted parties. Thus, the simulator cannot use this functionality in the ideal world to extract information about the network. However, when using this network-functionality (in the real-world protocol) to emulate, e.g., a complete communication network, the adversary might use the messages exchanged in the protocol to extract information that the simulator cannot. In fact, the challenge of a topology-hiding protocol is exactly to ensure that the exchanged messages cannot be used by the adversary in such a way.

Definition 1. *Let $\mathcal{G}$ be a family of graphs with vertex set $\mathcal{P}$. Let also $\mathcal{F}$ be a functionality and $\mathcal{N}_{\mathcal{G}}$ denote the network functionality (as specified above) and π be a $\mathcal{N}_{\mathcal{G}}$-hybrid protocol. We say that $\pi^{\mathcal{N}_{\mathcal{G}}}$ securely realizes the functionality $\mathcal{F}$ in a topology-hiding manner with respect to network class $\mathcal{G}$ if and only if π securely realizes the composite functionality $\{\mathcal{F}, \mathcal{N}_{\mathcal{G}}\}$.*

3 Topology-Hiding Communication

In this section we present a construction which allows to securely and topology-hidingly realize different types of communication channels using black-box PKE. The section consists of the following four steps, each treated in a separate sub-section.

RR-MHT-PKE: In Section 3.1 we introduce *multi-homomorphic threshold encryption with reversible randomization* (RR-MHT-PKE), a special type of threshold public-key encryption. In addition to the (common) homomorphic property of ciphertexts RR-MHT-PKE features homomorphic public-keys and decryption-shares. This allows for a decentralized generation of shared keys which enables parties to generate securely and topology-hidingly a public-key setup where the private-key is shared among all parties. Its reversible randomization property allows parties to transmit public-keys and/or ciphertexts through the network such that the adversary can not track them. We also give a practical implementation of RR-MHT-PKE based on the DDH assumption (see Appendix B).

Topology-Hiding Encryption: In Section 3.2, we present a topology-hiding threshold encryption protocol based on *black-box* RR-MHT-PKE. More precisely, we provide (1) a distributed setup protocol, (2) an information-transmission protocol, and (3) a distributed decryption protocol.

Topology-hiding Boolean-OR: In Section 3.3 we present a protocol which, for networks with moderate degree and diameter, securely and topology-hidingly realizes the multiparty Boolean-OR functionality using the topology-hiding threshold encryption protocol from the previous section.

Topology-hiding Broadcast and Secure Channels: Finally, in Section 3.4 we use the Boolean-OR functionality to securely and topology-hidingly realize secure channels and broadcast. The main result of this section is the following theorem.

Theorem 1. *Given a network $\mathcal{N}_{\mathcal{G}}$ with diameter D and maximum degree d where $d^D = \mathrm{poly}(\kappa)$ there exists a protocol which securely and topology-hidingly realizes broadcast using black-box RR-MHT-PKE. The protocol communicates at most $(d+1)^D \cdot n \cdot \lambda$ bits within $5 \cdot D$ rounds, where λ is linear (with small constant, less than 5) in κ.*

3.1 Multi-Homomorphic Threshold Encryption with Reversible Randomization

In this section we introduce *multi-homomorphic threshold encryption with reversible randomization*, a special type of threshold public-key encryption, which will allow us to securely and topology-hidingly realize a distributed encryption scheme. We first start by recalling some standard definitions. A public-key encryption (PKE) scheme consists of three algorithms, `KeyGen` for key generation, `Enc` for encryption and `Dec` for decryption. Since in this work we consider semi-honest adversaries, we will only need encryption satisfying the standard IND-CPA security definition. For completeness this definition is provided in Appendix A. *Threshold* public-key encryption (T-PKE) is PKE in which the private key SK is distributed among l parties $p_1, \dots, p_l$, such that each party p_i holds a share (aka sub-key) sk_i of SK with the property that any $l-1$ sub-keys have no information on SK. Importantly, such a scheme allows for distributed decryption of any given ciphertext: any party p_i can locally compute, using its own sub-key sk_i of the private key SK, a decryption share x_i, so that if someone gets a hold of decryption shares (for the same c) from all parties (i.e., with each of the shares of the private key) he can combine them and recover the plaintext. For the classical definition of T-PKE we refer to Appendix A. *Homomorphic* (threshold) PKE allows to add up encrypted messages. Here, the message space $\langle \mathcal{M}, + \rangle$ and the ciphertext space $\langle \mathcal{C}, \cdot \rangle$ are groups such that $m_1 + m_2 = \mathtt{Dec}(\mathsf{SK}, \mathtt{Enc}(\mathsf{PK}, m_1; r_1) \cdot \mathtt{Enc}(\mathsf{PK}, m_2; r_2))$. for any key pair $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathtt{KeyGen}$ and any messages $m_1, m_2 \in \mathcal{M}$.

Multi-Homomorphic Threshold Encryption We first present *multi-homomorphic threshold encryption* which is in essence HT-PKE with two additional properties. The first property is a decentralized key-generation. The idea is that parties locally generate public/private-key pairs. By combining those local public keys they can then generate a public key with shared private-key where the local private keys act as key shares. More formally, its required that the public-key space $\langle \mathcal{PK}, \cdot \rangle$ and the private-key space $\langle \mathcal{SK}, + \rangle$ are groups. Moreover its is required (1) that there exists a *key-generation algorithm* `KeyGen`, which outputs a public/private-key pair $(\mathsf{pk}_i, \mathsf{sk}_i) \in \mathcal{PK} \times \mathcal{SK}$, and (2) that for any key pairs $(\mathsf{pk}_1, \mathsf{sk}_1), (\mathsf{pk}_2, \mathsf{sk}_2) \in \mathcal{PK} \times \mathcal{SK}$ it holds that $\mathsf{pk}_1 \cdot \mathsf{pk}_2$ is the public key corresponding to private key $\mathsf{sk}_1 + \mathsf{sk}_2$. In other words a multi-homomorphic threshold encryption scheme is homomorphic with respect to public/private keys. We point out this is not a standard property of threshold PKE schemes. For instance, the scheme of [Pai99], does not satisfy this property. Secondly, a versatile homomorphic threshold encryption scheme is required to be homomorphic with respect to decryption shares and private keys. That is, for any key pairs $(\mathsf{pk}_1, \mathsf{sk}_1), (\mathsf{pk}_2, \mathsf{sk}_2)$ and any ciphertext c it must hold that $\mathtt{ShareDecrypt}(\mathsf{sk}_1, c) \cdot \mathtt{ShareDecrypt}(\mathsf{sk}_2, c) = \mathtt{ShareDecrypt}(\mathsf{sk}_1 + \mathsf{sk}_2, c)$.

Definition 2. *A* multi-homomorphic threshold encryption *(MHT-PKE) scheme with security parameter κ consists of four spaces $\mathcal{M}$, $\mathcal{C}$, $\mathcal{SK}$, and $\mathcal{PK}$ and four algorithms* `KeyGen`,`Enc`,`ShareDecrypt`, *and* `Combine` *which are parametrized by κ where:*

1. *The message space $\langle \mathcal{M}; + \rangle$, the ciphertext space $\langle \mathcal{C}; \cdot \rangle$, the public-key space $\langle \mathcal{PK}; \cdot \rangle$, the private-key space $\langle \mathcal{SK}; + \rangle$, and the decryption-share space $\langle \mathcal{DS}; \cdot \rangle$ are cyclic groups of prime order.*
2. *The (probabilistic) key-generation algorithm* `KeyGen` *outputs a public key $\mathsf{pk} \in \mathcal{PK}$ and a private key $\mathsf{sk} \in \mathcal{SK}$ where for any key pairs $(\mathsf{pk}_1, \mathsf{sk}_1)$, $(\mathsf{pk}_2, \mathsf{sk}_2) \in \mathcal{PK} \times \mathcal{SK}$ it holds that $\mathsf{pk}_1 \cdot \mathsf{pk}_2$ is the public key corresponding to private key $\mathsf{sk}_1 + \mathsf{sk}_2$.*
3. *The (probabilistic)* encryption algorithm `Enc` *takes a public key $\mathsf{pk} \in \mathcal{PK}$ and a message $m \in \mathcal{M}$ and outputs a ciphertext $c \leftarrow \mathtt{Enc}(\mathsf{PK}, m; r)$.*

4. *The* decryption share algorithm $\texttt{ShareDecrypt}$ *takes a private key* $\mathsf{sk}_i \in \mathcal{SK}$ *and a ciphertext* $c \in \mathcal{C}$ *as inputs and outputs a decryption share* $\mathsf{x}_i \leftarrow \texttt{ShareDecrypt}(\mathsf{sk}_i, c)$*. For any ciphertext* $c \in \mathcal{C}$ *and private keys* $\mathsf{sk}_1, \mathsf{sk}_2 \in \mathcal{SK}$ *where* $\mathsf{x}_1 \leftarrow \texttt{ShareDecrypt}(\mathsf{sk}_1, c)$ *and* $\mathsf{x}_2 \leftarrow \texttt{ShareDecrypt}(\mathsf{sk}_2, c)$ *it holds that* $\mathsf{x}_1 \cdot \mathsf{x}_2 = \texttt{ShareDecrypt}(\mathsf{sk}_1 + \mathsf{sk}_2, c)$*.*
5. *The* combining algorithm $\texttt{Combine}$ *takes a decryption share* $\mathsf{x} \in \mathcal{DS}$ *and a ciphertext* $c \in \mathcal{C}$ *and outputs a message* $m \leftarrow \texttt{Combine}(\mathsf{x}, c)$*.*

A MHT-PKE scheme satisfies the following correctness property: For any key pairs $(\mathsf{pk}_1, \mathsf{sk}_1), \dots, (\mathsf{pk}_l, \mathsf{sk}_l) \leftarrow$ $\texttt{KeyGen}$ *and any message* $m \in \mathcal{M}$ *it holds that* $m = \texttt{Combine}(\mathsf{x}_1 \cdot \dots \cdot \mathsf{x}_l, c)$ *where* $\mathsf{x}_i = \texttt{ShareDecrypt}(\mathsf{sk}_i, c)$*,* $c = \texttt{Enc}(\mathsf{pk}, m; r)$ *and* $\mathsf{pk} = \mathsf{pk}_1 \cdot \dots \cdot \mathsf{pk}_l$*. Moreover, given a message* m *and a ciphertext* c *one can efficiently invert* $\texttt{Combine}$*, i.e., compute a decryption share* x *with* $m = \texttt{Combine}(\mathsf{x}, c)$*.*

We define the security of MHT-PKE with respect to a threshold variant of the IND-CPA security definition.

