Efficient Secure Communication Over Dynamic Incomplete Networks With Minimal Connectivity

Ivan Damgård[1], Divya Ravi[2][0000−0001−6423−8331], Lawrence Roy[1], Daniel Tschudi[3][0000−0001−6188−1049], and Sophia Yakoubov[1][0000−0001−7958−8537] ⋆

[1] Aarhus University, Denmark; {ivan, lance.roy, sophia.yakoubov}@cs.au.dk
[2] University of Amsterdam, Netherlands;
[3] Concordium, Zurich, Switzerland;

Abstract. We study the problem of implementing unconditionally secure reliable and private communication (and hence secure computation) in dynamic incomplete networks. Our model assumes that the network is always k-connected, for some k, but the concrete connection graph is adversarially chosen in each round of interaction. We show that, with n players and t malicious corruptions, perfectly secure communication is possible if and only if $k > 2t$. This disproves a conjecture from earlier work, that $k > 3t$ is necessary. Our new protocols are much more efficient than previous work; in particular, we improve the round and communication complexity by an exponential factor (in n) in both the semi-honest and the malicious corruption setting, leading to protocols with polynomial complexity.

⋆ Funded in part by the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 669255 (MPCPRO) and No 803096 (SPEC), the Danish Independent Research Council (grant DFF-0165-00107B "C3PO"), and the DARPA SIEVE program (contract HR001120C0085 "FROMAGER"). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited). We thank the authors of [DRTY23] for the source code of the tables and protocol boxes.

Table of Contents

1 Introduction

This paper studies unconditionally secure communication (and by extension multiparty computation) when parties communicate over an incomplete and dynamic network. More specifically, we assume synchronous communication with secure point-to-point channels; however, only some of the point-to-point connections actually exist, so the network is incomplete. Moreover, the set of active connections can change from one round to the next, and the graph describing the active connections is adversarially chosen in each round. This is called a *dynamic* incomplete network, in contrast to a *static* incomplete network, where the active connections stay the same throughout the protocol.

Static incomplete networks were first studied by Dolev [Dol82]. Here, it was shown that when t of the n parties are malicious, one can do secure broadcast if and only if the network is at least $2t + 1$-connected, and $3t < n$ [4]. Later, Dolev et al. [DDWY93] showed that, in a static incomplete network, one can communicate privately and reliably if and only if the network is $2t + 1$-connected. Using the protocols from that work, one can emulate a complete network with secure point-to-point channels. This can be combined with any MPC protocol that is based on a complete network, and one can then conclude that, in a static incomplete network, $2t + 1$-connectivity is necessary and sufficient for unconditionally secure MPC to be possible.

The case of dynamic incomplete networks was first considered in Maurer *et al.* [MTD15], who studied reliable (non-private) communication. They defined a notion called *dynamic min-cut* which is a number that can be derived from the entire sequence of networks graphs. They then showed that reliable communication is possible between any pair of parties if and only if the dynamic min-cut is larger than $2t$. Although this is the weakest possible condition that allows for reliable communication, it makes a rather complicated statement on the entire sequence of network graphs, and in a given practical setting it would be hard to assess whether it is going to

[4] A graph is k-connected if any pair of distinct nodes are connected by at least k disjoint paths, or equivalently, if it remains connected when one removes any set of less than k vertices.

be satisfied or not. Also, from a theoretical standpoint, it is natural to ask if there is a condition on each *individual* network graph that would enable both reliable and private communication.

These questions were considered in [DRTY23]. They initiated the study of (unconditionally) private communication in dynamic networks, and introduced a model that is a natural extension of the static case, where it is required that in each round, the network graph for that round is at least k-connected (and this is the only condition assumed). The model further assumes that that honest parties do not know the network topology. This is reasonable, as connections may be down because mobile devices move, or equipment crashes, and such events cannot be predicted locally. Finally, it is assumed that at most t parties are corrupted by an adversary, either passively (semi-honest) or actively (malicious).

The main results from [DRTY23] are as follows: reliable communication is possible in a dynamic network if and only if $k > 0$ in the passive case and $k > 2t$ in the active case. Private communication can be done for a passive adversary, if and only if $k > t$. For an active adversary, $k > 3t$ is sufficient for perfectly secure private communication, whereas $k > 2t$ is necessary (which follows from known results in the static case). It was conjectured that in fact $k > 3t$ is necessary for perfect security (whereas they showed that $k > 2t$ is sufficient for statistical security).

The protocols from [DRTY23] (and [MTD15]) introduce quite a large performance penalty compared to the static case. They work by trying many different paths from sender to receiver in the hope that the message will eventually arrive along enough disjoint paths. For private communication, key material is sent along many different paths in the hope that some of it will make it to the sender via paths consisting of only honest players, which allows extraction of a key that is completely unknown to the adversary. Unfortunately, the upper bounds shown for both round and communication complexity of these approaches were exponential in n in the worst case.

1.1 Our Contributions

In this paper, we disprove the conjecture from [DRTY23] and show that in fact $k > 2t$ is sufficient for perfect private communication. Since this condition was already known to be necessary, this completes the characterisation of dynamic incomplete networks allowing for private and reliable communication. We also give new protocols for private communication in the passive case, and for reliable and private communication in the active case. The protocols are based on several new techniques, allowing us to remove the exponential dependency on n in the protocols from [DRTY23]. Indeed, all our protocols have complexity polynomial in n for all values of n, k and t for which secure communication is possible.

We stick to the network model from [DRTY23] for most of the paper, as it allows for simple descriptions of protocols and clean statements about their properties. But we emphasize that our protocols do not need to assume that the network graph is k-connected in every round, and in fact they work under much weaker assumptions on connectivity. We discuss this in more detail in Section 6.

The complexities of our constructions are summarized in Tables 1 and 2.

1.2 Technical Overview

Reliable Communication Reliable communication with a passive adversary can be done as long as $k > 0$. A simple flooding approach will work in n rounds, as was already noted in [DRTY23]. For reliable communication in the active adversary case we introduce a new protocol and proof. In a nutshell, we also use a flooding approach here, but each copy of the message carries metadata, namely a graph describing the paths the message travelled along

Table 1: Reliable Communication Protocols for a message m, over k-connected networks. The communication complexity is the total communication of honest parties in the protocol. M denotes the total bits of all messages sent by corrupt parties (which assuming PPT adversaries remains polynomial).

	Corruption		Graph	Complexity	
Scheme	Type	Threshold	k	Rounds	Communication
[DRTY23], Protocol 1	passive	$t < n$	$k \geq 1$	n	$\mathcal{O}\big(n^3\lvert m\rvert\big)$
[DRTY23], Protocol 2	active	$t < \frac{n}{2}$	$k > 2t$	$\mathcal{O}(n2^n)$	$\mathcal{O}\big(n^4 2^{2n}\lvert m\rvert\big)$
This work, Protocol 3	active	$t < \frac{n}{2}$	$k > 2t$	$\leq n$	$\mathcal{O}\big(n^3\lvert m\rvert + (M+1)n^6 \log_2(n)\big)$

Table 2: Private Communication Protocols for a message m, over k-connected networks with corruption threshold t. The complexity of Protocol 4 is in terms of the complexities of its building blocks, namely reliable communication and secure message transmission; where $\rho_{\texttt{Rel}}$, $\rho_{\texttt{SMT}}$ denote the round complexities and $c_{\texttt{Rel}}(x)$, $c_{\texttt{SMT}}(x)$ denote the communication complexities for transmitting x bits. There exist instantiations of these building blocks that maintain these complexities to be polynomial (elaborated in the relevant technical section). Note that the protocols in this paper are the first to offer round and communication complexity which is sub-exponential in the number of parties.

		Corruption		Graph		Complexity	
Scheme	Security	Type	t	k	directed	Rounds	Communication
[DRTY23], Protocol 3	perfect	passive	$t < n$	$k > t$	✓	$\mathcal{O}(n2^n)$	$\mathcal{O}\big(2^{2n}n^3(n+\lvert m\rvert)\big)$
[DRTY23], Protocol 5	perfect	active	$t < \frac{n}{3}$	$k > 3t$	✓	$\mathcal{O}(n2^n)$	$\mathcal{O}\Big(2^{n^2+n}n^3(n+\lvert m\rvert)\Big)$
[DRTY23], Protocol 6	perfect	active	$t < \frac{n}{4}$	$k > 4t$	✓	$\mathcal{O}(n2^n)$	$\mathcal{O}\big(2^{3n}n^5(n+\lvert m\rvert)\big)$
This work, Protocol 1, Corollary 3	perfect	passive	$t < n$	$k > t$	✓	$2\lfloor\frac{n}{k}\rfloor + 3$	$\mathcal{O}\Big(\frac{n^4}{k}(n\log(n)+\lvert m\rvert)\Big)$
This work, Protocol 2, Corollary 6	perfect	passive	$t < n$	$k > t$	✗	$\lfloor\frac{n}{k}\rfloor + 2$	$\mathcal{O}\Big(\frac{n^4\lvert m\rvert}{k}\Big)$
This work, Protocol 4	perfect	active	$t < \frac{n}{3}$	$k > 2t$	✓	$1 + 2\rho_{\texttt{Rel}} + \rho_{\texttt{SMT}}\rho_{\texttt{Rel}}$	$\mathcal{O}\big(nc_{\texttt{Rel}}(n^2) + nc_{\texttt{SMT}}(\lvert m\rvert)c_{\texttt{Rel}}(n\log(n))\big)$

to reach the current party. The party assembles new metadata based on what it received, and passes it along in the next round. After some number of rounds, the final receiver R has received possibly several different messages and metadata (as the adversary is free to fabricate incorrect messages and metadata). We show that after $\mathcal{O}(n)$ rounds, R has enough data to decide with probability 1 what the correct message is.

