Secure Communication in Dynamic Incomplete Networks

Ivan Damgård[1], Divya Ravi[1], Daniel Tschudi[2], and Sophia Yakoubov[1⋆]

[1] Aarhus University, Denmark; {ivan, divya, sophia.yakoubov}@cs.au.dk
[2] Concordium, Zurich, Switzerland;

Abstract. In this paper, we explore the feasibility of reliable and private communication in *dynamic* networks, where in each round the adversary can choose which direct peer-to-peer links are available in the network graph, under the sole condition that the graph is k-connected at each round (for some k).

We show that reliable communication is possible in such a dynamic network if and only if $k > 2t$. We also show that if $k = cn > 2t$ for a constant c, we can achieve reliable communication with polynomial round and communication complexity.

For unconditionally private communication, we show that for a passive adversary, $k > t$ is sufficient (and clearly necessary). For an active adversary, we show that $k > 2t$ is sufficient for statistical security (and clearly necessary), while $k > 3t$ is sufficient for perfect security. We conjecture that, in contrast to the static case, $k > 2t$ is not enough for perfect security, and we give evidence that the conjecture is true.

Once we have reliable and private communication between each pair of parties, we can emulate a complete network with secure channels, and we can use known protocols to do secure computation.

⋆ 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).

Table of Contents

1 Introduction

In this paper, we study the feasibility of unconditionally secure communication (and hence multiparty computation) when parties communicate over an incomplete and dynamic network. More precisely, we assume a synchronous network with secure point-to-point channels where, in each round, only some of the point-to-point connections work, so the network is incomplete. Furthermore, the adversary can decide to change the set of active connections from one round to the next. We call this a *dynamic* incomplete network, in contrast to a *static* incomplete network, where the graph describing the active connections stays the same throughout the protocol.

The study of reliable communication on an incomplete network starts with the work of Dolev [Dol82], who showed that, for a static incomplete network where 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 $3t < n$. (A network is k-connected if it remains connected when one removes any set of less than k vertices. By Menger's theorem, this is equivalent to requiring that any pair of distinct nodes are connected by at least k disjoint paths.)

Later, Dolev et al. [DDWY93] showed that any two parties in a static incomplete network can communicate with perfect security (privacy and reliability) 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, so combining this with well-known feasibility results for MPC, one can conclude that, in a static network, $2t+1$-connectivity is necessary and sufficient for unconditionally secure MPC to be possible.

However, as mentioned, we are interested in what happens if the network is dynamic. To the best of our knowledge, very little work has been done in this direction. The problem was first considered in Mauer *et. al* [MTD15], who define a notion called *Dynamic Min-Cut* for two nodes a, b, denoted by $DynMinCut(a, b)$. They show that reliable (non-private) communication in a dynamic network from a to b is possible if and only if $DynMinCut(a, b) > 2t$. Intuitively, this condition makes a statement on how the network evolves over time, and says that it does so in a way that is "kind enough" to allow a message to travel from p to q on at least $2t+1$ disjoint paths. It is not surprising that this is the condition for reliable communication, given what was known about static networks. However, dynamic min-cut makes a rather complicated statement about the entire sequence of network graphs. It is natural to ask if there is a simple property we can require for each individual network graph that would then imply the $DynMinCut$-condition. This is a natural question from a theory point of view, but also a question one would ask in order to decide if secure communication

is feasible in a given application scenario. Finally, the work of Mauer *et. al* did not consider unconditionally private communication.

1.1 Our Contribution

In this paper, we aim to fill the gaps pointed out above. We introduce a model where an adversary chooses, in each round, a network graph to be used, under the sole condition that it is k-connected, for some k. This is a natural generalization of the static case and also a model that might be relevant in practice — consider, for instance, a mobile network where connections come and go, but where each possible connection is active with some probability p. Known results on random graphs indicate that if p is large enough, then the resulting graph is highly connected.