Definition 3. *A MHT-PKE scheme is* IND-TCPA *secure if the adversary's advantage in winning the following game is negligible in* κ*.*

1. *The game generates key pairs* $(\mathsf{pk}_1, \mathsf{sk}_1), \dots (\mathsf{pk}_l, \mathsf{sk}_l) \leftarrow \texttt{KeyGen}$ *and chooses a random bit* b*. Then the adversary gets* $\mathsf{pk} = \mathsf{pk}_1 \cdot \dots \cdot \mathsf{pk}_l$*,* $\mathsf{pk}_1, \dots, \mathsf{pk}_l$ *and* $\mathsf{sk}_2, \dots, \mathsf{sk}_l$*. This allows him to generate encryptions of arbitrary messages and to generate decryption shares for all key pairs except* $(\mathsf{pk}_1, \mathsf{sk}_1)$*.*
2. *The adversary specifies two messages* m_0 *and* m_1 *and the game returns* $c = \texttt{Enc}(\mathsf{PK}, m_b)$*.*
3. *The adversary specifies a bit* b'*. If* $b = b'$ *the adversary has won the game.*

Furthermore for any chosen public-key $\mathsf{pk} \in \mathcal{PK}$, it should be hard to distinguish between $(\mathsf{pk}, \mathsf{pk} \cdot \mathsf{pk}_1)$ and $(\mathsf{pk}, \mathsf{pk}_2)$ where $\mathsf{pk}_1, \mathsf{pk}_2$ are distributed according to $\texttt{KeyGen}$. More formally, we require that the scheme has the *indistinguishability under chosen public-key attack* (IND-CKA) property.

Definition 4. *A MHT-PKE scheme is* IND-CKA *secure if the adversary's advantage in winning the following game is negligible in* κ*.*

1. *The adversary specifies a public key* $\mathsf{pk} \in \mathcal{PK}$*.*
2. *The game generates a key pair* $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \texttt{KeyGen}$ *and chooses a uniform random bit* b*. Then the adversary gets public key* pk_2 *where*

$$\mathsf{pk}_2 = \begin{cases} \mathsf{pk}_1 & \text{if } b = 0 \\ \mathsf{pk}_1 \cdot \mathsf{pk} & \text{if } b = 1 \end{cases}$$

3. *The adversary specifies a bit* b'*. If* $b = b'$ *the adversary has won the game.*

Reversible Randomization We can now introduce multi-homomorphic threshold encryption with *reversible randomization* which is MHT-PKE with additional randomization properties.

Randomization of Public Keys The first property required is the randomization of public keys. More concretely, a MHT-PKE with reversible randomization allows a party P_i with public key pk_i to "randomize" pk_i, i.e., compute a new masked public-key $\widetilde{\mathsf{pk}}_i$ so that anyone seeing $\widetilde{\mathsf{pk}}_i$ is unable to tell whether it is a freshly generated public-key or a randomized version of pk_i. Importantly, we require the randomization algorithm to be reversible in the following sense. The randomization algorithm must provide P_i with information rk_i, the *de-randomizer*, which allows it to map any encryption with $\widetilde{\mathsf{pk}}_i$ back to an encryption with its original key pk_i. Looking ahead, the randomization of public-keys property will ensure that the adversary can not trace public keys while they travel the network. This allows us to build a topology-hiding information-transmission protocol.

Randomization of Ciphertexts The second property required is the randomization of ciphertexts. More concretely, a MHT-PKE with reversible randomization allows a party P_i with ciphertext c_i to "randomize" c_i, i.e., compute a new masked ciphertext $\widehat{c}_i$ so that anyone seeing $\widehat{c}_i$ is unable to tell whether it is a freshly generated ciphertext (using an arbitrary public-key) or an randomized version of c_i. Importantly, we require the randomization algorithm to be reversible. This means it must provide P_i with information rk_i, the *de-randomizer*, which allows it to map any decryption share of $\widehat{c}_i$ and decryption key sk back to a decryption share of the original ciphertext c_i and sk. Looking ahead, the randomization of ciphertexts will ensure that the adversary can not trace ciphertexts and decryption-shares while they travel the network. This will allow us to build a topology-hiding decryption protocol. We remark that this property differs from the usual ciphertext re-randomization in homomorphic PKE schemes where one randomizes a ciphertext by adding up an encryption of 0.

MHT-PKE with Reversible Randomization We can now give the formal definition of a MHT-PKE with reversible-randomization scheme.

Definition 5. *A* MHT-PKE with reversible-randomization *(RR-MHT-PKE) scheme is a MHT-PKE scheme with extra algorithms* $\texttt{RandKey}$*,* $\texttt{DerandCipher}$*,* $\texttt{RandCipher}$*,* $\texttt{DerandShare}$ *where:*

1. *The (probabilistic)* (key) randomization algorithm $\texttt{RandKey}$ *takes a public key* $\mathsf{pk} \in \mathcal{PK}$ *and outputs a new public key* $\widetilde{\mathsf{pk}} \in \mathcal{PK}$ *and* a de-randomizer $\mathsf{rk} \in \mathcal{RK}_P$*.*
2. *The* (ciphertext) de-randomization algorithm $\texttt{DerandCipher}$ *takes a de-randomizer* $\mathsf{rk} \in \mathcal{RK}_P$ *and a ciphertext* $\widetilde{c} \in \mathcal{C}$ *and outputs a new ciphertext* $c \in \mathcal{C}$ *such that the following property holds. For any key pair* $(\mathsf{pk}, \mathsf{sk})$*,* $(\widetilde{\mathsf{pk}}, \mathsf{rk}) \leftarrow \texttt{RandKey}(\mathsf{pk}; r')$*, any message* $m \in \mathcal{M}$*, and any ciphertext* $\widetilde{c} \leftarrow \texttt{Enc}(\widetilde{\mathsf{pk}}, m; \tilde{r})$ *there exists an* r *such that* $\texttt{Enc}(\mathsf{pk}, m; r) = \texttt{DerandCipher}(\mathsf{rk}, \widetilde{c})$*. Moreover, given a ciphertext* c *and a de-randomizer* rk *one can efficiently invert* $\texttt{DerandCipher}$*, i.e., compute a ciphertext* $\widetilde{c}$ *such that* $c = \texttt{DerandCipher}(\mathsf{rk}, \widetilde{c})$*.*
3. *The (probabilistic)* (ciphertext) randomization algorithm $\texttt{RandCipher}$ *takes a ciphertext* $c \in \mathcal{C}$ *and outputs a new ciphertext* $\hat{c} \in \mathcal{C}$ *and a de-randomizer* $\mathsf{rk} \in \mathcal{RK}_C$*.*
4. *The* (share) de-randomization algorithm $\texttt{DerandShare}$ *takes a de-randomizer* $\mathsf{rk} \in \mathcal{RK}_C$ *and a decryption share* $\hat{\mathsf{x}} \in \mathcal{DS}$ *and outputs a share* $\mathsf{x} \in \mathcal{DS}$ *such that the following property holds. For any key pair* $(\mathsf{pk}, \mathsf{sk})$*, any ciphertext* $c \in \mathcal{C}$*,* $(\mathsf{rk}, \hat{c}) \leftarrow \texttt{RandCipher}(c; r)$*, and* $\hat{\mathsf{x}} \leftarrow \texttt{ShareDecrypt}(\mathsf{sk}_i, \hat{c})$ *we have* $\texttt{DerandShare}(\mathsf{rk}, \hat{\mathsf{x}}) = \texttt{ShareDecrypt}(\mathsf{sk}_i, c)$*. More over given a decryption share* x *and a de-randomizer* rk *one can efficiently invert* $\texttt{DerandShare}$*, i.e., compute a decryption shares* $\hat{\mathsf{x}}$ *such that* $\mathsf{x} = \texttt{DerandShare}(\mathsf{rk}, \hat{\mathsf{x}})$*.*

For any public key pk it should be hard (for the adversary) to distinguish between $(\mathsf{pk}, \texttt{RandKey}(\mathsf{pk}))$ and $(\mathsf{pk}, \mathsf{pk}')$ where pk' is freshly generated using $\texttt{KeyGen}$. Similar, for any ciphertext c it should be hard to distinguish between $(c, \texttt{RandCipher}(c))$ and (c, c') where c' is a randomly chosen ciphertext. More formally, the scheme should have the *indistinguishability under chosen public-key and chosen ciphertext attack* (IND-CKCA) property.

Definition 6. *A RR-MHT-PKE scheme is* IND-CKCA *secure if the adversary's advantage in winning the following game is negligible in* κ*.*

1. *The adversary specifies a public key* $\mathsf{pk} \in \mathcal{PK}$ *and a ciphertext* $c \in \mathcal{C}$*.*
2. *The game generates key pairs* $(\mathsf{pk}_1, \mathsf{sk}_1), (\mathsf{pk}_2, \mathsf{sk}_2) \twoheadleftarrow \texttt{KeyGen}$ *and a uniform random message* $m \in \mathcal{M}$*. The game then chooses uniform random bits* b_1 *and* b_2*. The adversary gets public key* $\widetilde{\mathsf{pk}}$ *and ciphertext* $\widehat{c}$ *where*
$$\widetilde{\mathsf{pk}} = \begin{cases} \texttt{RandKey}(\mathsf{pk}) & \textit{if } b_1 = 0 \\ \mathsf{pk}_1 & \textit{if } b_1 = 1 \end{cases}$$
and
$$\widehat{c} = \begin{cases} \texttt{RandCipher}(c) & \textit{if } b_2 = 0 \\ \texttt{Enc}(\mathsf{pk}_2, m) & \textit{if } b_2 = 1 \end{cases}.$$
3. *The adversary specifies bits* b'_1 *and* b'_2*. If* $b_1 = b'_1$ *or* $b_2 = b'_2$ *the adversary has won the game.*

The security of a RR-MHT-PKE scheme is defined with respect to the above security properties.

Definition 7. *A RR-MHT-PKE scheme is* secure *if it is* IND-TCPA*,* IND-CKA*, and* IND-CKCA *secure.*

DDH based RR-MHT-PKE One can practically implement secure RR-MHT-PKE using an extended variant of the ElGamal cryptosystem [ElG84] over a group G of prime order $q(\kappa)$ where the DDH assumption holds. We refer to Appendix B for more details.

Lemma 1. *Given a DDH group one can securely implement RR-MHT-PKE.*

3.2 Topology-Hiding Threshold Encryption

In this section we build a topology-hiding threshold encryption protocol using a secure RR-MHT-PKE scheme. More precisely, we provide (1) a distributed setup protocol, (2) an information-transmission protocol, and (3) a distributed decryption protocol. Looking ahead, those protocols will allow us to topology-hidingly realize the Boolean-OR functionality.