An overview of how we arrive at this result: Earlier work for dynamic networks tracked all paths on which a message has travelled, and the receiver would believe a message if it arrives on more than t disjoint paths, as this implies it must have travelled on a path with only honest players. Unfortunately, the number of potential paths can be exponential, leading to inefficient protocols. An obvious idea for improvement is to collect the paths on which the message travelled into a graph that can hopefully be described more compactly than the set of paths. In case of a static network, it is quite straightforward to see that this will work: the receiver can decide whether to believe a certain message m based on the max-flow or equivalently min-cut of the graph that is sent along with m (considering cuts separating sender from receiver). If this number is larger that t, the sender believes the message.

However, things get more complicated in the dynamic case. Min-cut is not defined for a dynamic network, so we construct a different type of graph that captures the history of a message travelling through the network. There is a node for each pair (P_i, j), where P_i is one of the parties and j indicates a round in the protocol. We put an edge from $(\mathsf{P}_i, j-1)$ to $(\mathsf{P}_{i'}, j)$ if P_i sent the message to $\mathsf{P}_{i'}$ in round j (and by default a party is connected to itself from each round to the next) [5]. We then introduce a notion called relaxed labelled min-cut in this type of graph, where we allow fractions as values, in contrast to the standard notion of min-cut which is always an integer. This allows us to formulate an efficiently computable predicate which we show is satisfied after n rounds by the graph traveling with the correct message, and cannot be satisfied for any incorrect message.

Private Communication For private communication, our main idea is as follows: instead of trying to send the message via a large number of paths, as in [DRTY23], we ask each party to choose a secret key for each other party and try to send, in a single round, each key to the party it was chosen for. Then we determine via public discussion which connections worked, using the fact that we already know how to do reliable (non-private) communication efficiently. Intuitively, this "freezes" the network graph G that existed in the round where the keys were sent.

Skipping a few details, the keys exchanged can be used to send data privately along a path in G, such that the adversary will have no information on what is sent if the path contains only honest players. This means that in the passive case, the sender can secret-share its message additively and send shares privately to the receiver along a set of disjoint paths in G. This will work because G is sufficiently connected so that at least one share will remain unknown to the adversary.

In the active case, we assume the G is at least $2t+1$-connected. We can think of the (at least) $2t+1$ disjoint paths from sender to receiver as channels connecting sender to receiver, of which at most t can be corrupted. We can then use known efficient protocols for maliciously secure perfect message transmission to send a private message.

[5] For technical reasons, we even need to have several nodes for one party in each round. As we explain later, this has to do with the fact that a corrupt party can claim they heard a fake message from an honest party P_h, but at the same time P_h might report the same fake message because he heard from another corrupt party. These two "stories" must be treated separately and we do this by having two different nodes for P_h in the relevant round.

This disproves the conjecture from [DRTY23], that connectivity must be at least $3t + 1$. In [DRTY23] evidence was given for the conjecture by arguing that any protocol in the class of solutions they considered would fail for connectivity less than $2t + 1$. As also noted there, this is of course not a proof, and indeed our protocol falls outside the class because it crucially uses public discussion to decide on the paths to use later. This option was not considered in [DRTY23].

2 Preliminaries

In this work we consider the setting as defined in [DRTY23]. Let $\mathbb{F}$ be a finite field. Let $\mathcal{P} = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$ denote the set of n parties. The *sender* $\mathsf{S} \in \mathcal{P}$ wants to send a message $m \in \mathbb{F}$ to *receiver* $\mathsf{R} \in \mathcal{P}$.

2.1 Adversary Model

A computationally-unbounded, central adversary corrupts at most t parties. The party corruption is *static*, i.e., the adversary is required to select the set of corrupted parties before the protocol execution. We distinguish between *passive* corruption where the adversary can access the internal state of corrupted parties and *active* corruption where the adversary additionally has full control over the behavior of corrupted parties.

2.2 Communication Network

Parties communicate over a *dynamic incomplete network* of secure (private and authentic) synchronous channels in the presence of a *rushing adaptive* network adversary. In more details, the protocol proceeds in rounds, also known as time steps. Let $\mathcal{G}$ be a publicly-known family of graphs over $\mathcal{P}$. Intuitively, the graph family $\mathcal{G}$ models the network guarantees for honest parties. For example, $\mathcal{G}$ could be the family of connected graphs. In each round a party *may attempt* to use any channel in their neighborhood of $\bar{\mathsf{G}} = \bigcup_{\mathsf{G} \in \mathcal{G}} \mathsf{G}$[6]. The adversary decides on the actual communication graph $\mathsf{G}_r \in \mathcal{G}$ for round r after having observed the communication attempts. Any message input on a channel within G_r will then be delivered by the end of the round. We note that honest parties are *oblivious* of the actual communication graph G_r. In particular, honest parties do not learn which of their outgoing transmissions were successful. If not specified otherwise, the graphs in $\mathcal{G}$ may be directed graphs.

2.3 Security

In this work we consider protocols that achieve a communication channel with *perfect security* between sender and receiver.

Reliable Communication. A reliable communication channel allows a sender S to send a message to a receiver R in a tamper-resilient manner.

Definition 1 (Reliable Communication). *A protocol achieves a* reliable *communication channel with perfect security between* S *and* R *in the presence of an adversary* $\mathcal{A}$ *if the following holds:*

> **Correctness** *The output message* m_{R} *of* R *is the input message* m *of* S*, i.e.* $\Pr[m_{\mathsf{R}} \neq m] = 0$ *where the randomness is over the coins of all honest parties and* $\mathcal{A}$*.*

[6] $\bar{\mathsf{G}}$ is often the fully-connected graph, so every party's neighborhood is then all of $\mathcal{P}$.

Private Communication. A private communication channel allows a sender S to send a message to a receiver R in a reliable and private manner, i.e. the adversary will not learn any information on the message.

Definition 2 (Private Communication). *A protocol achieves a* private *communication channel with perfect security in the presence of an adversary $\mathcal{A}$ if it achieves a* reliable *communication channel, and additionally the following holds:*

> **Privacy** *The view of adversary $\mathcal{A}$ can be simulated from the inputs and outputs of corrupted parties. In particular, for honest S, R the adversarial view is independent of the sender's input message m.*

2.4 Graphs

In this section we define the graph properties we consider for the communication network in our protocols.

Definition 3. *A (directed) graph is* (u, v)-k-connected *if for nodes u, v there exist k disjoint (directed) paths from u to v.*

Definition 4. *A (directed) graph is* k-connected *if it is (u, v)-k-connected for any pair (u, v) of nodes.*

A 1-connected graph is simply called *connected*. We say graph family $\mathcal{G}$ has property X if every graph in $\mathcal{G}$ has property X.

3 Private Communication with Passive Corruption

Damgård et al. [DRTY23] presented a private communication protocol that tolerates $t < n$ passive corruptions in a dynamic network with connectivity $k > t$. Both the round and communication complexity of this protocol scales linearly with the cardinality of a given set of paths Paths between sender and receiver. If Paths happens to be the set of all possible paths, its cardinality is exponentially large in n. Motivated by the goal of improving the efficiency of this protocol, we propose a private communication protocol in this setting where the round and communication complexity is polynomial in n.

Reliable multicast as building block. Before describing our protocol, we note that in the passive setting reliable multi-cast is easy to achieve using a flooding approach if $\mathcal{G}$ is connected. All parties that have seen the sender's message will repeatedly attempt to send it to all their neighbors. We will use this multi-cast primitive in our protocol for secure communication.

Lemma 1 (From [DRTY23]). *For any connected $\mathcal{G}$ there exists a protocol* $\texttt{MultiCast}(\mathsf{P}, m)$ *that allows party P to safely distribute m in the presence of a passive adversary. The protocol runs for n rounds and has a total communication complexity of $\mathcal{O}(n^3|m|)$ bits, where $|m|$ denotes the number of message bits.*

Remark 1. The flooding protocol in [DRTY23] is used to construct a reliable communication channel from the sender to a specific receiver. However, we note that the flooding approach guarantees that at the end of the protocol every party will have the sender's message. So by modifying the protocol slightly, such that every party outputs the received message, the construction directly achieves multi-cast.

3.1 Private Communication on Directed Graphs

Assume that directed $\mathcal{G}$ has connectivity $k > t$, i.e. every graph in $\mathcal{G}$ has connectivity $k > t$. The main idea of Protocol 1 is as follows: In the first round, parties attempt to send a random field element to each of their potential neighbors. Then they each use `MultiCast` to announce the set of nodes from whom they actually received randomness. This publicly defines a (undirected) 'meta graph', say $\mathbb{G}$, whose edges correspond to pairs of nodes who now have shared randomness. Essentially, this serves as a means to freeze[7] the first graph chosen by the dynamic adversary. Now, the problem of private communication becomes much simpler as we can focus on this meta graph $\mathbb{G}$ which is guaranteed to have connectivity $k > t$. The sender S splits their message into $t+1$ sum shares which are now passed along $t+1$ disjoint paths in $\mathbb{G}$ between S and R. Communication along the edges in $\mathbb{G}$ is emulated using `MultiCast` and the shared randomness is used to encrypt messages.

Comparison with previous work. We point out that the idea of additively secret sharing the secret among a set of $t+1$ disjoint paths is similar to the protocol of Damgård et al. The crucial difference is that we depend on a *fixed set* of disjoint paths in the meta graph while their protocol considered *all possible sets* of disjoint paths over dynamic graphs. This allows us to achieve complexity that is polynomial in n.

Theorem 1. *Protocol 1 is a private communication protocol that achieves perfect security against $t < n$ passive corruptions for network $\mathcal{G}$ with connectivity $k > t$. The protocol communicates at most $n^2|m|$ bits over network $\mathcal{G}$, and sends at most $\mathcal{O}\big(n^2 \log(n) + n|m|\big)$ bits over multi-cast channels.*

Proof. We start with an observation on $\mathbb{G}$. Multi-casting the sets In ensures that parties agree on the meta-graph $\mathbb{G}$. The connectivity $k > t$ of the first round graph ensures that there are at least $t+1$ disjoint paths from S to R in $\mathbb{G}$. We also note that due to Lemma 1 multi-casting is possible as $\mathcal{G}$ is connected.