For a static network, it seems reasonable to assume that the parties know the topology of the network, while this is quite unrealistic for a dynamic network, where the absence on connections may be caused by movement of mobile devices, or equipment crashing. These are events that a party cannot predict locally. We therefore assume that honest parties do not know the network topology. In one version of the model, parties know their immediate neighbourhood, but our protocols work in the worst case where parties do not know which connections they have.

We show that reliable communication is possible in a dynamic network if and only if $k > 2t$ (note that we inherit impossibility results from the static case since the adversary could choose to keep the network graph constant). For unconditionally private communication, we show that for a passive adversary, $k > t$ is sufficient (and clearly necessary). For an active adversary, we show that $k > 2t$ is sufficient for statistical security (and clearly necessary), while $k > 3t$ is sufficient for perfect security. We conjecture that, in contrast to the static case, $k > 2t$ is not enough for perfect security, and we give evidence that the conjecture is true. As mentioned above, once we can emulate a complete network with secure channels, we can use known protocols to do secure computation.

Even though we can provide secure communication on dynamic networks, it is natural to expect that there is a performance penalty in going from static to dynamic networks: in the static case we can use a fixed set of paths for communication, while this clearly will not work in the dynamic case, as the adversary could block these paths. Intuition clearly suggests that one needs to try a large number of paths from sender to receiver to make sure something gets through.

We study a class of protocols where the main step is that the sender S tries to send data along a set of paths Paths, and where the protocol will be successful if the receiver R receives something on at least k disjoint paths. All the protocols we construct are in this class. We will say that Paths *is k-connected with respect to a family of graphs $\mathcal{G}$* if, for each graph $\mathsf{G} \in \mathcal{G}$, for any pair (S, R), there exist k disjoint paths between S and R, such that these paths are contained in both G and in Paths. Here, one should think of $\mathcal{G}$ as a sequence of network graphs chosen by the adversary under the constraints in our model, so we assume throughout that all graphs in $\mathcal{G}$ are k-connected. As short hand, we say that Paths *is k-connected* if it is k-connected with respect to *all* graph families (within our model). Paths being k-connected can be thought of as a condition saying that Paths is large enough, in comparison to the set of connections that adversary allows at any one time.

We show that if Paths is k-connected (with respect to the family of graphs chosen by the adversary), then our protocols terminate correctly in at most $L|\mathsf{Paths}|$ rounds, where L is the maximal length of paths in Paths and $|\mathsf{Paths}|$ is the cardinality of Paths. We can make sure that Paths is always k-connected by simply choosing Paths to be the set of all possible paths, which unfortunately is exponentially large in n.

However, if k is $\Omega(n)$, we can do much better: we show that in this case, among the k disjoint paths that exist in every round, there must be a large number of short (constant length) paths. More precisely, if $k = dn$ for a constant d, and we let Paths_c the set of paths of length c for some constant c, then Paths_c is $k' = d'n$-connected for $d' < d$, and by choosing c large enough, we can make d' be arbitrarily close to d. Moreover, $|\mathsf{Paths}_c|$ is polynomial in n for any constant c, so if we run our protocols with Paths_c as the target set of paths, their complexity will be polynomial in n.

This efficient version is almost as robust against corruptions as is possible: from the discussion above, it follows that the maximum number of corruptions that can be tolerated with connectivity k is roughly

$(k-1)/2$. So if k is a constant times n, it is also the case that $t = \alpha n$ for a constant α. Since we can get connectivity almost k with constant length paths, our efficient version can be designed to tolerate βn corruptions, for any $\beta < \alpha$.

Note that if we want to do MPC, one usually wants to tolerate $\Omega(n)$ corruptions, which implies that k must also be $\Omega(n)$, even for a static network.