The RR-MHT-PKE Scheme: We assume that the parties have access to a secure RR-MHT-PKE scheme with security parameter κ, where $n = \text{poly}(\kappa)$. In particular, each party has local (black-box) access to the algorithms of the RR-MHT-PKE scheme.

The Network Graph: A prerequisite for our protocols to work is that the network graph G of $\mathcal{N}_{\mathcal{G}}$ is connected. Otherwise (global) information transmission is not possible. The parties also need to know upper bounds on the maximum degree and the diameter of the network graph. We therefore assume that the parties have access to an initialized network $\mathcal{N}_{\mathcal{G}}^{d,D}$ where the graphs in the family $\mathcal{G}$ are connected, have a maximum degree of $d \leq n$, and a diameter of at most $D \leq n$ where d and D are publicly known. For simplicity we restrict ourselves to present protocols for d-regular network graphs. We point out that one can extend the presented protocols to the general case where parties may have less than d neighbors. The idea is that a party which lacks d neighbors pretends to have d neighbors by emulating (messages from) virtual neighbors (cf. [MOR15]).

Setup Protocol In this section we present a protocol which allows to topology-hidingly generate a threshold-setup where each party P_i holds a public key PK_i such that the corresponding private-key is shared among all parties. The high-level idea of our protocol is as follows. We first observe that the D-neighborhood of P_i consists of all parties. The setup thus provides party P_i with a public key where the corresponding private-key is shared among the parties in the D-neighborhood $\mathbf{N}_G[i]^D$ of P_i. This implies that one can generate the setup recursively. In order to generate a k-neighborhood public-key $\mathsf{PK}_i^{(k)}$, P_i asks each of its neighbors to generate a public key where the private key is shared in the neighbors $(k-1)$-neighborhood. It can then compute $\mathsf{PK}_i^{(k)}$ by combining the received public-keys.

Definition 8. *A* setup for topology-hiding threshold encryption over a network $\mathcal{N}_{\mathcal{G}}^{d,D}$ *consists of the following parts.*

Private-Key Shares: *Each party P_i holds a vector $(\overline{\mathsf{SK}}_i^{(0)}, \ldots, \overline{\mathsf{SK}}_i^{(D)})$ of $D+1$ private keys which we call its* private-key shares. *For any $0 \leq k \leq D$ we denote by $\overline{\mathsf{PK}}_i^{(k)}$ the public key corresponding to $\overline{\mathsf{SK}}_i^{(r)}$.*

Public-Keys: *Each party P_i holds a vector $(\mathsf{PK}_i^{(0)}, \ldots, \mathsf{PK}_i^{(D)})$ of $D+1$ public keys where $\mathsf{PK}_i^{(0)} = \overline{\mathsf{PK}}_i^{(0)}$ and $\mathsf{PK}_i^{(k)} = \overline{\mathsf{PK}}_i^{(k)} \cdot \prod_{P_j \in \mathbf{N}_G(i)} \mathsf{PK}_j^{(k-1)}$. We call $\mathsf{PK}_i^{(k)}$ the* level-k public-key *of P_i and denote by $\mathsf{SK}_i^{(r)}$ the corresponding (shared) private key. The* public-key *of P_i is $\mathsf{PK}_i \coloneqq \mathsf{PK}_i^{(D)}$ and the* shared private-key *is $\mathsf{SK}_i \coloneqq \mathsf{SK}_i^{(D)}$.*

Local Pseudonyms: *Each party P_i privately holds a injective random function $\nu_i(\cdot) : \mathbf{N}_G(i) \to \{1, \ldots, d\}$ which assigns each neighbor $P_j \in \mathbf{N}_G(i)$ a unique local identity $\nu_i(j) \in \{1, \ldots, d\}$. W.l.o.g. we will assume that $\nu_i(i) = 0$.*

We remark that the condition on the public-keys ensures that any $0 \leq k \leq D$ (and for reasonably large $\mathcal{PK}$) the private key $\mathsf{SK}_i^{(k)}$ is properly shared among the k neighborhood of P_i, i.e., each party in the k-neighborhood holds a non-trivial share.

Definition 9. *A protocol is a* secure (topology-hiding) setup protocol over a network $\mathcal{N}_{\mathcal{G}}^{d,D}$ *if it has the following properties.*

Correctness: *The protocol generates with overwhelming probability a setup for topology-hiding threshold encryption over the network $\mathcal{N}_{\mathcal{G}}^{d,D}$.*

Topology-Hiding Simulation: *The adversarial view in an actual protocol-execution can be simulated with overwhelming probability given the neighborhood of dishonest parties in $\mathcal{N}_{\mathcal{G}}^{d,D}$ and the output of dishonest parties, i.e., given the values*

$$\{\mathbf{N}_G(i), \nu_i(\cdot), \overline{\mathsf{SK}}_i^{(0)}, \ldots, \overline{\mathsf{SK}}_i^{(D)}, \mathsf{PK}_i^{(0)}, \ldots, \mathsf{PK}_i^{(D)}\}_{P_i \in \overline{H}}$$

The simulation property ensures in particular that (a) the adversary does not learn more about the network topology and that (b) the adversary does not learn the private key corresponding to the public key PK_i of party P_i unless it corrupts the entire k-neighborhood of P_i.

Protocol `GenerateSetup`

```
Require: Parties have access to an initialized $\mathcal{N}_{\mathcal{G}}^{d,D}$.
1: Each $P_i$ generates the local identities $\nu_i(\cdot)$ and sub-key pair $(\overline{\mathsf{PK}}_i^{(0)}, \overline{\mathsf{SK}}_i^{(0)}) \twoheadleftarrow \mathtt{KeyGen}$. Then it sets $\mathsf{PK}_i^{(0)} = \overline{\mathsf{PK}}_i^{(0)}$.
2: for $k = 1, \ldots, D$ do
3:     Each $P_i$ sends $\mathsf{PK}_i^{(k-1)}$ to each $P_j \in \mathbf{N}_G(i)$ using $\mathcal{N}$.
4:     Each $P_i$ generates sub-key pair $(\overline{\mathsf{PK}}_i^{(k)}, \overline{\mathsf{SK}}_i^{(k)}) \twoheadleftarrow \mathtt{KeyGen}$.
5:     Each $P_i$ computes $\mathsf{PK}_i^{(k)} = \overline{\mathsf{PK}}_i^{(k)} \cdot \prod_{P_j \in \mathbf{N}_G(i)} \mathsf{PK}_j^{(k-1)}$.
6: end for
Output: $P_i$ outputs $\nu_i(\cdot)$, $(\overline{\mathsf{SK}}_i^{(0)}, \ldots, \overline{\mathsf{SK}}_i^{(D)})$, and $(\mathsf{PK}_i^{(0)}, \ldots, \mathsf{PK}_i^{(D)})$.
```

Lemma 2. *Given a secure RR-MHT-PKE scheme the protocol* `GenerateSetup` *is a secure setup protocol. The protocol communicates $D \cdot d \cdot n \cdot \log|\mathcal{PK}|$ bits within D rounds.*

Proof. (sketch) **Correctness:** It follows directly from protocol inspection that the setup generated by `GenerateSetup` is valid for $\mathcal{N}_{\mathcal{G}}^{d,D}$. **Topology-Hiding Simulation:** The view of the adversary during an actual protocol execution is

$$\{\mathbf{N}_G(i), \nu_i(\cdot), \left\{\mathsf{PK}_i^{(k)}, \overline{\mathsf{PK}}_i^{(k)}, \overline{\mathsf{SK}}_i^{(k)}\right\}_{0 \le r \le D}, \left\{\mathsf{PK}_j^{(k)}\right\}_{P_j \in \mathbf{N}_G(i), 0 \le r \le D-1}\}_{P_i \in \overline{H}}.$$

Now consider the view where the public keys $\left\{\mathsf{PK}_j^{(k)}\right\}_{P_j \in \mathbf{N}_G(i) \cap H, 0 \le r \le D-1}$ are replaced by freshly generated public keys using `KeyGen`, i.e.,

$$\{\mathbf{N}_G(i), \nu_i(\cdot), \left\{\mathsf{PK}_i^{(k)}, \overline{\mathsf{PK}}_i^{(k)}, \overline{\mathsf{SK}}_i^{(k)}\right\}_{0 \le r \le D}, \left\{\widetilde{\mathsf{PK}}_j^{(k)}\right\}_{P_j \in \mathbf{N}_G(i) \cap H, 0 \le r \le D-1}\}_{P_i \in \overline{H}}.$$

Note that the second view can be easily computed by a simulator given the outputs of dishonest parties. It remains to show that those views are computationally indistinguishable. Note that for any $P_j \in \mathbf{N}_G(\overline{H}) \cap H$ the public-key $\mathsf{PK}_j^{(k)}$ has the form $\mathsf{pk}_1 \cdot \mathsf{pk}$ where $\mathsf{pk}_1 = \overline{\mathsf{PK}}_j^{(k)}$ and $\mathsf{pk} = \prod_{P_i \in \mathbf{N}_G(j)} \mathsf{PK}_i^{(k-1)}$. The indistinguishability therefore follows from the IND-CKA security of the RR-MHT-PKE scheme. **Communication Complexity:** The protocol runs for D rounds and in each round $n \cdot d$ public-keys are sent.

Information-Transmission Protocol In this section we present a topology-hiding information-transmission protocol. Here, each party has a message m_i and a public-key pk_i[13] as input. The output of party P_i is a ciphertext c_i under the public key pk_i. If all parties input the 0-message, c_i is an encryption of 0. Otherwise, c_i is an encryption of a random, non-zero message. The information-transmission protocol has a recursive structure and is thus parametrized by a level k. The protocol requires that parties have generated local pseudonyms. We therefore assume that the parties have access to a setup for topology-hiding threshold encryption over $\mathcal{N}_{\mathcal{G}}^{d,D}$.

Definition 10. *A protocol is a* level-k (topology-hiding) secure information-transmission protocol over a network $\mathcal{N}_{\mathcal{G}}^{d,D}$ *if it has the following properties.*

Setup, Inputs, and Outputs: *The parties initially hold a setup for topology-hiding threshold encryption over $\mathcal{N}_{\mathcal{G}}^{d,D}$* (cf. *Definition 8). Each party holds as input a message $m_i \in \mathcal{M}$ and a public key $\mathsf{pk}_i \in \mathcal{PK}$ (not necessarily part of its setup).The output of each party P_i is a ciphertext $c_i \in \mathcal{C}$.*

[13] For notational simplicity we use uppercase letters for public-/private-keys which are part of the setup for $\mathcal{N}_{\mathcal{G}}^{d,D}$ and lowercase letters for arbitrary public-/private-keys.