Correctness. There are $t+1$ disjoint paths from S to R in $\mathbb{G}$, thus $\mathsf{GoodPaths}$ exists. The agreement on $\mathbb{G}$ implies agreement on $\mathsf{GoodPaths}$. This ensures that parties agree on all necessary invocations of multi-cast in the second phase. For each path p it holds that

$$s'_p = \left(\sum_{p_i \in \mathcal{N}_p \setminus \{\mathsf{R}\}} m_{p,p_i} \right) - o_{p_{\ell-1},p_\ell} = s_p + o_{p_1,p_2} + \left(\sum_{p_i \in \mathcal{N}_p \setminus \{\mathsf{R},\mathsf{S}\}} o_{p_i,p_{i+1}} - o_{p_{i-1},p_i} \right) - o_{p_{\ell-1},p_\ell} = s_p.$$

This means the receiver will compute the right shares from the multi-cast messages and will output the sender's message m.

Privacy. The view of the adversary can be simulated from the output of an ideal secure channel between S and R. If sender or receiver are corrupted, the simulator knows the message m, and the selected graphs. This allows simulation of the adversarial view by executing the real protocol in the head. If neither the sender nor receiver are corrupted the view of the adversary can be simulated by executing the protocol in the head with $m = 0$ (or any other default value). The values in the first phase will be distributed exactly as in the real world. In the second phase there exists a path p in $\mathsf{GoodPaths}$ that consists of honest parties. The multi-cast messages for this path are, without knowledge of the involved shared randomness, distributed independent of the actual message. So the simulated view is indistinguishable from the real protocol. In particular, the adversary does not learn s_p. On the other paths the adversary may learn the shares. However, these shares are uniform random and independent of the message (if one does

[7] Technically, $\mathbb{G}$ is an undirected graph with the same node set as the first graph. The edge (u, v) is in $\mathbb{G}$ iff (u, v) or (v, u) are in the first graph.

Protocol 1: $\Pi^{\mathsf{prv}}_{\mathsf{perf},\mathsf{sh}}(\mathsf{S},\mathsf{R},m)$

The sender S and receiver R are public input. The sender S has message m as private input. Let $\bar{\mathsf{G}} = \bigcup_{\mathsf{G}\in\mathcal{G}} \mathsf{G}$.

Establishing meta graph and shared randomness. Party P_i does the following:

- For each neighbor P_j in $\bar{\mathsf{G}}$, sample uniform random $r_{i,j}$ (of size $|m|$).
- In round 1, attempt to send $r_{i,j}$ to P_j using the communication network.
- Let In_i denote the set of parties P_j from whom P_i actually received randomness $r_{j,i}$ in round 1.
- Parties jointly invoke $\texttt{MultiCast}(\mathsf{P}_i, \mathsf{In}_i)$ where In_i is encoded as an n-bit vector.
- Build the meta graph $\mathbb{G}$ as follows: there is an edge from P_u to P_v if $\mathsf{P}_u \in \mathsf{In}_v$ or $\mathsf{P}_v \in \mathsf{In}_u$.
- For each neighbor P_j in $\mathbb{G}$ set $o_{i,j} \coloneqq r_{j,i} + r_{i,j}$. (Missing values $r_{\cdot,\cdot}$ are set to 0; that is, if $\mathsf{P}_j \notin \mathsf{In}_i$, $o_{i,j} = r_{j,i}$, and if $\mathsf{P}_i \notin \mathsf{In}_j$, $o_{i,j} = r_{i,j}$.)

Secure message transfer in $\mathbb{G}$. All the parties can now locally determine the same set of $t+1$ disjoint paths, say $\mathsf{GoodPaths}$, in $\mathbb{G}$ between S and R (using Ford-Fulkerson algorithm) [a]. Then message transfer is done as follows:

- The sender S selects $t+1$ uniform random shares $\{s_p\}_{p\in\mathsf{GoodPaths}}$, such that $m = \sum_{p\in\mathsf{GoodPaths}} s_p$.
- The parties $\mathcal{N}_p = \{\mathsf{P}_{p_1}, \ldots, \mathsf{P}_{p_\ell}\}$ on each path p do the following:
  - The sender $\mathsf{S} = \mathsf{P}_{p_1}$ computes $m_{p,p_1} = s_p + o_{p_1,p_2}$ and all parties jointly invoke $\texttt{MultiCast}(\mathsf{S}, (p, m_{p,p_1}))$ where p is encoded in $n\log(n)$ bits.
  - Each party $\mathsf{P}_{p_i} \in \mathcal{N}_p \setminus \{\mathsf{S},\mathsf{R}\}$ computes $m_{p,p_i} = o_{p_i,p_{i+1}} - o_{p_{i-1},p_i}$ and all parties jointly invoke $\texttt{MultiCast}(\mathsf{P}_{p_i}, (p, m_{p,p_i}))$ where p is encoded in $n\log(n)$ bits.
- For each path p the receiver R computes share $s'_p = \left(\sum_{p_i\in\mathcal{N}_p\setminus\{\mathsf{R}\}} m_{p,p_i}\right) - o_{p_{\ell-1},p_\ell}$.
- The receiver outputs $m' = \sum_{p\in\mathsf{GoodPaths}} s'_p$.

[a] In case of multiple disjoint sets, we can assume a lexicographic ordering among them and choose the smallest one.

Fig. 1: An efficient perfectly-secure private communication protocol against $t < n$ passive corruptions in a network with connectivity $k > t$.

not know s_p), so the simulated view for those paths is again indistinguishable from the real protocol. *Complexity.* In the first round each party sends at most $n|m|$ bits. The first phase additionally sends n^2 bits over multi-cast channels. The second phase communicates at most $n(n\log(n)+|m|)$ over multi-cast channels. □

Corollary 1. *If the multi-cast channels in Protocol 1 are instantiated with* `MultiCast` *from Lemma 1, the protocol runs for* $2n+1$ *rounds and has a total communication complexity of* $\mathcal{O}\big(n^5\log(n)+n^4|m|\big)$ *bits per party.*

Proof. This follows from the numbers in Theorem 1 and Lemma 1. □

Observe that Protocol 1 only requires connectivity k in the first round; after that, connectivity 1 suffices.

Corollary 2. *Protocol 1 is a private communication protocol that achieves perfect security against* $t<n$ *passive corruptions for connected network* $\mathcal{G}$ *if the first round graph is guaranteed to be* (S,R)*-*k*-connected for* $k>t$*.*

We can improve on the round complexity of Protocol 1 by optimizing the multicast protocol.

Lemma 2. *If* $\mathcal{G}$ *has connectivity* k*, then the flooding protocol from Lemma 1 requires at most* $\lfloor\frac{n}{k}\rfloor+1$ *rounds and total communication at most* $(\lfloor\frac{n}{k}\rfloor+1)n|m|$ *to distribute a message* m *among all parties.*

Proof. We show this by induction. In each round at least k parties learn the message, until the final round where all remaining parties learn the message. Consider any round r and a party P that did not receive the message before r. There must be at least k disjoint paths from the sender to P in the round r graph. Along each such path, there must be a party who knows the message (possibly the sender) and a party who does not (possibly P). This means that P learns the message, or at least k other parties learn the message. Thus, in round r at least k parties will learn the message or all parties will know the message after the round. As there are n parties, there can be at most $\lfloor\frac{n}{k}\rfloor$ rounds where at least k parties learn the message. This means after $\lfloor\frac{n}{k}\rfloor+1$ rounds, every party knows the message.

Corollary 3. *If* $\mathcal{G}$ *has connectivity* k*, and the multi-cast channels in Protocol 1 are instantiated with* `MultiCast` *as in Lemma 2, the protocol runs for* $2\lfloor\frac{n}{k}\rfloor+3$ *rounds, and has total communication complexity of* $\mathcal{O}\Big(\frac{n^4}{k}(n\log(n)+|m|)\Big)$*.*

In particular, if the connectivity k is a constant fraction of n, the protocol achieves a constant round complexity.

3.2 Private Communication on Undirected Graphs

In this section we consider undirected $\mathcal{G}$ with connectivity $t+1$. The undirectedness allows us to save (almost) half the round complexity compared to Protocol 1 using the following idea.

Each party starts with a local value. For the sender, this is the message m. Every other party starts with 0. As in Protocol 1, parties attempt to send a random field element to each potential neighbor. This defines an (undirected) meta-graph $\mathbb{G}$ of parties that have successfully exchanged random values. Since the graph is undirected, parties know their own neighborhood.[8]

[8] This is because if a party received randomness from another party, it can infer that the other party received the randomness sent by it as well.

In contrast to the above protocol, parties do not need to learn the full $\mathbb{G}$; it is enough to know one's own neighborhood. Parties use successfully sent random values to define a sharing of their local value in their closed neighborhood; i.e., their own share is their local value minus the sum of sent out random values. Next, parties add up all the shares they hold (including their own share of their local value). This establishes an additive sharing of the sender's message. Everyone but the receiver multi-casts their shares, allowing the receiver to reconstruct the message.

Remark 2. This protocol can be extended to undirected graphs. All that is needed is for parties to learn if their sent random values were received. To achieves this, parties could multi-cast the set from whom they received a random value in the first round. However, this brings the round complexity back to the one of Protocol 1.

Protocol 2: $\Pi(\mathsf{S}, \mathsf{R}, m)$

The identities of the sender S and receiver R are public input. The sender S additionally has message m as private input.
Let $\bar{\mathsf{G}} = \bigcup_{\mathsf{G}\in\mathcal{G}} \mathsf{G}$.