	Corruption		Graph	Complexity	
Scheme	Type	Threshold	Connectivity	Rounds	Communication
Protocol 1	passive	$t < n$	1	n	$\mathcal{O}(n\|m\|)$
Protocol 2	active	$t < \frac{n}{2}$	$k > 2t$	LM	$\mathcal{O}(n^2 2^n \|m\|)$
Protocol 2 with constant-length paths	active	$t < \frac{n}{2}$	$k = cn > 2t$	$c\|\mathsf{Paths}'\| = \mathsf{poly}(n)$	$\mathsf{poly}(n)\|m\|$

Table 1: Reliable Communication Protocols for a message m, over k-connected path set Paths of cardinality M and with maximal path length L. The communication complexity is given per party per round. Paths' denotes a k-connected path set having paths of length at most c, where c is a constant.

		Corruption		Graph	Complexity	
Scheme	Security	Type	Threshold	Connectivity	Rounds	Communication
Protocol 3	perfect	passive	$t < n$	$k > t$	$LM + 2n$	$\mathcal{O}(Mn^2(\log M + \|m\|))$
Protocol 5	perfect	active	$t < \frac{n}{3}$	$k > 3t$	$3LM$	$\mathcal{O}(M^n n(\log M + \|m\|))$
Protocol 5 communication efficient variant	perfect	active	$t < \frac{n}{3}$	$k > 3t$	$\mathsf{poly}(LM)$	Per attempt same as Protocol 4. In the worst case overall same as Protocol 5.
Protocol 6	perfect	active	$t < \frac{n}{4}$	$k > 4t$	$3LM$	$\mathcal{O}(Mn^3 2^n(\log M + \|m\|))$
Protocol 4	statistical	active	$t < \frac{n}{2}$	$k > 2t$	$3LM$	$\mathcal{O}(n^3 2^n(\|m\| + \log M) + M(\|m\| + \lambda \cdot M))$

Table 2: Private Communication Protocols: The communication complexity is given per party per round.

1.2 Technical overview

The case of sending a public message reliably from S to R is relatively straightforward: S can send the same message on all paths in the pathset used. Here, as in all of our protocols, each copy of the message will be

accompanied by metadata that specifies on which path the message travelled. Once R has received something on sufficiently many disjoint paths, she can determine what the correct message is. This is basically the same protocol that was considered in [WW20] for a static and asynchronous network. However, the proof that it terminates correctly is completely different in our case.

Sending unconditionally private messages comes with new and bigger challenges. For the static case, Dolev *et. al* [DDWY93] designated a fixed set of k disjoint paths to be used for communicating between S and R. After this, they could abstract away the network and simply assume that S and R are connected by k channels where t of these are controlled by the adversary. This is the problem of *secure message transmission* that has been studied in many subsequent works. However, this abstraction cannot be used for a dynamic network: we cannot predict on which paths R will receive something. Moreover, corrupt parties may claim that they heard something on a path, even if the adversary's choice of network graphs actually did not allow transmission via that path. R has no obvious way to tell that such a claim is false — after all, with a different scheduling of network graphs, the claim might have been true.

We therefore need a new approach to sending private messages. For simplicity, we first sketch the idea for a passive adversary: we let S send independent randomness on sufficiently many different paths, and once R has received something on k disjoint paths, she can report to S the identity of these paths (but not the randomness she received). The reporting can be done using the protocol we already have for reliable public communication. If $t < k$, at least one random value that made it to R is unknown to the adversary, so S can derive a key (by XORing together all of the random values that made it), use that key to one-time pad encrypt the message, and send the resulting ciphertext in public.