Correctness: *With overwhelming probability the output c_i is the encryption of message s_i under pk_i and randomness ρ_i (i.e. $c_i = \mathtt{Enc}(\mathsf{pk}_i, s_i; \rho_i)$) with*

$$s_i = \begin{cases} 0 & \text{if } m_j = 0 \text{ for all } P_j \in \mathbf{N}_G[i]^k \\ x_i & \text{if } m_j \neq 0 \text{ for at least one } P_j \in \mathbf{N}_G[i]^k \end{cases}$$

where $x_i \in \mathcal{M} \setminus \{0\}$ uniform at random.

Topology-Hiding Simulation: *The adversarial view in a real protocol-execution can be simulated with overwhelming probability given the following values*

$$\{\mathbf{N}_G(i), m_i, \mathsf{pk}_i, c_i, \nu_i(\cdot)\}_{P_i \in \overline{H}} \cup \{s_i, \rho_i\}_{\mathbf{N}_G[i]^k \subseteq \overline{H}}.$$

In other words the simulator gets the neighborhood of dishonest parties (in $\mathcal{N}_{\mathcal{G}}^{d,D}$), their protocol in- and outputs, and their local pseudonyms from the setup. For any party P_i where the whole k-neighborhood is dishonest the simulator is additionally given the content s_i and the randomness ρ_i of output c_i.

The simulation property ensures in particular that (a) the adversary does not learn more about the network topology and that (b) the adversary does not learn the content of ciphertext c_i of party P_i unless it corrupts the entire k-neighborhood of P_i.

Protocol $\mathtt{InfoTransmisson}\big(k, (m_1, \mathsf{pk}_1), \ldots, (m_n, \mathsf{pk}_n)\big)$

```
Require: Parties have access to an initialized $\mathcal{N}_{\mathcal{G}}^{d,D}$ and have generated local pseudonyms.
Input: Each $P_i$ inputs a message $m_i$ and a public key $\mathsf{pk}_i$.
 1: if $k = 0$ then
 2:     Each $P_i$ computes $c_i = \mathtt{Enc}(\mathsf{pk}_i, 0)$ if $m_i = 0$ or $c_i = \mathtt{Enc}(\mathsf{pk}_i, x_i)$ if $m_i \neq 0$ where $x_i \in \mathcal{M} \setminus \{0\}$ uniform at random.
 3: else
 4:     Each $P_i$ computes $(\widetilde{\mathsf{pk}}_i, \mathsf{rk}_i) \leftarrow \mathtt{RandKey}(\mathsf{pk}_i)$ and sends $\widetilde{\mathsf{pk}}_i$ to each $P_j \in \mathbf{N}_G[i]$ which denotes the received key by $\mathsf{pk}_{j,\nu_j(i)}$.
 5:     for $l = 0, \ldots, d$ do
 6:         The parties compute ciphertexts $(\widetilde{c}_{1,l}, \ldots, \widetilde{c}_{n,l})$ by invoking $\mathtt{InfoTransmisson}\big(k - 1, (m_1, \mathsf{pk}_{1,l}), \ldots, (m_n, \mathsf{pk}_{n,l})\big)$.
 7:     end for
 8:     Each $P_i$ sends $\widetilde{c}_{i,\nu_i(j)}$ to $P_j \in \mathbf{N}_G[i]$.
 9:     Each $P_i$ computes $c_i = \big(\prod_{P_j \in \mathbf{N}_G[i]} \mathtt{DerandCipher}(\mathsf{rk}_i, \widetilde{c}_{j,\nu_j(i)})\big)^{r_i}$ for a uniform random $r_i \in \{1, \ldots, |\mathcal{M}| - 1\}$.
10: end if
Output: Each $P_i$ outputs $c_i$.
```

Lemma 3. *Given a secure RR-MHT-PKE scheme and for any parameter $0 \leq k \leq D$ with $d^k = \mathrm{poly}(\kappa)$, $\mathtt{InfoTransmisson}\big(k, (m_1, \mathsf{pk}_1), \ldots, (m_n, \mathsf{pk}_n)\big)$ is a secure level-k information-transmission protocol. The protocol communicates at most $(d+1)^k \cdot n \cdot (\log|\mathcal{PK}| + \log|\mathcal{C}|)$ bits within $2k$ rounds.*

Proof. (sketch) **Correctness:** For $k = 0$ each party locally computes c_i as specified by the correctness property. The protocol thus achieves correctness perfectly. For $k > 0$ assume that the protocol achieves correctness for $(k-1)$. More precisely, the output of a party P_j for parameter $(k-1)$ is computed perfectly correct if all $(k-1)$-neighbors have input 0. Otherwise, the output of P_j for parameter $(k-1)$ is computed correctly except with error probability ε_{k-1}. First, we consider the case where all parties in the k-neighborhood of P_i have input 0. The assumption for $(k-1)$ implies that all $\widetilde{c}_{i,\nu_i(j)}$ contain 0. The properties of the RR-MHT-PKE scheme imply that $s_i = r_i \cdot 0 = 0$. In the second case at least one party in the k-neighborhood of P_i has a non-zero input. This implies that at least one $\widetilde{c}_{i,\nu_i(j)}$ contains a uniform random, non-zero message (with error probability of at most ε_{k-1}). The properties of the RR-MHT-PKE thus ensure that c_i contains a uniform random, non-zero message (except with error probability $\varepsilon_k := \varepsilon_{k-1} + \frac{1}{|\mathcal{M}|}$). This implies an overall success probability of at least $1 - (\frac{k \cdot n}{|\mathcal{M}|})$. **Topology-Hiding Simulation:** To simulate the view of the adversary the simulator is given

$$\{\mathbf{N}_G(i), m_i, \mathsf{pk}_i, c_i, \nu_i(\cdot)\}_{P_i \in \overline{H}} \cup \{s_i, \rho_i\}_{\mathbf{N}_G[i]^k \subseteq \overline{H}}.$$

For $k = 0$ those values correspond exactly to the view of the adversary during an actual protocol execution. Simulation is thus easy. For the case $k > 0$ assume that the view of the adversary can be simulated for $k' < k$. The view of the adversary can now be simulated as follows. At the beginning, the simulator generates all public keys and de-randomizers seen by the adversary. For each dishonest P_i the simulator computes $\mathsf{rk}_i, \widetilde{\mathsf{pk}}_i$ using $\texttt{RandKey}$. For each honest P_j in the neighborhood of $\overline{H}$ the simulator sets $\widetilde{\mathsf{pk}}_j$ to a random public-key using $\texttt{KeyGen}$. Due to the IND-CKCA property of the RR-MHT-PKE scheme these public keys are indistinguishable from the corresponding public-keys seen by the adversary in an actual protocol-execution. The above values also determine all keys $\mathsf{pk}_{i,\nu_i(j)}$ for $P_i \in \overline{H}$ and $P_j \in \mathbf{N}_G(i)$. Now, we consider the ciphertexts seen by the adversary in the second part of the protocol. In essence the simulator must generate all $\widetilde{c}_{j,\nu_j(i)}$ where P_i and/or P_j are dishonest. If the whole $(k-1)$-neighborhood of P_j is dishonest the simulator must also provide the content and the randomness of $\widetilde{c}_{j,\nu_j(i)}$ which are required for the sub-simulation of the recursive protocol invocations. We recall that $\texttt{DerandCipher}$ is efficiently invertible if the de-randomizer is known. First, simulator generates a random $r_i \in \{1, \dots, |\mathcal{M}| - 1\}$ for each dishonest P_i. If the whole k-neighborhood of P_i is dishonest (i.e., $\mathbf{N}_G[i]^k \subseteq \overline{H}$) the simulator is additionally given s_i and ρ_i. This allows the simulator to compute for each neighbor $P_j \in \mathbf{N}_G[i]$ a valid $s_{j,\nu_j(i)}$, randomness $\rho_{j,\nu_j(i)}$, and an encryption $\widetilde{c}_{j,\nu_j(i)} = \texttt{Enc}(\widetilde{\mathsf{pk}}_i, s_{j,\nu_j(i)}; \rho_{j,\nu_j(i)})$ such that $c_i = \big(\prod_{P_j \in \mathbf{N}_G[i]} \texttt{DerandCipher}(\mathsf{rk}_i, \widetilde{c}_{j,\nu_j(i)})\big)^{r_i}$. If there exists a honest party in the k-neighborhood of P_i, the simulator is not given s_i and ρ_i. However, in this case there is at least one P_j in $\mathbf{N}_G[i]$ such that $\mathbf{N}_G[j]^{k-1} \nsubseteq \overline{H}$. This allows the simulator to first generates all $s_{j,\nu_j(i)}, \rho_{j,\nu_j(i)}$ and $\widetilde{c}_{j,\nu_j(i)} = \texttt{Enc}(\widetilde{\mathsf{pk}}_i, s_{j,\nu_j(i)}; \rho_{j,\nu_j(i)})$ where $\mathbf{N}_G[j]^{k-1} \subseteq \overline{H}$. Then it chooses the remaining $\widetilde{c}_{j,\nu_j(i)}$ randomly under the constraint that $c_i = \big(\prod_{P_j \in \mathbf{N}_G[i]} \texttt{DerandCipher}(\mathsf{rk}_i, \widetilde{c}_{j,\nu_j(i)})\big)^{r_i}$. In a final step the adversary generates for any honest $P_j \in \mathbf{N}_G[i]$ the values $s_{i,\nu_i(j)}, \rho_{i,\nu_i(j)}$ and $\widetilde{c}_{i,\nu_i(j)} = \texttt{Enc}(\widetilde{\mathsf{pk}}_j, s_{i,\nu_i(j)}; \rho_{i,\nu_i(j)})$. The $IND-TCPA$ property of the RR-MHT-PKE scheme and the correctness property of the protocol ensure that the generated ciphertexts are indistinguishable from the ones seen by the adversary in an actual protocol execution. Now all values required for the simulation of the $d+1$ invocations of $\texttt{InfoTransmisson}$ with parameter $(k-1)$ are given. The simulator can thus use the sub-simulator to generate the view of the adversary in the middle part of the protocol. **Communication Complexity:** Let $f(k)$ be the communication complexity of $\texttt{InfoTransmisson}\big(k, \dots\big)$. Then we have $f(0) = 0$ and $f(k) = d \cdot n \cdot (\log|\mathcal{PK}| + \log|\mathcal{C}|) + (d+1) \cdot f(k-1)$. This results in a communication complexity of at most $(d+1)^k \cdot n \cdot (\log|\mathcal{PK}| + \log|\mathcal{C}|)$ bits. The round complexity follows from the observation that one can invoke the subprotocols $\texttt{InfoTransmisson}(k-1, \dots)$ in parallel.