Establishing sharing of message. Party P_i does the following:
- If $\mathsf{P}_i = \mathsf{S}$, set $m_i = m$. Otherwise, set $m_i = 0$.
- For each neighbour P_j in $\bar{\mathsf{G}}$, sample uniform random $r_{i,j} \in \mathbb{F}$.
- Attempt to send $r_{i,j}$ to P_j using the normal channels. Denote by In_i the set of all parties P_j from whom P_i actually received randomness $r_{j,i}$ in this round.
- Compute $r_{i,i} = m_i - \sum_{j\in\mathsf{In}_i} r_{i,j}$.
- Compute $r_i = r_{i,i} + \sum_{j\in\mathsf{In}_i} r_{j,i}$.

Send message shares to receiver. Parties jointly invoke $\texttt{MultiCast}(\mathsf{P}_i, r_i)$ for $\mathsf{P}_i \neq \mathsf{R}$.
Reconstruction of message. The receiver $\mathsf{P}_i = \mathsf{R}$ computes $m = \sum_{\mathsf{P}_i\in\mathcal{P}} r_i$.

Fig. 2: An efficient perfectly-secure private communication protocol against $t < n$ passive corruptions in an undirected network with connectivity $k > t$.

Theorem 2. *Protocol 2 is a private communication protocol that achieves perfect security against $t < n$ passive corruptions for an undirected, connected network $\mathcal{G}$ with connectivity $k > t$. The protocol communicates at most $n^2|m|$ over $\mathcal{G}$ and sends at most $n|m|$ bits over multi-cast channels.*

Proof. Correctness. The symmetry of $\mathcal{G}$ ensures that if $\mathsf{P}_j \in \mathsf{In}_i$, then P_j has received a random value from P_i and will use it to compute r_j. This implies

$$\begin{aligned} m &= \sum_{\mathsf{P}_i\in\mathcal{P}} r_i = \sum_{\mathsf{P}_i\in\mathcal{P}} \left(r_{i,i} + \sum_{j\in\mathsf{In}_i} r_{j,i} \right) = \sum_{\mathsf{P}_i\in\mathcal{P}} \left(m_i - \sum_{j\in\mathsf{In}_i} r_{i,j} + \sum_{j\in\mathsf{In}_i} r_{j,i} \right) \\ &= \sum_{\mathsf{P}_i\in\mathcal{P}} m_i + \underbrace{\sum_{\mathsf{P}_i\in\mathcal{P}} \sum_{j\in\mathsf{In}_i} r_{j,i} - \sum_{\mathsf{P}_i\in\mathcal{P}} \sum_{j\in\mathsf{In}_i} r_{i,j}}_{=0} \\ &= m \end{aligned}$$

The last two double sums cancel out as one sums up all the received random values and the other all the sent random values which is the same set. The multi-cast ($\mathcal{G}$ is connected) ensures that the receiver gets all r_i. The receiver will therefore output m.

Privacy. If the sender or the receiver are corrupted, the adversarial view can be simulated by running the actual protocol in the head.

Otherwise, observe that the only message that depends on m is r_i for $\mathsf{P}_i = \mathsf{S}$ (which is multi-cast). There must exist an all-honest path from S to R, as $\mathcal{G}$ has connectivity of at least $t+1$. Let $\mathsf{S} = \mathsf{P}_{i_1}, \ldots, \mathsf{P}_{i_l} = \mathsf{R}$ be the parties on that path. If all other parties are corrupt, what the adversary learns is

$$\begin{aligned}
x_1 &= m - r_{i_1,i_2} + r_{i_2,i_1}, \\
x_2 &= 0 - r_{i_2,i_1} - r_{i_2,i_3} + r_{i_1 i_2} + r_{i_3,i_2}, \\
&\ldots \\
x_{l-2} &= 0 - r_{i_{l-2},i_{l-3}} - r_{i_{l-2},i_{l-1}} + r_{i_{l-3},i_{l-2}} + r_{i_{l-1},i_{l-2}}, \\
x_{l-1} &= 0 - r_{i_{l-1},i_{l-2}} - r_{i_{l-1},i_l} + r_{i_{l-2},i_{l-1}} + r_{i_l,i_{l-1}}.
\end{aligned}$$

Note that the values $r_{i_l,i_{l-1}}$ and r_{i_{l-1},i_l} only appear as summands in x_{l-1}; these are both randomly chosen, and thus $r_{i_l,i_{l-1}} - r_{i_{l-1},i_l}$ perfectly masks $r_{i_{l-2},i_{l-1}} - r_{i_{l-1},i_{l-2}}$. That in turn perfectly masks $r_{i_{l-3},i_{l-2}} - r_{i_{l-2},i_{l-3}}$ in x_{l-2}, and so on; finally, we get that $r_{i_1,i_2} - r_{i_2,i_1}$ perfectly masks m in x_1. We can conclude that r_i for $i \in \{i_1, \ldots, i_l\}$ (where r_i contains x_i as a summand) are independent and uniform. Thus, the adversarial view can be simulated by running the actual protocol in the head with $m = 0$.

Complexity. In the first round each party sends at most $n|m|$ bits. Additionally, there is communication of $n|m|$ bits over multi-cast channels. □

Corollary 4. *If the multi-cast channels in Protocol 2 are instantiated with* `MultiCast` *from Lemma 1, the protocol runs for $n+1$ rounds and has a total communication complexity of $\mathcal{O}(n^4|m|)$.*

Proof. This follows from the numbers in Theorem 2 and Lemma 1. □

Observe that Protocol 2 only requires connectivity k in the first round; after that, connectivity 1 suffices.

Corollary 5. *Protocol 2 is a private communication protocol that achieves perfect security against $t < n$ passive corruptions for undirected connected network $\mathcal{G}$ if the first round graph is guaranteed to have connectivity $t+1$.*

We can improve on the round complexity of Protocol 2 by optimizing the multicast protocol.

Corollary 6. *If $\mathcal{G}$ has connectivity k, and the multi-cast channels in Protocol 1 are instantiated with* `MultiCast` *as in Lemma 2, the protocol runs for $\lfloor \frac{n}{k} \rfloor + 2$ rounds, and has total communication complexity of $\mathcal{O}\left(\frac{n^4|m|}{k}\right)$.*

4 Reliable Communication with Active Corruption

In this section, we construct a network flooding protocol for reliable communication with security against active adversaries. In this kind of protocol, the best attack is essentially to select t parties to corrupt, and have these parties ignore the honest messages and transmit lies. This attack is limited in two ways: every lie must originate from one of only t parties, while if the graph is at

least k-connected then there must be at least $k-t$ disjoint paths from the original sender along which the honest message can pass. Since the sender did not send the lie, every path the lie was purported to have been sent along must go through some corrupt party. [DRTY23] therefore proposed flooding the network until a message is heard from more than t disjoint paths, which will happen eventually if $k - t > t$ (i.e., $k > 2t$). Unfortunately, there can be exponentially many paths from the sender in the graph, so their protocol does not work in polynomial time or polynomial communication.[9]

We present a more efficient flooding protocol, based on tracking the (polynomial-size) communication graph along which a message has been sent, instead of all paths. As motivation, we start with a protocol for the case of static graphs. Recall that every path along which a lie was purported to have been transmitted from S to a R must go through a corrupted party. Rephrased in terms of graphs, this means that the corrupted parties must form a (S, R)-cut[10] of this graph, with size t. This suggests an protocol along these lines:

1. Flood the network with the sender's message m. Alongside the messages, transmit graphs that somehow representing where the message has come from.
2. If a receiver R notices that a particular message m is associated with a minimum (S, R)-cut greater than t, then output m.

There are three challenges with this approach:

1. Min-cut is only defined for a static graph, and we want a protocol that works for dynamic communication graphs.
2. Corrupt parties must be restricted in how they can influence the graph showing where a message came form. Otherwise, as we explain below, they might introduce a fake path from S to R that only goes through honest parties.
3. For efficiency, the graphs must have polynomial size.

Challenge 1 we overcome by introducing multiple vertices for each party — one for each round — with each vertex labeled by the party it represents, and by introducing the notion of a labeled cut. In a labeled cut, all vertices with the same label P can be cut at once, for a cost of 1. For Challenge 2, we additionally introduce multiple vertices for the same party even within the same round, representing different claims about how that party heard about the message. To see why this is needed, imagine that during round r an honest P_h hears l from a corrupt party P_j (who claims to have heard it from S), then in round $r+1$, P_h sends l to another honest party $\mathsf{P}_{h'}$ (who reports it to R in round $r+2$), while simultaneously P_j claims to R that in round r it heard l from P_h (who heard it from S). Even though l has only been claimed to have traveled down the two paths $\mathsf{S} \to \mathsf{P}_j \to \mathsf{P}_h \to \mathsf{P}_{h'} \to \mathsf{R}$ and $\mathsf{S} \to \mathsf{P}_h \to \mathsf{P}_j \to \mathsf{R}$, which both go through the corrupt party, if you treat all instances P_h in round r as a single node then you would add the path $\mathsf{S} \to \mathsf{P}_h \to \mathsf{P}_{h'} \to \mathsf{R}$ to the graph, which does not go through a corrupted party. Therefore, we treat the P_h who heard l from P_j as a separate node from the P_h who was claimed to have heard l from S. This means that if R hears l from P_h and P_i, it won't think that there was a path directly from S through P_h to R. Finally, for Challenge 3 we perform a kind of deduplication on the graphs to keep them polynomially sized, while still keeping separate the nodes representing different claims of message provenance.

4.1 Labeled min-cut

An (s, d)-(vertex)-cut of a directed graph G is a partition $\{S, C, D\}$ of the vertices of G where $s \in S$ and $d \in D$, such that there are no edges from S to D. The minimum (s, d)-cut problem is

[9] Even polynomial round complexity has not been proven.
[10] In this paper we only consider vertex cuts, not edge cuts.

to find an (s,d)-cut that minimizes $|C|$. Now let G be a labeled directed graph, where we have a function l mapping from vertices of G to labels in some set L. One may think of L as the set of parties in our protocol. Recall that in the previous section we argued that we need graphs with several vertices labelled as one party. Similarly, we can now define an integral labeled (s,d)-cut as follows.

Definition 5. *Let G be a directed graph with labels L and labeling function l. A integral* labeled (s,d)*-(vertex)-cut of G is a partition $\{S, C', D\}$ of the vertices of G such that there are no edges from S to D and C' is the preimage $l^{-1}(C)$ for some set $C \subseteq L$.*

That is, we only want to consider cuts that correspond to an actual set of parties (i.e., labels). The integral labeled minimum (s,d)-cut problem is then to minimize $|C|$ over all such cuts. This is the vertex-labeled cut analog of the edge-labeled cuts introduced by [DHKM16].

In the protocol sketched above, the adversary will corrupt a subset C of the parties (i.e., labels), and any lie will have to traverse this subset. Letting S be the connected component of s after removing $C' = l^{-1}(C)$ and D be the remainder of the graph then gives an integral labeled (s,d)-cut. Indeed, our final protocol would still be secure if it used integral labeled cuts. However, unlike for unlabeled cuts, we do not know how to efficiently compute minimal integral labeled (s,d)-cuts. In fact the problem is NP-hard: there is an easy reduction to weighted monotone satisfiability, which is NP-hard (see, e.g., [DF98]).

$$\begin{aligned} &\min \sum_{v \in G} C_v \\ &\text{subject to} \\ &S_v + D_w \le 1 && \forall (v,w) \in G \\ &S_v + D_v + C_v = 1 && \forall v \in G \\ &S_v, D_v \in [0,1] && \forall v \in G \\ &C_v \in [0,1] && \forall v \in G \\ &S_s = D_d = 1 \end{aligned}$$

(a) Linear program for (s,d)-min-cut.

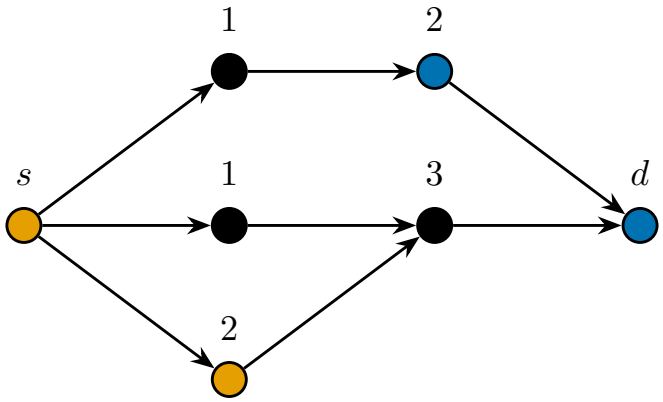
$$\begin{aligned} &\min \sum_{\ell \in L} C_\ell \\ &\text{subject to} \\ &S_v + D_w \le 1 && \forall (v,w) \in G \\ &S_v \le S_w && \forall (v,w) \in G, l(v) = l(w) \\ &S_v + D_v + C_{l(v)} = 1 && \forall v \in G \\ &S_v, D_v \in [0,1] && \forall v \in G \\ &C_\ell \in [0,1] && \forall \ell \in L \\ &S_s = D_d = 1 \end{aligned}$$

(b) Linear program for labeled (s,d)-min-cut.

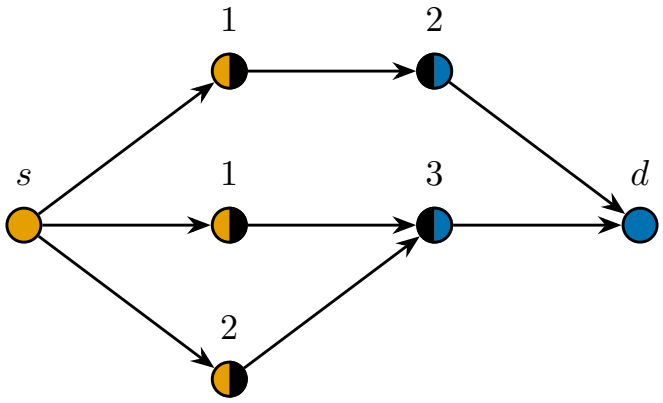
Fig. 3: Linear programs for min-cut.

To see why, view these problems as integer linear programs. In Figure 3a we have written a linear program for unlabelled min-cut, which exactly matches the definition above if the variables $\{S_v, C_v, D_v\}$ are restricted to being integers. For each vertex v, exactly one of (S_v, C_v, D_v) will be 1 and the others will be zero, so these variables will define a partition of the vertices of G. The constraint that no edges connect S to D is enforced by $S_v + D_w \le 1$ for all edges (v,w), i.e., we cannot have both $v \in S$ and $w \in D$. We have $s \in S$ and $d \in D$ because of the constraints $S_s = 1$ and $D_d = 1$.

While in general integer linear programs can be hard to solve, this particular one is easy. That is, the constraint matrix is totally unimodular (after eliminating the D_v variables using the equality constraint $D_v = 1 - S_v - C_v$), and constants in the constraints are integers, so every vertex of the polytope defined by the constraints has integer coordinates. Therefore, if you



(a) An integral labeled min-cut.



(b) A relaxed labeled min-cut.

Fig. 4: Example graph where the integral labeled (s, d) min-cut has size 2, but the relaxed labeled min-cut has size $\frac{3}{2}$. The cut is black, the source component orange, and the sink component blue. Any subset of two of three labels $\{1, 2, 3\}$ gives an integral labeled min-cut, while the relaxed labeled min-cut is half of all three labels.

solve the linear programming relaxation of the problem (i.e., solve the same linear programming problem, but without requiring that the variables take integer values) then the minimal $\sum_{v \in G} C_v$ will be exactly the same as in the integer problem. This shows that min-cut can be solved in polynomial time [Kha79].

We present a similar linear program for labeled min-cut in Figure 3b. The only significant change is that cut variables C_v are now defined on labels instead of vertices, to match the set C' of cut vertices being defined as $C' = l^{-1}(C)$ in this problem.[11] Unfortunately, this new system is not totally unimodular, because the same cut variable C_ℓ gets used with many different vertices. In fact, the linear programing relaxation can now have a min-cut with a fractional min $\sum_{\ell \in L} C_\ell$ (see Figure 4). Because we desire a computationally efficient protocol, we will use only the relaxed labeled (s, d)-cuts, rather than integral labeled (s, d)-cuts. Our analysis will show that the protocol will still be secure, even though these relaxed labeled cuts are only an approximation to the integral labeled cuts that describe the possible attacks.

[11] There is another change, which is the added constraint $S_v \leq S_w$ for all edges $(v, w) \in G$ where v and w have the same label ℓ. This does not affect the integer linear program, as either $C_\ell = 0$, in which case it's implied by $S_v + D_w \leq 1$ and $D_w = 1 - S_w - C_\ell$, or $C_\ell = 1$, in which case $S_v = S_w = 1$. The constraint should be viewed as tightening the approximation of the linear program relaxation, as the constraint becomes non-trivial if C_ℓ is not an integer.

4.2 Multicast Protocol

Protocol 3: $\texttt{MultiCast}(\mathsf{S}, m)$

The sender S is public input. S has message $m \in \{0,1\}^*$. Denote by $\bar{\mathsf{G}} = \bigcup_{\mathsf{G} \in \mathcal{G}} \mathsf{G}$.

Initialization. Every party P_i has an associative array D_i mapping from possible messages (i.e, from $\{0,1\}^*$) to DAGs with nodes labeled by party identifiers.

- For all i, set D_i to be empty, so that $\mathsf{D}_i[m']$ is the empty graph for all $m' \in \{0,1\}^*$.
- For $\mathsf{P}_i = \mathsf{S}$, set $\mathsf{D}_i[m] := (\{s\}, \{s \mapsto \mathsf{S}\})$, i.e., the graph containing a single node labeled as S.

Network flooding. For ρ rounds, every party P_i does the following:

- For each neighbor P_j in $\bar{\mathsf{G}}$, attempt to send D_i to P_j.
- For each message $\mathsf{D}_{j,i}$ that received from P_j, and for each m' such that $\mathsf{D}_{j,i}[m']$ is non-empty:
  - Check that $\mathsf{D}_{j,i}[m']$ has exactly one sink, which must be labeled P_j.
  - If so, update $\mathsf{D}_i[m'] := \mathsf{D}_i[m'] + \mathsf{D}_{j,i}[m']$. (I.e., take the disjoint union.)
- For any m' where $\mathsf{D}_i[m']$ is not empty,
  - Connect every sink in $\mathsf{D}_i[m']$ to a new node d, labeled as P_i. Now d is a unique sink.
  - Compress the DAG with $\mathsf{D}_i[m'] := \texttt{Dedup}(\mathsf{D}_i[m'])$.

Output condition. Each party P_i locally decides when it knows the message sent by S. If there exists a $m' \in \{0,1\}^*$ such that $\mathsf{D}_i[m']$ is non-empty, output m' if

- $\mathsf{D}_i[m']$ has a source s labeled as S;[a] and
- The relaxed labeled (s, d) min-cut of $\mathsf{D}_i[m']$ is greater than t, where d is the unique sink (labeled as P_i) in $\mathsf{D}_i[m']$.

[a] s is unique if it exists, because of `Dedup`.

Function $\texttt{Dedup}(\mathsf{D})$

Repeat the following merger operation until a fixed point is reached.

- Find two nodes x, y of D such that x and y have the same label and the same direct predecessors.
- Merge x and y into a node z that has the same label and direct predecessors, and the union of the x's and y's direct successors.

Output D.

Fig. 5: Our actively secure multicast protocol.