Decryption Protocol In this section we describe a distributed decryption protocol which allows each party P_i to decrypt a ciphertext c_i under its shared private-key SK_i which has been generated by the setup protocol. The decryption protocol consists of two parts. First the parties jointly compute for each ciphertext c_i a decryption-share x_i under the shared private-key of P_i. In a second phase each party P_i can locally decrypt c_i using the decryption share x_i. First, we present a subprotocol which allows to compute the required decryption shares. The key-idea is to use the homomorphic property of decryption-shares which allows a recursive computation. The subprotocol is therefore parametrized by k.

Definition 11. *A protocol is a* secure level-k (topology-hiding) decryption-share protocol *over a network $\mathcal{N}_{\mathcal{G}}^{d,D}$ if it has the following properties.*

Setup, Inputs, and Outputs: *The parties initially hold a setup for topology-hiding threshold encryption over $\mathcal{N}_{\mathcal{G}}^{d,D}$ (*cf.* Definition 8). Each party P_i inputs a ciphertext $c_i \in \mathcal{C}$. The output of party P_i is a decryption share $\mathsf{x}_i \in \mathcal{DS}$.*

Correctness: *With overwhelming probability $\mathsf{x}_i = \texttt{ShareDecrypt}(\mathsf{SK}_i^{(k)}, c_i)$ where $\mathsf{SK}_i^{(k)}$ is the level-k shared private-key of P_i from the setup.*

Topology-Hiding Simulation: *The adversarial view in a real protocol-execution can be simulated with overwhelming probability given the following values*

$$\big\{\mathbf{N}_G(i), c_i, \mathsf{x}_i, \nu_i(\cdot), \overline{\mathsf{SK}}_i^{(0)}, \dots, \overline{\mathsf{SK}}_i^{(k)}\big\}_{P_i \in \overline{H}}$$

In other words the simulator gets the neighborhood of dishonest parties (in $\mathcal{N}_{\mathcal{G}}^{d,D}$), their protocol in- and outputs, their local pseudonyms, and their private-key shares (up to level-k) of the assumed setup.

The simulation property ensures in particular that the adversary does not learn more about the network topology.

Protocol $\texttt{DecShares}(k, c_1, \ldots, c_n)$

```
Require: Parties have access to an initialized $\mathcal{N}_{\mathcal{G}}^{d,D}$ and have generated a setup for topology-hiding
    threshold encryption over $\mathcal{N}_{\mathcal{G}}^{d,D}$.
Input: Each $P_i$ inputs a ciphertext $c_i$.
 1: if $k = 0$ then
 2:     Each $P_i$ computes $\mathsf{x}_i = \texttt{ShareDecrypt}(\overline{\mathsf{SK}}_i^{(0)}, c_i)$.
 3: else
 4:     Each $P_i$ computes $(\mathsf{rk}_i, \widehat{c_i}) = \texttt{RandCipher}(c_i)$ and sends $\widehat{c_i}$ to each $P_j \in \mathbf{N}_G(i)$ which denotes the
        received value by $c_{j,\nu_j(i)}$.
 5:     for $l = 1, \ldots, d$ do
 6:         Parties compute $(\mathsf{x}_{1,l}, \ldots, \mathsf{x}_{n,l}) = \texttt{DecShares}(k-1, c_{1,l}, \ldots, c_{n,l})$.
 7:     end for
 8:     Each $P_i$ sends $\mathsf{x}_{i,\nu_i(j)}$ to each $P_j \in \mathbf{N}_G(i)$.
 9:     Each $P_i$ computes first $\widehat{\mathsf{x}}_i = \prod_{P_j \in \mathbf{N}_G(i)} \mathsf{x}_{j,\nu_j(i)}$ and then $\mathsf{x}_i = \texttt{DerandShare}(\mathsf{rk}_i, \widehat{\mathsf{x}}_i) \cdot$
        $\texttt{ShareDecrypt}(\overline{\mathsf{SK}}_i^{(k)}, c_i)$.
10: end if
Output: Each $P_i$ outputs $\mathsf{x}_i$.
```

Lemma 4. *Given a secure RR-MHT-PKE scheme and for any parameter $0 \le k \le D$ with $d^k = \mathrm{poly}(\kappa)$ the above protocol $\texttt{DecShares}(k, c_1, \ldots, c_n)$ is a secure level-k decryption-share protocol. The protocol communicates $d^k \cdot n \cdot (\log|\mathcal{DS}| + \log|\mathcal{C}|)$ bits within $2k$ rounds.*

Proof. (sketch) **Correctness:** The correctness essentially follows from the structure of the assumed setup and from the properties of the RVHT-PKE scheme. In the case $k = 0$ we have $\mathsf{SK}_i^{(0)} = \overline{\mathsf{SK}}_i^0$ which implies $\mathsf{x}_i = \texttt{ShareDecrypt}(\mathsf{SK}_i^{(0)}, c_i)$. For $k > 0$ we have $\mathsf{SK}_i^{(k)} = \overline{\mathsf{SK}}_i^{(k)} + \sum_{P_j \in \mathbf{N}_G(i)} \mathsf{SK}_j^{(k-1)}$. The properties of the RVHT-PKE scheme thus imply that $\mathsf{x}_i = \texttt{ShareDecrypt}(\mathsf{SK}_i^{(k)}, c_i)$ (c.f. protocol line 9). **Simulation:** In the case $k = 0$ the view of the adversary is directly determined by values given to the simulator. Simulation is therefore easy to achieve. In the case $k > 0$ the simulation of the adversarial view works similar as for the information-transmission protocol (we recall that $\texttt{DerandShare}$ is efficiently invertible if the de-randomizer is known). The simulator essentially emulates the protocol run. The IND-CKCA property of the RVHT-PKE scheme allows the simulator to choose random ciphertexts for $c_{i,\nu_i(j)}$ of honest P_j. Moreover, the decryption shares $\mathsf{x}_{j,\nu_j(i)}$ for honest P_j can also be chosen randomly (where the distribution is conditioned on the outputs of dishonest parties). The view during the executions of $\texttt{DecShares}$ with parameter $k-1$ can be generated using the $(k-1)$-subsimulator guaranteed by the induction hypothesis. **Communication Complexity:** Denote by $f(k)$ be the communication complexity of $\texttt{DecShares}(k, \ldots)$. Then we have $f(0) = 0$ and $f(k) = n \cdot d \cdot (\log|\mathcal{DS}| + \log|\mathcal{C}|) + d \cdot f(k-1)$. This results in a communication complexity of $f(k) = d^k \cdot n \cdot (\log|\mathcal{DS}| + \log|\mathcal{C}|)$. The round complexity follows from the observation that one can invoke the subprotocols $\texttt{DecShares}(k-1, \ldots)$ in parallel.

Definition 12. *A protocol is a* secure (topology-hiding) threshold decryption protocol *for network $\mathcal{N}_{\mathcal{G}}^{d,D}$ if it has the following properties.*

Setup, Inputs and Outputs: *The parties initially hold a setup for topology-hiding threshold encryption over $\mathcal{N}_{\mathcal{G}}^{d,D}$ (*cf. *Definition 8). Each party P_i inputs a ciphertext $c_i \in \mathcal{C}$. The output of party P_i is a message m_i.*

Correctness: *With overwhelming probability it holds for each party P_i that $m_i = \texttt{Combine}(\texttt{ShareDecrypt}(\mathsf{SK}_i, c_i))$ where SK_i is the shared private-key of P_i.*

Topology-Hiding Simulation: *The adversarial view in a real-protocol execution can be simulated with overwhelming probability given the following values*

$$\{\mathbf{N}_G(i), c_i, m_i, \nu_i(\cdot), \overline{\mathsf{SK}}_i^{(0)}, \ldots, \overline{\mathsf{SK}}_i^{(D)}\}_{P_i \in \overline{H}}$$

In other words the simulator gets the neighborhood of dishonest parties (in $\mathcal{N}_{\mathcal{G}}^{d,D}$), their protocol in- and outputs, their local pseudonyms, and their private-key shares of the assumed setup.

Protocol `Decryption`$(c_1, \ldots, c_n)$

```
Require: Parties have access to an initialized $\mathcal{N}_{\mathcal{G}}^{d,D}$ and have generated a setup for topology-hiding
    threshold encryption over $\mathcal{N}_{\mathcal{G}}^{d,D}$.
Input: Each $P_i$ inputs a ciphertext $c_i$.
 1: The parties compute $(\mathsf{x}_1, \ldots, \mathsf{x}_n) = \mathtt{DecShares}(D, c_1, \ldots, c_n)$.
Output: Each $P_i$ outputs $\mathtt{Combine}(\mathsf{x}_i, c_i)$.
```

Lemma 5. *Given a secure RR-MHT-PKE scheme,* `Decryption`$(k, c_1, \ldots, c_n)$ *is a secure threshold decryption protocol. The protocol communicates* $d^D \cdot n \cdot (\log|\mathcal{DS}| + \log|\mathcal{C}|)$ *bits within* $2D$ *rounds.*

Proof. (sketch) The correctness follows directly from Lemma 4 and the properties of the RVHT-PKE scheme. The adversarial view in a real protocol execution can be simulated as follows (recall that `Combine` is efficiently invertible). First the simulator computes for each pair (c_i, m_i) a decryption share x_i such that $m_i = \mathtt{Combine}(\mathsf{x}_i, c_i)$. The rest of the view can then be generated using the sub-simulator for `DecShares`$(D, \ldots)$. The communication complexity and the number of rounds follows directly from the invocation of `DecShares` with parameter D.

3.3 Multi-Party Boolean OR

In this section we present a protocol which securely and topology-hidingly realizes the multi-party Boolean-OR functionality $\mathcal{F}_{\mathtt{OR}}$ using the topology-hiding threshold encryption protocol from the previous section. The functionality $\mathcal{F}_{\mathtt{OR}}$ takes from each party P_i an input bit b_i and computes the OR of those bit, i.e., $b = b_1 \vee \cdots \vee b_n$.

Functionality $\mathcal{F}_{\mathtt{OR}}$

1. Every party P_i (and the adversary on behalf of corrupted parties) sends (input) bit b_i; if P_i does not submit a valid input, then $\mathcal{F}_{\mathtt{OR}}$ adopts $b_i = 0$.
2. Every party P_i receives (output) $b = b_1 \vee \cdots \vee b_n$.