We present our actively secure multicast Protocol 3. In each round, every party sends to its neighbors an associative array (i.e. a key-value store) containing all of the messages they've heard so far (i.e. the keys), together with the labeled directed acyclic graphs representing the paths along which the message is purported to come from the sender S (i.e. the values). Each party then takes a kind of union of all the graphs it has heard (together with the graph from the previous round), where nodes representing the same message provenance are identified. This

is done by taking the disjoint union of the graphs, then running `Dedup` to merge nodes when they are labeled with the same party and have the same predecessor nodes. Finally, each party computes the relaxed labeled min-cut of the each message's graph, which lower bounds the number of lies needed to cause this graph to appear for a message that was not sent by S.[12] The party determines that the message is correct and outputs it if this min-cut is greater than t.

Theorem 3. *Protocol 3 is a multicast protocol that achieves perfect security against t parties for networks $\mathcal{G}$ with connectivity $k > 2t$. It completes in $\rho = 1 + \lfloor \frac{n-t-2}{k-2t} \rfloor$ rounds, uses at most $\mathcal{O}\big(n\rho d_{avg}|m| + (M+1)n^2\rho^2 d_{avg}^2 \log_2(n\rho)\big)$ bits of total communication, and runs in polynomial time. Here, $|m|$ is an upper bound on the size of the message, d_{avg} is an upper bound on the average degree of the communication graph of every round, and the M is the total bits of all messages sent by corrupt parties.*

Proof. We must show that the protocol does not output a lie ("security"), that it eventually outputs the correct message ("correctness"), and that the communication cost is bounded ("efficiency").

Security. Security comes down to the following property: for every honest party P_i and for any $m' \neq m$, after every round all paths from source s to sink d in $\mathsf{D}_i[m']$ will go through a corrupted party. We call this the "corrupt paths property" for d. If this property holds, then cutting the t labels corresponding to corrupted parties in $\mathsf{D}_i[m']$ will disconnect s from d, so there is a labeled min-cut of size t, and so P_i won't output m'.

We prove this property by induction. In the base case, all $\mathsf{D}_i[m']$ is the empty graph, so the property holds vacuously. In each round, $\mathsf{D}_i[m']$ grows only by receiving $\mathsf{D}_{j,i}[m']$ from some other party P_j.[13] P_j can either be corrupt or honest. If it is corrupt, before adding $\mathsf{D}_{j,i}[m']$ to the graph, P_j checks that it it's unique sink is labeled P_j. Therefore, every path in $\mathsf{D}_{j,i}[m']$ to this sink must go through a corrupt party, P_j. If P_j is honest, then by the induction hypothesis every path from s to the sink must go through a corrupt party. Next, a new sink d is added (labeled P_i), and all existing sinks are attached to it. Since the corrupt paths property was satisfied by all previous sinks, it will be satisfied by d.

Finally, P_i runs $\mathsf{D}_i[m'] := \texttt{Dedup}(\mathsf{D}_i[m'])$ to remove redundant nodes. We must show that this operation preserves the corrupt paths property. For any path from s' to d' in the deduplicated graph, there exists a corresponding path from s to d in the original graph, sharing exactly the same labels. To see this, follow the path in reverse, from d' to s'. Note d must not have been merged, because it is the unique sink in the original graph and the graph is acyclic, so initially the correspondence is unambiguous. At each step, if x' has predecessor y' in the path, select a predecessor y of x such that y was merged into y' during `Dedup`. Such a node must exist because x is a node that was merged into x', and all nodes merged together shared the same set of direct predecessors. The path will have exactly the same labels because nodes are only merged if they have the same label. Since the corrupt paths property holds for the original graph, it therefore holds for the deduplicated graph. We have now proven that the hypothesis holds after the round is finished.

Correctness. Let the sequence of network graphs be $\mathsf{G}_1, \ldots, \mathsf{G}_\rho$. Remove the corrupt nodes, then combine them into a single acyclic graph $\mathbb{G}$ representing all honest communication paths that could occur. In more detail, construct $\mathbb{G}$ by creating vertices v_{ir} for all honest parties P_i and all

[12] Our protocol would also work with the integral labeled min-cut instead, which would give the exact minimum number of lies, except that this is hard to compute.

[13] Or if no new graphs are received for m', it grows only by adding a new sink node d; this is discussed later.

rounds $r \in \{0, \ldots, \rho\}$. For each edge (P_i, j) in G_r, create an edge $(v_{i(r-1)}, v_{jr})$ in $\mathbb{G}$ to represent that P_i will send a message to P_j in round r. Additionally, add an edge $(v_{i(r-1)}, v_{ir})$ for all parties P_i and rounds r, since P_i preserves its state from rounds. Then we have:

Lemma 3. *The relaxed labeled $(v_{\mathsf{S}0}, v_{\mathsf{R}\rho})$-min-cut of $\mathbb{G}$ is greater than t, for any honest party R.*

Using this lemma, we will show that every honest party P_i in our protocol will output the correct message m by round ρ. That is, we will show that $\mathsf{D}_i[m]$ will contain a copy of the relevant subgraph of $\mathbb{G}$: all nodes that are on some path from $v_{\mathsf{S}0}$ to $v_{i\rho}$.[14]

At the start, the only non-empty $\mathsf{D}_i[m]$ is when $\mathsf{P}_i = \mathsf{S}$, where it contains a solitary node. Call this node $v_{\mathsf{S}0}$. In every round, honest parties will send each other their graphs, take the disjoint union, deduplicate, and add new sinks d to their own graphs. Let v_{ir} be the sink added by P_i at the end of round r. It will have an edge from the previous sink $v_{i(r-1)}$, as well as edges from $v_{j(r-1)}$ for all honest P_j that P_i heard from in round r. Note that any such node v_{ir} will always get deduplicated, as there the honest parties do not modify the predecessors of nodes in their graphs before sending them on, so there will be at most one copy of any v_{ir} in any of these graph. Since all paths in $\mathbb{G}$ are exactly the paths of honest parties are communicating along, P_i will have heard about v_{jr} by round ρ if and only if v_{jr} is on some path from $v_{\mathsf{S}0}$ to $v_{i\rho}$.

Nodes not on any path $v_{\mathsf{S}0}$ to $v_{i\rho}$ are irrelevant for determining the minimum cut, so by Lemma 3 each honest party P_i will output m by round ρ. It remains only to prove this lemma.

Proof (Proof of Lemma 3.). Assume that there exists some relaxed labeled $(v_{\mathsf{S}0}, v_{\mathsf{R}\rho})$-cut $(\{C_{\mathsf{P}_i}\}_i, \{S_v, D_v\}_{v\in\mathbb{G}})$ where $\sum_i C_{\mathsf{P}_i} \le t$. To avoid using double subscripts, we will write S_{ir} for $S_{v_{ir}}$, and similarly for C and D. Let $S_r = \sum_i S_{ir}$ represent the "progress" made in the flooding on round r. It is essentially the total number of nodes that are on the sender's side of the cut, except that the variables S_v need not be integers. For all r we have $S_{\mathsf{S}r} = D_{\mathsf{R}r} = 1$, as $S_{\mathsf{S}0} = D_{\mathsf{R}\rho} = 1$ by assumption that it is a $(v_{\mathsf{S}0}, v_{\mathsf{R}\rho})$-cut, and there are edges from $v_{\mathsf{R}(r-1)}$ to $v_{\mathsf{R}r}$ for all r. We also have $S_0 \ge S_{\mathsf{S}0} = 1$.

Next, for all rounds r there exists (S, R)-cut of the honest subgraph of G_r with size $t + S_r - S_{r-1}$: let $S'_i = S_{i(r-1)}$, $C'_i = C_i + S_{ir} - S_{i(r-1)}$, and $D'_i = D_{ir}$. We can now show that (S', C', D') is a cut, assuming that (S, C, D) is a labeled cut:

$$
\begin{aligned}
S'_i + D'_j &= S_{i(r-1)} + D_{jr} \le S_{ir} + D_{jr} \le 1 \\
S'_i + D'_i + C'_i &= S_{i(r-1)} + D_{jr} + C_i + S_{ir} - S_{i(r-1)} = S_{ir} + D_{jr} + C_i = 1 \\
S'_i &= S_{i(r-1)} \in [0, 1] \\
D'_i &= D_{jr} \in [0, 1] \\
C'_i &= C_i + S_{ir} - S_{i(r-1)} \ge C_i \ge 0 \\
C'_i &= 1 - S'_i - D'_i \le 1.
\end{aligned}
$$

The size of C' is $\sum_i C'_i = \sum_i (C_i + S_{ir} - S_{i(r-1)}) \le t + S_r - S_{r-1}$. While this is a relaxed cut, the relaxed min-cut coincides with the integral min-cut, so there exists some cut of honest subgraph of G_r with size at most $t + S_r - S_{r-1}$. By assumption, G_r is k-connected, so after removing the corrupted nodes it is $(k - t)$-connected. Therefore, G_r's honest subgraph has no cuts smaller than $k - t$. We now have $t + S_r - S_{r-1} \ge k - t$, or $S_r - S_{r-1} \ge k - 2t$.

[14] In general it can contain much more, because the corrupted parties can make up whatever graph they choose. But it will always contain this subgraph of $\mathbb{G}$.

We have lower bounded S_0, and lower bounded its increase in each round, so at the end we get

$$S_\rho \geq 1 + \rho(k - 2t) \geq 1 + \left(1 + \left\lfloor \frac{n - t - 2}{k - 2t} \right\rfloor\right)(k - 2t) > 1 + n - t - 2 = n - t - 1.$$

However, there are only $n - t - 1$ honest parties other than R, and $S_{\mathsf{R}\rho} = 0$, so $S_\rho \leq n - t - 1$. This is a contradiction.