Assumptions We assume in the following that the parties have access to a secure RR-MHT-PKE scheme with security parameter κ, where $n = \text{poly}(\kappa)$. Moreover, parties are given the network $\mathcal{N}_{\mathcal{G}}^{d,D}$ where the graphs in the family $\mathcal{G}$ are connected, have a maximum degree of $d \leq n$, and a diameter of at most $D \leq n$ where d and D are publicly known.

Protocol `Boolean-OR`$(b_1, \ldots, b_n)$

```
Initialization:
 1: Each party $P_i$ inputs its neighborhood $\mathbf{N}_G[i]$ into $\mathcal{N}_{\mathcal{G}}^{d,D}$.
 2: The parties generate a setup for topology-hiding threshold encryption over $\mathcal{N}_{\mathcal{G}}^{d,D}$ using
    GenerateSetup.

Computation:
Input: Each party $P_i$ inputs a bit $b_i$.
 1: Each party $P_i$ sets $m_i = 0$ if $b_i = 0$. Otherwise, its sets $m_i$ to an arbitrary message in $\mathcal{M} \setminus \{0\}$.
 2: The parties compute $(c_1, \ldots, c_n) = \mathtt{InfoTransmisson}\big(D, (m_1, \mathsf{PK}_1), \ldots, (m_n, \mathsf{PK}_n)\big)$.
 3: The parties compute $(m'_1, \ldots, m'_n) = \mathtt{Decryption}(c_1, \ldots, c_n)$.
Output: If $m'_i = 0$ $P_i$ outputs 0. Otherwise it outputs 1.
```

Lemma 6. *Given a secure RR-MHT-PKE scheme and for* d, D *with* $d^D = \text{poly}(\kappa)$ *the protocol* `Boolean-OR`$(b_1, \ldots, b_n)$ *securely and topology-hidingly realizes* $\mathcal{F}_{\mathtt{OR}}$ *(in the* $\mathcal{N}_{\mathcal{G}}^{d,D}$*-hybrid model). In the initialization phase the protocol* `Boolean-OR`$(b_1, \ldots, b_n)$ *communicates* $D \cdot d \cdot n \cdot \log|\mathcal{PK}|$ *bits within* D *rounds. In the computation phase the protocol communicates at most* $(d+1)^D \cdot n \cdot (\log|\mathcal{DS}| + \log|\mathcal{PK}| + 2\log|\mathcal{C}|)$ *bits within* $4 \cdot D$ *rounds.*

Proof. **Correctness:** We assume the condition $d^D = \text{poly}(\kappa)$. The correctness thus follows directly from the properties of Lemmas 2,3, and 5 as the information-transmission protocol essentially allows to compute Boolean-ORs.

Topology-Hiding Simulation: Given the values $\{\mathbf{N}_G(i), b_i, b\}_{P_i \in \overline{H}}$. the view of the adversary can be simulated as follows. First the simulator generates a setup for $\mathcal{N}_{\mathcal{G}}^{d,D}$. Next, for each dishonest P_i the simulator computes the messages m_i and m_i'. It generates the corresponding ciphertext c_i (including the randomness). With those values the simulator now runs the the sub-simulators for `GenerateSetup`, `InfoTransmisson`$(D, \dots)$, and `Decryption`$(\dots)$. The properties of Lemmas 2,3, and 5 ensure that the generated view is indistinguishable (for the adversary) from a real protocol execution.
Communication Complexity: The claimed communication complexity follows directly from the used subprotocols.

Remark 1. If the RR-MHT-PKE is instantiated using the DDH based construction, the computation complexity of the `Boolean-OR` protocol is similar to its communication complexity.

3.4 Topology-hiding Broadcast and Secure Channels

In this section we describe a protocol which securely realizes the (bit) broadcast functionality $\mathcal{F}_{\mathtt{BC}}^s$, while making-black box use of the $\mathcal{F}_{\mathtt{OR}}$ functionality from the previous section. The functionality $\mathcal{F}_{\mathtt{BC}}^s$ allows sender P_s to input a bit b_s which is output to all parties. This result directly implies that one can securely and topology-hidingly realize secure channels and broadcast using black-box RR-MHT-PKE.

Protocol `Broadcast`(P_s, b_s)

```
Require: The sender $P_s$ inputs a bit $b_s$.
 1: The parties compute $(b, \dots, b) = \mathcal{F}_{\mathtt{OR}}(0, \dots, b_s, \dots, 0)$.
Output: Each party $P_i$ outputs $b$.
```

Lemma 7. *The protocol* `Broadcast`(P_s, b_s) *securely realizes the* $\mathcal{F}_{\mathtt{BC}}^s$ *functionality in the* $\mathcal{F}_{\mathtt{OR}}$*-hybrid model.*

Proof. We have that $b = 0 \vee \dots \vee b_s \vee \dots \vee 0 = b_s$ which implies correctness. The view of the adversary in an actual protocol execution consists of inputs and outputs of dishonest parties and is therefore easy to simulate.

Corollary 1. *For* d, D *with* $d^D = \mathrm{poly}(\kappa)$ *one can securely and topology-hidingly realize* $\mathcal{F}_{\mathtt{BC}}^s$ *(in the* $\mathcal{N}_{\mathcal{G}}^{d,D}$*-hybrid model) using black-box RR-MHT-PKE while communicating at most* $(d+1)^D \cdot n \cdot (\log|\mathcal{DS}| + \log|\mathcal{PK}| + 2\log|\mathcal{C}|) + D \cdot d \cdot n \cdot \log|\mathcal{PK}|$ *bits within* $5 \cdot D$ *rounds per invocation.*

Moreover, parties can simply realize secure channels given broadcast. First the receiver generates a key pair and broadcasts the public-key. The sender then broadcasts his message encrypted under this public-key.

Corollary 2. *For* d, D *with* $d^D = \mathrm{poly}(\kappa)$ *one can securely and topology-hidingly realize secure channels (in the* $\mathcal{N}_{\mathcal{G}}^{d,D}$*-hybrid model) using black-box RR-MHT-PKE. The communication complexity is twice the one of the broadcast protocol.*

4 Applications

In this section we provide two applications of our network-hiding communication protocols. Namely, one can securely and topology-hidingly realize MPC and anonymous brodcast.

4.1 Topology-hiding Secure Multi-Party Computation

The protocols from the previous section allow parties to topology-hidingly realize a complete network of secure channels (including broadcast channels). They can then use this network to execute a multi-party protocol of their choice, e.g., [GMW87, Pas04]. This easily proves the following result.

Theorem 2. *For* d, D *with* $d^D = \mathrm{poly}(\kappa)$ *one can securely and topology-hidingly realize any given multiparty functionality (in the* $\mathcal{N}_{\mathcal{G}}^{d,D}$*-hybrid model) using black-box RR-MHT-PKE.*

4.2 Anonymous Broadcast

Theorem 2 implies that one can topology-hidingly realize anonymous channels given black-box access to a RR-MHT-PKE scheme. But using generic MPC to achieve an anonymous channel is expensive in terms of communication complexity. We therefore provide a protocol in the $\mathcal{F}_{\texttt{OR}}$-hybrid model which directly realizes *anonymous broadcast* $\mathcal{F}_{\texttt{ABC}}$.

The functionality $\mathcal{F}_{\texttt{ABC}}$ generates for each party a unique but random pseudonym. In the subsequent communication rounds each party can publish messages under its pseudonym. Message are linkable which means that parties can relate messages to pseudonyms. Parties can prevent this by generating fresh pseudonyms (e.g., after each communication round).

Functionality $\mathcal{F}_{\texttt{ABC}}$

Initialization:

```
1: The functionality generates a random permutation $\sigma$ of $n$ elements.
2: Each party $P_i$ gets output $\sigma(i)$.
```

Communication Step:

Require: Each party P_i inputs a bit b_i.
Output: The parties get the vector $(o_1, \dots, o_n)$ as output where $o_{\sigma(i)} = b_i$.

Anonymous Broadcast Protocol The high-level idea of our construction is as follows. In a scheduling phase each party gets a random (but unique) communication slot $\sigma(i)$ assigned. In a communication round for each slot $\sigma(i)$ the $\mathcal{F}_{\texttt{OR}}$ functionality is invoked which allows P_i to broadcast its bit.

The major challenge is to compute the slot assignment. We solve this issue with a scheduling loop[14]. At the beginning each party selects a random slot. Then over several scheduling rounds the parties resolve colliding selections by computing a reservation matrix. The size of this matrix (parametrized by m) determines the collision detection probability. A larger m means a faster expected run time at the cost of increased communication costs per round.

Protocol $\texttt{AssignSlots}(m)$

```
1: Each party $P_i$ chooses a random slot $s_i \in \{1, \dots, n\}$.
2: repeat
3:     Each party $P_i$ chooses a random token $r_i \in \{1, \dots, m\}$ and computes the $n \times m$-matrix $A^{(i)} = (a^{(i)}_{x,y})$
       where $a^{(i)}_{s_i,r_i} = 1$ and $a^{(i)}_{x,y} = 0$ otherwise.
4:     The parties compute the matrix $A = (a_{x,y})$ where $a_{x,y} = a^{(1)}_{x,y} \vee \dots \vee a^{(n)}_{x,y}$ by invoking $\mathcal{F}_{\texttt{OR}}$.
5:     If there exists an $r < r_i$ such that $a_{s_i,r} = 1$ party $P_i$ chooses a new random slot $s_i \in \{1, \dots, n\}$
       such that $s_i$-th row of $A$ contains only zeros.
6: until Each row of $A$ contains exactly one 1.
Output: Each party $P_i$ outputs $s_i$.
```

Lemma 8. *The protocol* $\texttt{AssignSlots}(m)$ *for the* $\mathcal{F}_{\texttt{OR}}$*-hybrid model securely computes a random permutation σ of n elements where each party P_i learns $\sigma(i)$. The expected number of rounds the protocol requires to compute the permutation is bounded by $\frac{m}{m-1} \cdot n$ where $\mathcal{F}_{\texttt{OR}}$ is invoked $n \cdot m$ times per round.*

Proof. The protocol terminates if each row of A contains exactly one non-zero entry. Thus each slot in $\{1, \dots, n\}$ has been chosen at least by one party. As there are n parties this also means that no slot was chosen twice. The output is therefore a valid permutation. Inspection of the protocol also reveals that the permutation is chosen uniform at random (we consider semi-honest security). Next, we show that the protocol eventually terminates. Each slot is in one of three states. Either its empty, or its selected by multiple parties, or it is assigned to a single party. We observe that the state transition function for slots is monotone. A selected slot cannot become empty and an assigned slot stays assigned to the same party. In each round where a collision is detected at least one empty slot becomes assigned. After at most n such rounds there are no empty slots left. But this also means that each slot is selected by at

[14] A similar idea was used recently in [AKS15].

least one party and the protocol terminates. This also leads to a crude upper bound on the number of expected rounds. We observe that a collision between two parties is detected with a probability of at least $p = (1 - \frac{1}{m})$. The expected number of rounds required to detect a collision is therefore at most $\frac{1}{p} = \frac{m}{m-1}$ (geometric distribution). The number of expected rounds is thus bounded by $\frac{m}{m-1} \cdot n$. It remains to consider the simulation of the adversarial view. We observe that the (current) slot selection of dishonest parties is enough to simulate the view of the adversary in a scheduling round. The simulator can therefore essentially emulate the protocol (conditioned on the final slots of dishonest parties).