Efficiency. We start by upper bounding the size of the graphs begin sent when all parties are honest. Let $\mathsf{D}[m] = \texttt{Dedup}(\sum_i \mathsf{D}_i[m])$ be the deduplication of the disjoint unions of the graphs of all honest parties. Every $\mathsf{D}_i[m]$ is then a subgraph of $\mathsf{D}[m]$, so the cost of sending $\mathsf{D}_i[m]$ is upper bounded by the cost of sending of $\mathsf{D}[m]$. This graph only grows as the protocol continues, so need only bound the size of the final value of $\mathsf{D}_i[m]$. In every round $\mathsf{D}[m]$ will increase by n nodes and $d_{\text{avg}}n$ edges, because every party will add a new sink connected to the sinks from the previous round of all parties it has heard from. Note that all of these honest nodes will always get deduplicated after the disjoint union operations, because all copies of these honest nodes will have the same predecessors.

We use a sparse representation of the graph, and compute it's communication cost as $2E \log_2 V + V \log_2 n$, where E and V are the number of vertices. That is, each edge specifies its two endpoints, and each vertex specifies its label. Note that $V \geq n$, so this communication cost is upper bounded by $(2E + V) \log_2 V$. If everybody is honest, this upper bounds the message size by

$$|m| + (2nd_{\text{avg}}\rho + n\rho + 1) \log_2(n\rho + 1) = |m| + (n\rho(2d_{\text{avg}} + 1) + 1) \log_2(n\rho + 1),$$

because the graph starts with one vertex (the sender) and zero edges. The total communication cost of graphs would then be $n\rho d_{\text{avg}}(n\rho(2d_{\text{avg}} + 1) + 1) \log_2(n\rho + 1)$, as there are ρ rounds and at most nd_{avg} messages sent in each round.

Next, we consider how the corrupt parties could increase the communication cost. They can do some combination of sending lies (i.e., sending $m' \neq m$), increasing the number of graphs for honest parties to send, and sending graphs, increasing the size of each graph the honest parties send. The total communication increases linearly with the number of lies, as essentially the same protocol is run for lies as for m. The communication also increases linearly with the size of each graphs sent by a corrupt party, as the corrupt graph will become part of $\mathsf{D}[m]$. However, sending lies increases the communication faster than sending graphs, as sending lies causes honest parties to add an additional $n\rho$ nodes to the graph themselves, while sending graphs only increases the size of an existing graph by the amount you sent. Therefore, the best attack is to spend the corrupt party's communication entirely on lies, multiplying the total communication for graphs by $M + 1$. Adding on the cost of sending every message on every round, this gives an total communication cost of

$$n\rho d_{\text{avg}}(M + 1)(n\rho(2d_{\text{avg}} + 1) + 1) \log_2(n\rho + 1) + n\rho d_{\text{avg}}(|m| + M).$$

Finally, note that based on these bounds all graphs remain polynomial in size. The relaxed labeled min-cut can be computed in polynomial time using linear programming, so the entire protocol then runs in polynomial time.

5 Private Communication with Active Corruption

5.1 Feasibility of Perfect Security When $k > 2t$

In this section, we disprove the conjecture by Damgård et al. [DRTY23] by showing feasibility of perfectly-secure private communication when $3t \geq k > 2t$. First, we describe the building blocks used by Protocol 4.

Reliable Channel as a Building Block. We note that actively-secure reliable communication protocols that are secure against t corruptions in a dynamic network exist, as long as $k > 2t$. Thus, for simplicity, we use reliable channels as a primitive in Protocol 4. These could be realized by using our reliable communication Protocol 3 or alternately by using the (more expensive) protocol of Damgård et al. [DRTY23]. The communication complexity of the former is upper bounded by $\mathcal{O}\big(n^3|m| + Mn^6 \log_2(n)\big)$ where $|m|$ denotes the number of message bits and M is an upper bound on the number of bits communicated by corrupt parties.

Perfect Secure Message Transmission (PSMT). We use a perfect secure message transmission protocol, which informally speaking, allows a sender to send a private message securely to a receiver in a network where the sender and receiver are connected by n secure channels, among which up to t channels could be controlled by the adversary. More formally, PSMT can be defined as below.

Definition 6. *Assume there are k secure channels between a sender S and a receiver R. A protocol between S (with input message m) and R is a perfect secure message transmission protocol if it satisfies* privacy *and* correctness *as per Definition 2 against any adversary $\mathcal{A}$ corrupting at most t out of the k channels.*

Dolev et al. [DDWY93] showed the following:

Lemma 4 ([DDWY93]). *PSMT is achievable if and only if $k \geq 2t + 1$.*

We observe that any such PSMT protocol should be secure even if there are potentially fewer than $2t+1$ channels (say, there are $\delta < 2t+1$ channels), as long as there are at least $t+1$ uncorrupted channels. This is because one can always augment this network with $2t + 1 - \delta$ dummy channels, where the dummy channels can be considered as being controlled by the adversary (as the adversary corruption budget of t allows for this). This observation allows us to state the below lemma.

Lemma 5. *Assume a party S and another party R are connected by $k \geq t+1$ secure channels, among which at least $t+1$ channels are uncorrupted. Then, there exists a protocol $\Pi_{\mathtt{SMT}}(\mathsf{S}, \mathsf{R}, m)$ that allows S to securely communicate a private message m to R.*

Looking ahead, in our Protocol 4, we use the protocol $\Pi_{\mathtt{SMT}}$ in a non-black box fashion by replacing the *secure* channels used by such protocols with reliable channels and suitably masking the private message using established shared keys. Shared keys are established as in our protocols in the passive setting. Lastly, we point that $\Pi_{\mathtt{SMT}}$ can be instantiated using any existing efficient PSMT construction, such as the two round protocol of [SZ16] which has a communication complexity of $\mathcal{O}\big(n^2 \log n\big)$ bits to securely communicate a 1-bit secret.

Theorem 4. *Protocol 4 is a private communication protocol that achieves perfect security against $t <$ nactive corruptions for network $\mathcal{G}$ with connectivity $k > 2t$.*

The protocol runs for $1 + 2\rho_{\mathtt{Rel}} + \rho_{\mathtt{SMT}}\rho_{\mathtt{Rel}}$ rounds, where $\rho_{\mathtt{SMT}}$ is the round complexity of the SMT protocol and $\rho_{\mathtt{Rel}}$ is the round complexity of the reliable communication protocol.

Protocol 4: $\Pi^{\mathsf{prv}}_{\mathsf{perf,mal}}(\mathsf{S},\mathsf{R},m)$

The identities of the sender S and receiver R are public input. The sender S has message $m \in \mathbb{F}$ as private input.
Let $\bar{\mathsf{G}} = \bigcup_{\mathsf{G}\in\mathcal{G}} \mathsf{G}$. We use the following building blocks:

- $\mathtt{Rel}(\mathsf{S},\mathsf{R},m)$, which allows reliable communication of a message m from S to R.
- $\Pi_{\mathtt{SMT}}(\mathsf{S},\mathsf{R},m)$: A perfect secure message transmission protocol that allows secure communication of a private message m over a network of $k \geq t+1$ secure channels, among which at least $t+1$ channels are uncorrupted.

Establishing shared randomness. Party P_i does the following:
- For each neighbor P_j in $\bar{\mathsf{G}}$, sample uniform random $r_{i,j}$ (of size $|m|$).
- In round 1, attempt to send $r_{i,j}$ to P_j using the communication network.
- Let In_i denote the set of parties P_j from whom P_i actually received randomness $r_{j,i}$ in round 1.
- For each neighbor P_j in $\mathbb{G}$ set $o_{i,j} \coloneqq r_{j,i} + r_{i,j}$ where missing values $r_{\cdot,\cdot}$ are set to 0.

Establishing meta graph of shared keys. This phase comprises of the following steps:
- For each party P_i, parties jointly invoke $\mathtt{Rel}(\mathsf{P}_i,\mathsf{S},\mathsf{In}_i)$, where In_i is encoded as an n-bit vector.
- The sender S builds the meta graph $\mathbb{G}$ as follows: Let $\mathcal{P}$ be the set of nodes. There is an edge between P_u and P_v if $\mathsf{P}_u \in \mathsf{In}'_v$ or $\mathsf{P}_v \in \mathsf{In}'_u$, where In'_u denotes the output of the instance $\mathtt{Rel}(\mathsf{P}_u,\mathsf{S},\mathsf{In}_u)$.
- For each party P_i, parties jointly invoke $\mathtt{Rel}(\mathsf{S},\mathsf{P}_i,\mathbb{G})$, where $\mathbb{G}$ is encoded in n^2 bits.

Secure message transfer in $\mathbb{G}$. All the parties can now locally determine the same set of disjoint paths, say $\mathsf{GoodPaths}$ in $\mathbb{G}$ between S and R (using the Ford-Fulkerson algorithm to get a maximal set of disjoint paths): If the algorithm returns at most $2t+1$ disjoint paths, use these as $\mathsf{GoodPaths}$, else use the first $2t+1$ paths returned. Consider a network $\mathbb{G}'$ where each disjoint path p in $\mathsf{GoodPaths}$ corresponds to a channel Ch_p. In order to emulate each step of an instance of $\Pi_{\mathtt{SMT}}(\mathsf{S},\mathsf{R},m)$ over $\mathbb{G}'$, parties do the following:
- If the step involves computation, it is done exactly as in the protocol $\Pi_{\mathtt{SMT}}$.
- If the step involves communicating a (private) message m' from S to R over a channel Ch_p:
  - Let $(\mathsf{P}_{i_1},\ldots,\mathsf{P}_{i_\ell})$ be the path corresponding to Ch_p, where $\mathsf{P}_{i_1} = \mathsf{S}$ and $\mathsf{P}_{i_\ell} = \mathsf{R}$.
  - The sender $\mathsf{S} = \mathsf{P}_{i_1}$ computes $m_{i_1} = m' + o_{i_1,i_2}$[a] and all parties jointly invoke $\mathtt{Rel}(\mathsf{S},\mathsf{R},(p,m'_{i_1}))$ where p is encoded in $n\log(n)$ bits.
  - Each party $\mathsf{P}_{i_j} \in (\mathsf{P}_{i_2},\ldots,\mathsf{P}_{i_{\ell-1}})$ computes $m'_{i_j} = o_{i_j,i_{j+1}} - o_{i_{j-1},i_j}$ and all parties jointly invoke $\mathtt{Rel}(\mathsf{P}_{i_j},\mathsf{R},(p,m'_{i_j}))$ where p is encoded in $n\log(n)$ bits.
  - The receiver R computes $m' = \left(\sum_{i_j\in(i_1,\ldots,i_{\ell-1})} m'_{i_j}\right) - o_{i_{\ell-1},i_\ell}$ as the value received via the channel Ch_p in this step.
- If the step involves communicating a (private) message m' from R to S over a channel Ch_p: Similar steps as above, with the roles of S and R interchanged.