Protocol `AnonymousBroadcast`(m)

Initialization:

```
1: The parties compute $(\sigma(1), \ldots, \sigma(n)) = \texttt{AssignSlots}(m)$.
```

Communication Step:

```
Require: Each party $P_i$ inputs a bit $b_i$.
1: for $s = 1, \ldots, n$ do
2:     The parties compute $(o_s, \ldots, o_s) = \texttt{Boolean-OR}(0, \ldots, b_{\sigma^{-1}(s)}, \ldots, 0)$.
3: end for
Output: Each party $P_i$ outputs vector $(o_1, \ldots, o_n)$.
```

Lemma 9. *The protocol* `AnonymousBroadcast`(m) *securely realizes the functionality* $\mathcal{F}_{\texttt{ABC}}$ *in the* $\mathcal{F}_{\texttt{OR}}$*-hybrid model.*

Proof. The statement follows directly from Lemmas 8 and 7.

Corollary 3. *For* d, D *with* $d^D = \text{poly}(\kappa)$ *one can securely and topology-hidingly realize* $\mathcal{F}_{\texttt{ABC}}$ *(in the* $\mathcal{N}_{\mathcal{G}}^{d,D}$*-hybrid model) using black-box RR-MHT-PKE.*

References

[AKS15] Moritz Neikes Anna Krasnova and Peter Schwabe. Footprint scheduling for dining-cryptographer networks. Cryptology ePrint Archive, Report 2015/1213, 2015. `http://eprint.iacr.org/`.

[Bd90] Jurjen N. Bos and Bert den Boer. Detection of disrupters in the DC protocol. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology – EUROCRYPT'89*, volume 434 of *Lecture Notes in Computer Science*, pages 320–327. Springer, April 1990.

[BGT13] Elette Boyle, Shafi Goldwasser, and Stefano Tessaro. Communication locality in secure multi-party computation - how to run sublinear algorithms in a distributed setting. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 356–376. Springer, March 2013.

[Bon98] Dan Boneh. *Algorithmic Number Theory: Third International Symposiun, ANTS-III Portland, Oregon, USA, June 21–25, 1998 Proceedings*, chapter The Decision Diffie-Hellman problem, pages 48–63. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[Can98] Ran Canetti. Security and composition of multi-party cryptographic protocols. Cryptology ePrint Archive, Report 1998/018, 1998. `http://eprint.iacr.org/1998/018`.

[Can00] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `http://eprint.iacr.org/2000/067`.

[CCG$^+$15] Nishanth Chandran, Wutichai Chongchitmate, Juan A. Garay, Shafi Goldwasser, Rafail Ostrovsky, and Vassilis Zikas. The hidden graph model: Communication locality and optimal resiliency with adaptive faults. In Tim Roughgarden, editor, *ITCS 2015: 6th Innovations in Theoretical Computer Science*, pages 153–162. Association for Computing Machinery, January 2015.

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

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981.

[Cha88] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[Cha03] David Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. In Dimitris Gritzalis, editor, *Secure Electronic Voting*, volume 7 of *Advances in Information Security*, pages 211–219. Springer, 2003.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, August 1984.

[GGOR14] Juan A. Garay, Clinton Givens, Rafail Ostrovsky, and Pavel Raykov. Fast and unconditionally secure anonymous channel. In Magnús M. Halldórsson and Shlomi Dolev, editors, *33rd ACM Symposium Annual on Principles of Distributed Computing*, pages 313–321. Association for Computing Machinery, July 2014.

[GJ04] Philippe Golle and Ari Juels. Dining cryptographers revisited. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 456–473. Springer, May 2004.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM Press, May 1987.

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

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques.* Cambridge University Press, 2001.

[HJ07] Markus Hinkelmann and Andreas Jakoby. Communications in unknown networks: Preserving the secret of topology. *Theor. Comput. Sci.*, 384(2-3):184–200, 2007.

[KS10] Valerie King and Jared Saia. Breaking the $o(n^2)$ bit barrier: scalable byzantine agreement with an adaptive adversary. In *Proceedings of the 29th Annual ACM Symposium on Principles of Distributed Computing, PODC 2010, Zurich, Switzerland, July 25-28, 2010*, pages 420–429, 2010.

[KSSV06a] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *SODA*, pages 990–999, 2006.

[KSSV06b] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Towards secure and scalable computation in peer-to-peer networks. In *FOCS*, pages 87–98, 2006.

[MOR15] Tal Moran, Ilan Orlov, and Silas Richelson. Topology-hiding computation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 159–181. Springer, March 2015.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, May 1999.

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In László Babai, editor, *36th Annual ACM Symposium on Theory of Computing*, pages 232–241. ACM Press, June 2004.

[RR98] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, November 1998.

[SGR97] Paul F. Syverson, David M. Goldschlag, and Michael 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. IEEE Computer Society, 1997.

[UKBM11] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the facebook social graph. *CoRR*, abs/1111.4503, 2011.

A Definitions of Public-Key Encryption

This section includes complimentary material to Section 3.1.

Definition 13. *A* public-key encryption *(PKE) scheme with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ consists of three algorithms* $(\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec})$ *where:*

1. *The (probabilistic)* key generation *algorithm* $\mathtt{KeyGen}$ *outputs a public* $\mathsf{pk} \in \mathcal{PK}$ *and a secret key* $\mathsf{sk} \in \mathcal{SK}$*.*
2. *The (probabilistic)* encryption algorithm $\mathtt{Enc}$ *takes a public key* $\mathsf{pk} \in \mathcal{PK}$ *and a message* $m \in \mathcal{M}$ *and outputs a ciphertext* $c \leftarrow \mathtt{Enc}(\mathsf{pk}, m; r)$*.*
3. *The* decryption algorithm $\mathtt{Dec}$ *takes a secret key* $\mathsf{sk} \in \mathcal{SK}$ *and a ciphertext* $c \in \mathcal{C}$ *and outputs message* $m \leftarrow \mathtt{Dec}(\mathsf{sk}, c)$*.*

We require the correctness property: for any key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{KeyGen}$ *and any message* $m \in \mathcal{M}$ *it holds that* $m = \mathtt{Dec}(\mathsf{sk}, \mathtt{Enc}(\mathsf{pk}, m; r))$*.*

Definition 14. *A PKE scheme is* IND-CPA *secure if the adversary has a negligible advantage in winning the following game.*

1. *The game generates a key pair* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{KeyGen}$ *and chooses a random bit b. Then the adversary gets* pk *(this allows him to generate encryptions of arbitrary messages).*
2. *The adversary specifies two messages m_0 and m_1 and the game returns* $c = \mathtt{Enc}(\mathsf{pk}, m_b)$*.*
3. *The adversary specifies a bit b'. If $b = b'$ the adversary has won the game.*

Definition 15. *A* threshold public-key encryption *(T-PKE) scheme with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ consists of four algorithms* $\mathtt{TKeyGen}$*,* $\mathtt{Enc}$*,* $\mathtt{CombineShares}$*,* $\mathtt{Combine}$ *all parameterized by a parameter l (the* threshold*) where:*

1. *The (probabilistic) key-generation algorithm* $\mathtt{TKeyGen}$ *outputs a public* $\mathsf{pk} \in \mathcal{PK}$ *and a secret key* sk *which consists of a vector of sub-keys* $\mathsf{sk}_i \in \mathcal{SK}$*, i.e.,* $\mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_l)$*.*
2. *The (probabilistic)* encryption algorithm $\mathtt{Enc}$ *takes a public key* $\mathsf{pk} \in \mathcal{PK}$ *and a message* $m \in \mathcal{M}$ *and outputs a ciphertext* $c \leftarrow \mathtt{Enc}(\mathsf{pk}, m; r)$*.*
3. *The* decryption-share algorithm $\mathtt{ShareDecrypt}$ *takes a sub-key* $\mathsf{sk}_i \in \mathcal{SK}$ *and a ciphertext* $c \in \mathcal{C}$ *as inputs and outputs a decryption share* $\mathsf{x}_i \leftarrow \mathtt{ShareDecrypt}(\mathsf{sk}_i, c)$*.*
4. *The* combining algorithm $\mathtt{Combine}$ *takes decryption shares* $\mathsf{x}_1, \ldots, \mathsf{x}_l \in \mathcal{DS}$ *and a ciphertext* $c \in \mathcal{C}$ *and outputs a message* $m \leftarrow \mathtt{Combine}(\mathsf{x}_1, \ldots, \mathsf{x}_l, c)$*.*

We require the correctness property: For any key pair $\mathsf{pk}, \mathsf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_l) \leftarrow \mathtt{KeyGen}$ *and any message* $m \in \mathcal{M}$ *it holds that* $m = \mathtt{Combine}(\mathsf{x}_1, \ldots, \mathsf{x}_l, c)$ *where* $\mathsf{x}_i = \mathtt{ShareDecrypt}(\mathsf{sk}_i, c)$ *and* $c = \mathtt{Enc}(\mathsf{pk}, m; r)$*.*

B Multi-Homomorphic Threshold Encryption with Reversible Randomization based on DDH

In this section we present a secure RR-MHT-PKE scheme based on the DDH assumption. Our construction can be seen as an extended variant of the ElGamal cryptosystem [ElG84].

B.1 MHT-PKE-Algorithms

Let $\langle G, \cdot \rangle$ be a cyclic group of prime order $q(\kappa)$ for security parameter κ. Denote by e_G the neutral element of G and let g be a generator of G. For our scheme we assume that (G, q, g) is publicly known. To ensure the various homomorphic properties we use G as message space, public key space, decryption share space, i.e., $\mathcal{M} = \mathcal{PK} = \mathcal{DS} = G$. Ciphertexts consist of two group elements, i.e., $\mathcal{C} = G \times G$. Moreover, the private-key space is $\mathcal{SK} = \langle \{0, \ldots, p-1\}, + \rangle$. The core part of our construction are the following four MHT-PKE algorithms.