R returns the output of $\Pi_{\mathtt{SMT}}(\mathsf{S},\mathsf{R},m)$ over $\mathbb{G}'$.

[a] We abuse notation here; assume that the shared key $o_{i,j}$ established earlier contains enough randomness that can be used to mask the private messages sent throughout all steps of $\Pi_{\mathtt{SMT}}$ and only the relevant part of the shared key corresponding to this particular private message m_p is used here.

Fig. 6: An perfectly-secure private communication protocol in a network with connectivity $k > 2t$.

It has a communication complexity of $\mathcal{O}\big(c_{\mathtt{Rel}}(n^2) + c_{\mathtt{SMT}}(|m|)(nc_{\mathtt{Rel}}(1 + n\log(n)))\big)$ *bits where* $c_{\mathtt{Rel}}(x)$ *denotes the communication complexity of reliably communicating* x *bits,* $c_{\mathtt{SMT}}(x)$ *denotes the communication complexity of SMT for* x *bits, and* $|m|$ *denotes the number of message bits.*

Lemma 6. *If none of* $(\mathsf{P}_{i_1}, \ldots, \mathsf{P}_{i_\ell})$ *are corrupt, then the steps from Protocol 4 for communicating a (private) message* m' *from* S *to* R *over channel* Ch_p *achieve private and reliable communication of* m'*.*

The proof of Lemma 6 is identical to that of Theorem 2, with the graph limited to the single path in question.

Proof (Proof of Theorem 4). The security of Theorem 4 then follows from the security of the SMT protocol used: clearly, all connections between honest parties will be present in $\mathbb{G}$ and since the subgraph of honest parties is at least $t+1$-connected, $\mathbb{G}'$ must contain at least $t+1$ all-honest paths. Moreover, we only care about security if S is honest, and in that case all honest parties will be in agreement regarding the network $\mathbb{G}'$. We can therefore rely on the security of the SMT protocol (that assumes a public network).

The round complexity of the protocol follows from the fact that establishing shared randomness takes one round; establishing the metagraph involves two phases of (concurrent) reliable communications; and finally secure transmission involves executing the SMT protocol, where each message in the SMT protocol involves (multiple parallel) invocations of the reliable communication protocol.

The communication complexity of the protocol follows from the fact that establishing the metagraph incurs $c_{\mathtt{Rel}}(n^2)$ complexity (since the reliable communication protocol directly achieves multicast); and secure message transfer involves n instances of reliable communication (each involving communication of $c_{\mathtt{Rel}}(1 + n\log(n))$) corresponding to each bit sent in the underlying SMT protocol.

Remark 3. If S is not honest, there are no requirements on security, but we can note that the protocol will still terminate and be efficient. In this case, parties may no longer agree on which instances of the protocol for reliable communication should be run in the last phase. So, effectively, the set of honest parties will be partitioned in subsets each trying to run their own set of instances. However, this will not affect the round complexity as each instance runs for a fixed number of rounds. As for the communication complexity, we can assume that honest parties will ignore messages from instances they do not think should be run, so we are effectively running at most a factor n more protocols in parallel. However, note that the complexity of the reliable communication protocol already incorporates a polynomial factor that depends on the behavior of the adversary (as indeed it must). This means that the adversary could make us work at least as hard by instead allowing parties to agree on $\mathbb{G}'$ but increase the number of incorrect messages it sends when the reliable communication protocols are run. Therefore, our expression for the communication complexity captures what can happen, even for a dishonest S.

Conjecture of Damgård et al. [DRTY23]. Lastly, recall that Damgård et al. conjectured that perfectly secure private communication in the active setting was impossible to achieve when $3t \geq k > 2t$. They gave evidence for the conjecture by arguing that any protocol in the class they considered would fail in this case. The assumption on the protocol was that the receiver would receive values from different sets of $t+1$ disjoint paths and would then try to identify the honest set of $t+1$ disjoint paths. However, this was shown to be impossible with 0 error probability when $3t \geq k > 2t$, since in the dynamic setting, an adversary can always fabricate paths. While this argument is correct, it does not cover all protocols, as also pointed out in

[DRTY23]. Indeed, our protocol does not fall in this class, as we crucially rely on fixing the meta graph using additional public communication, and such additional interaction was not considered in [DRTY23].The metagraph essentially gives us a means to work with the static setting instead, where perfect private communication is known to be possible for $k > 2t$.

6 Communication with Less Connectivity

In the model we have used in the paper so far, the network graph is k-connected in every round. It is natural to ask if this is required for our protocols to work. As we shall see, the answer is no, in some cases we can work with the weakest possible network assumption.

Let us first consider the case of reliable communication. We will use the concept of a dynamic path, which is taken from [MTD15]. Consider some sequence of network graphs as chosen by the adversary. Now, informally, if there is a path in round 1 from party P_i to party P_j and in round 2 we have one from party P_j to party P_k, we say there is a dynamic path from P_i to P_k; and this generalizes in the natural way to any number of rounds. We say the network is dynamically connected if there is a dynamic path from any party to any other party. If this is the case, then for a passive adversary, the simple flooding protocol will allow any party to multicast a message to anyone else. On the other hand, if there is no dynamic path from P_i to P_j, nothing P_i says can reach P_j.

For an active adversary, we have shown an efficient multicast protocol working in at most n rounds if we have k-connectivity in every round. A first observation is that since the protocol is based on flooding, it will clearly also work if we run for $m > n$ rounds and we have k-connectivity in at least n of the m rounds.

However, we can even work with the weakest possible network model as defined in [MTD15]. For a given set Ω of dynamic paths between nodes P_i and P_j, they define the dynamic min-cut of Ω as the minimal number of nodes one needs to remove, in order to cut all paths in Ω. Finally, the dynamic min-cut between P_i and P_j is defined as the min-cut of the set of *all possible* dynamic paths from P_i to P_j. It is shown that reliable communication from P_i to P_j with t active corruptions is possible if and only if the dynamic min-cut between P_i and P_j is larger than $2t$.

We can rephrase this in terms of the notions we define in this paper, namely for a set of dynamic paths Ω as above, we define a labelled graph G_Ω as we did in Section 4, where for each party P_a and each round there is a node labelled as P_a. If a path in Ω connects P_a to P_b in some round r, we put an edge between the node for round $r-1$ labelled as P_a and the one for round r labeled as P_b (also, each party is connected to itself in the next round). It is then straightforward to see that the labeled (integer) min-cut of G_Ω as defined in Section 4 equals the dynamic min-cut of Ω.

Now, since our actively secure multicast protocol works with labelled min-cuts, we can make a variant that will work making the minimal assumption that the dynamic min-cut between any pair of players is $> 2t$. The idea is to change the output condition such that the graph accompanying the message m' ($D_i[m']$) must have *integer* labelled min-cut greater than t – rather than the relaxed labelled min-cut that allows fractional values. Computing the non-relaxed labeled min-cut is not computationally efficient, but we still use only polynomial communication. This protocol therefore compares favorably to the protocol from [MTD15] for the same setting. That protocol tracks every individual path a message travels and so is exponential both in communication and computation.

Finally, we consider private communication. Observe that in both the passive and active case, we use one round to decide on a sufficiently connected graph according to which keys are shared pairwise between parties. Clearly, we could also run this phase for several rounds,

where in each round players attempt to share keys with other players. As long as the *union* of the network graphs in all of the rounds is sufficiently connected, this will be enough for the protocol to work. Once the keys have been established, reliable communication is used, so we only need that the network supports non-private communication in that last phase. We note that for passive security this means that we only need $k > t$ for the key sharing phase, and $k > 0$ (or a dynamically connected network) for the rest of the protocol.

References

DDWY93. Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. *J. ACM*, 40(1):17–47, 1993.

DF98. R.G. Downey and M.R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer New York, 1998.

DHKM16. Tridib Dutta, Lenwood S. Heath, V.S. Anil Kumar, and Madhav V. Marathe. Labeled cuts in graphs. *Theoretical Computer Science*, 648:34–39, 2016.

Dol82. Danny Dolev. The byzantine generals strike again. *Journal of algorithms*, 3(1):14–30, 1982.

DRTY23. Ivan Damgård, Divya Ravi, Daniel Tschudi, and Sophia Yakoubov. Secure communication in dynamic incomplete networks. In *4th Conference on Information-Theoretic Cryptography, ITC 2023*, 2023.

Kha79. L. G. Khachiyan. A polynomial algorithm in linear programming. *Dokl. Akad. Nauk SSSR*, 244:1093–1096, 1979.

MTD15. Alexandre Maurer, Sébastien Tixeuil, and Xavier Defago. Communicating reliably in multihop dynamic networks despite byzantine failures. In *2015 IEEE 34th Symposium on Reliable Distributed Systems (SRDS)*, pages 238–245, 2015.

SZ16. Gabriele Spini and Gilles Zémor. Perfectly secure message transmission in two rounds. In *TCC 2016-B*, 2016.