Key Generation: The key-generation algorithm $\mathtt{KeyGen}$ chooses a private-key sk uniformly at random from $\mathcal{SK}$ and sets the public-key $\mathsf{pk} = g^{\mathsf{sk}}$, i.e.,

$$(g^{\mathsf{sk}}, \mathsf{sk}) \twoheadleftarrow \mathtt{KeyGen}().$$

Encryption: The encryption algorithm $\mathtt{Enc}$ encrypts message m under public-key pk as $c := (g^r, \mathsf{pk}^r \cdot m)$ where $r \in \{1, \ldots, p-1\}$ is chosen uniformly at random.

$$\mathtt{Enc}(\mathsf{pk}, m; r) = (g^r, \mathsf{pk}^r \cdot m)$$

Decryption Shares: The decryption share algorithm `ShareDecrypt` takes a ciphertex $c = (a, b)$ and a private-key sk and computes decryption share $\mathsf{x} = a^{-\mathsf{sk}}$, i.e.,

$$\texttt{ShareDecrypt}(\mathsf{sk}, c = (a, b)) = a^{-\mathsf{sk}}.$$

Combine: The combining algorithm `Combine` takes a ciphertext $c = (a, b)$ and a decryption shares x and computes message $m = \mathsf{x} \cdot b$, i.e.,

$$\texttt{Combine}(\mathsf{x}, c = (a, b)) = \mathsf{x} \cdot b.$$

Note also that $\mathsf{x} = m \cdot b^{-1}$, i.e., `Combine` is efficiently invertible.

It is easy to show that those four algorithms satisfy the correctness property required by Definition 2. In the following we assume that decisional Diffie-Hellman (DDH) assumption holds for G. This means it is computationally hard to distinguish for given group elements (α, β, γ) whether they are independent uniform random in G or whether $\alpha = g^a$ and $\beta = g^b$ are independent uniform random and $\gamma = g^{ab}$. A simple choice for G is a Schnorr Group, which is a q-order subgroup of $\mathbb{Z}_p^\times$ where p, q are primes with $p = qr + 1$ for some r. A more efficient and therefore preferred alternative is to use an appropriate elliptic curve group[Bon98].

Lemma 10. *If DDH holds for G, the MHT-PKE scheme is IND-TCPA secure.*

Proof. We give a reduction of DDH to IND-TCPA. Consider a winner $\widetilde{W}$ for the IND-TCPA game (c.f. Definition 3). Then the following reduction gives a winner W for DDH.

Reduction DDH to IND-TCPA

1. On input of (α, β, γ) from the outside set $\mathsf{pk} := \alpha$, choose a random bit b, generate key pairs $(\mathsf{pk}_2, \mathsf{sk}_2), \ldots, (\mathsf{pk}_l, \mathsf{sk}_l) \twoheadleftarrow \texttt{KeyGen}$, and set $\mathsf{pk}_1 := \mathsf{pk} \cdot g^{-\mathsf{sk}_2} \cdot \ \ldots\ \cdot g^{-\mathsf{sk}_l}$. Then send pk, $\mathsf{pk}_1, \ldots, \mathsf{pk}_l$ and $\mathsf{sk}_2, \ldots, \mathsf{sk}_l$ to the winner $\widetilde{W}$.
2. Given the messages m_0 and m_1 from $\widetilde{W}$ return $(\beta, \gamma \cdot m_b)$ to $\widetilde{W}$.
3. If $\widetilde{W}$ guesses b correctly output 1, else output 0.

If the input (α, β, γ) is of the form (g^a, g^b, g^{ab}) the reduction returns in the second step $(g^b, g^{ab} \cdot m_b) = (g^b, \mathsf{pk}^b \cdot m_b)$ to $\widetilde{W}$ which is a valid encryption of m_b under $\mathsf{pk} := \alpha$. The winner $\widetilde{W}$ should therefore be able to distinguish between $(\beta, \gamma \cdot m_0)$ and $(\beta, \gamma \cdot m_1)$ with non-negligible advantage. If γ is uniform at random the tuple $(\beta, \gamma \cdot m_b)$ is independent of bit b. The winner $\widetilde{W}$ should therefore have a negligible advantage in guessing the bit correctly.

Lemma 11. *The MHT-PKE scheme is IND-CKA secure*

Proof. For any public key pair $(\mathsf{pk}_2, \mathsf{pk})$ there exists a public-key pk_1 such that $\mathsf{pk}_2 = \mathsf{pk}_1 \cdot \mathsf{pk}$. The key pair $(\mathsf{pk}_2, \mathsf{pk})$ is thus distributed independently of b. The adversary thus has a negligible advantage in guessing the b correctly.

B.2 Reversible-Randomization Algorithms

The final part of our construction are the four (de)randomization algorithms which complete the RR-MHT-PKE scheme. The used de-randomizer spaces are $\mathcal{RK}_P = \mathcal{RK}_C = \{0, \ldots, p-1\}$.

Public-Key Randomization: To randomize a public-key pk the key randomization algorithm `RandKey` generates a fresh key pair $(g^{\mathsf{rk}}, \mathsf{rk}) \twoheadleftarrow \texttt{KeyGen}$ and multiplies pk with g^{rk}. The private rk acts as the de-randomizer.

$$(\widetilde{\mathsf{pk}}, \mathsf{rk}) = (g^{\mathsf{rk}} \cdot \mathsf{pk}, \mathsf{rk}) \twoheadleftarrow \texttt{RandKey}(\mathsf{pk})$$

Ciphertext De-Randomization: To de-randomize a given ciphertext $\widetilde{c}$ the de-randomization algorithm `DerandCipher` essentially computes a decryption share for rk and strips it from $\widetilde{c}$.

$$\texttt{DerandCipher}\left(\mathsf{rk}, \widetilde{c} = (a, b)\right) = (a, a^{-\mathsf{rk}} \cdot b)$$

We observe that `DerandCipher` is efficiently invertible.

Ciphertext Randomization: To randomize a ciphertext $c = (a, b)$ the algorithm `RandCipher` chooses a random $r, r' \in G$, and a random $d \in \{1, \ldots, p-1\}$ and computes[15] $e \in \{1, \ldots, p-1\}$ with $e \cdot d \equiv_p 1$. If $a \neq e_G$ it exponentiates a with e, otherwise it replaces a by r.

$$(\hat{c}, \mathsf{rk}) = \begin{cases} \big((a^e, r'), d\big) & \text{if } a \neq e_G \\ \big((r\ , r'), 0\big) & \text{if } a = e_G \end{cases} \twoheadleftarrow \mathtt{RandCipher}\,\big(c = (a, b)\big)$$

Decryption-Share De-Randomization: To de-randomize a given decryption share $\hat{\mathsf{x}}$ the algorithm computes $\hat{\mathsf{x}}^{\mathsf{rk}}$.

$$\mathtt{DerandShare}(\mathsf{rk}, \hat{\mathsf{x}}) = \hat{\mathsf{x}}^{\mathsf{rk}}$$

We observe that `DerandShare` is efficiently invertible.

We can now show that the above algorithms satisfy the correctness properties of Definition 5. For a public key PK let $(\widetilde{\mathsf{pk}, \mathsf{rk}}) \twoheadleftarrow \mathtt{RandKey}(\mathsf{pk})$. For any message m consider the encryption $\tilde{c} = \mathtt{Enc}(\widetilde{\mathsf{pk}}, m; r)$ of m under public-key $\widetilde{\mathsf{pk}}$ with randomness r. Then we have

$$\begin{aligned} \mathtt{DerandCipher}(\mathsf{rk}, \tilde{c}) &= (g^r, (g^r)^{-\mathsf{rk}} \cdot g^{\mathsf{rk} \cdot r} \cdot \mathsf{pk}^r \cdot m) \\ &= (g^r, \mathsf{pk}^r \cdot m) = \mathtt{Enc}(\mathsf{pk}, m; r). \end{aligned}$$

Next, for a ciphertext $c = (a, b)$ let $\big(\widehat{c} = (\widehat{a}, \widehat{b}), \mathsf{rk}\big) \twoheadleftarrow \mathtt{RandCipher}(c)$. For any private-key sk consider the decryption-share $\hat{\mathsf{x}} = \mathtt{ShareDecrypt}\,\big(\mathsf{sk}, (\widehat{c})\big)$. Then we have

$$\begin{aligned} \mathtt{DerandShare}(\mathsf{rk}, \widehat{\mathsf{x}}) &= \widehat{\mathsf{x}}^{\mathsf{rk}} = \widehat{a}^{-\mathsf{sk} \cdot \mathsf{rk}} \\ &= \begin{cases} a^{-\mathsf{sk} \cdot e \cdot d} = a^{-\mathsf{sk}} = \mathtt{ShareDecrypt}(\mathsf{sk}, c) & \text{if } a \neq e_G \\ r^{-\mathsf{sk} \cdot 0} = e_G = e_G^{-\mathsf{sk}} = \mathtt{ShareDecrypt}(\mathsf{sk}, c) & \text{if } a = e_G. \end{cases} \end{aligned}$$

The correctness properties are thus fulfilled. It remains to show that our construction satisfies the IND-CKCA security property.

Lemma 12. *The RR-MHT-PKE scheme is IND-CKCA secure*

Proof. For any public key pair $(\widetilde{\mathsf{pk}}, \mathsf{pk})$ there exists a $\mathsf{rk} \in \mathcal{RK}_P$ such that $\widetilde{\mathsf{pk}} = g^{\mathsf{rk}} \cdot \mathsf{pk}$. The key pair $(\widetilde{\mathsf{pk}}, \mathsf{pk})$ is thus distributed interdependently from b_1. The adversary thus has a negligible advantage in guessing the b_1 correctly. Consider ciphertext pair $\big(\widehat{c} = (\widehat{a}, \widehat{b}), c = (a, b)\big)$. As long as $a = e_G$ or $\widehat{a} \neq e_G$ there exist r, r', d such that $(\hat{c}, \mathsf{rk}) = \mathtt{RandCipher}(c = (a, b); r, r', d)$. In this case the adversary thus has a negligible advantage in guessing the b_2 correctly. If $a = e_G$ and $\widehat{a} = e_G$ the ciphertext $\widehat{c}$ cannot be the randomization of c and guessing b_2 is easy. However, the probability that $a = e_G$ and $\widehat{a} = e_G$ is negligible for $q(\kappa)$ large enough.

Theorem 3. *The above scheme is a secure RR-MHT-PKE scheme*

Proof. Follows directly from Definition 7 and Lemmas 10,11, and 12.

[15] This can be done efficiently using the Extended Euclidean algorithm.