This will not work for an active adversary, as we need to make sure the correct message is received. Our high-level strategy to solve this is as follows: if S sends data on a k-connected pathset, R will eventually receive data on $k-t$ disjoint paths (but not necessarily more, as t paths can be blocked by corrupt players). However, due to misreporting by corrupt players, R may have data delivered that claim to come from a large set of paths, containing many sets of $k-t$ disjoints paths. The idea is now to use reliable public communication to identify a sufficiently large set C of disjoint paths where R received *correct* data on *all* paths in C. We call such a set a *good* set. It turns out that if $k \geq 3t+1$ we can identify a good path set of size $2t+1$ with zero error probability by exploiting the fact that any set of at least $2t+1$ disjoint paths must contain a majority of paths with only honest players. This requires a lot of work, but if $k \geq 4t+1$, it can be done much more efficiently using error correction. If $k \geq 2t+1$ we can identify a good set of size $t+1$, except with negligible error probability, by using unconditionally secure authentication[3]. Now, from the data received via paths in C we can extract a value that can be used to one-time pad encrypt the secret message, as in the passively secure solution. We therefore get statistical security for $k \geq 2t+1$ and perfect security for $k \geq 3t+1$.

However, if the goal is perfect security and we only assume $k \geq 2t+1$, there are serious problems, and we conjecture that in fact perfect security cannot be achieved in this case. We give evidence for the conjecture: For connectivity $2t+1$, we construct an example scenario where the set of paths delivered to R contains several maximal sets of disjoint paths of size $k-t = t+1$. One of these sets contains only honest paths, but the others have the same number of honest and corrupt paths and may contain incorrect data. We show that R cannot perfectly decide which one is the all-honest set: for each choice there exists an adversarial strategy that would be consistent with that choice[4]. Now, consider any protocol which (as we do) would select one of these path sets and try to use the data received there and public communication to get a message across with perfect security. Any such protocol will fail with non-zero probability: if the wrong set is selected, we may have a situation where half the received values are known and/or manipulated by the adversary. Intuitively, if the receiver's output depends on only half the values (and the public communication) this may be the half the adversary knows and the output is not secret. If it depends on more than half the values, it must depend

[3] It may seem that authentication cannot be used here, as S needs to send a key to R, and this key must be sent privately and correctly, just like the message. We solve this apparent circularity by observing that S could send many keys on different paths and it is sufficient that one key makes it to R unseen by the adversary. See more details within.

[4] note that unconditionally secure authentication does not give perfect security and so does not help.

on values the adversary may have changed, and the output is not correct. We therefore conjecture that, in contrast to the static network case, $k > 3t$ is needed for perfect security.

1.3 Related Work

Construction	Assumptions	Corruption	Guarantees on Graph
auth. P2P [MTD15]	-	$t < \frac{n}{2}$	dynamic, dynamic min cut $> 2t$
auth. P2P [MTD15]	signatures	$t < \frac{n}{2}$	dynamic, dynamic min cut $> t$
auth. P2P [WW20]	-	$t < \frac{n}{3}$	static, connectivity $k > 2t$
broadcast [WW20]	-	$t < \frac{n}{3}$	static, connectivity $k > 2t$

Table 3: Related Work

Mauer *et. al* [MTD15] consider the setting closest to our own. They identify the minimal and necessary condition for authenticated peer-to-peer communication; that is, there should exist $2t+1$ (or $t+1$, assuming digital signatures) disjoint paths from the sender to the receiver *over time*. However, they do not consider the invariants that we need in our graph in order for this condition to be met. We address this gap in our work.

Wang and Wattenhofer [WW20] consider *static* incomplete networks (and additionally consider *asynchrony*, where messages can take arbitrarily long to traverse a link in the graph). We build on one of their protocols, adapting it to our dynamic, synchronous setting.

2 Preliminaries

2.1 Model

We consider a set $\mathcal{P} = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$ of n parties.

Corruption. A central adversary corrupts at most t parties. The 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 has full control over the behavior of corrupted parties.

Communication network. Parties communicate over a *dynamic incomplete network* of secure (private and authentic) synchronous channels. In each round r (a.k.a. time-step), parties can communicate over the network graph G_r that has been selected by the adversary from a publicly-known family of graphs $\mathcal{G}$. The graph family $\mathcal{G}$ models the guarantees for honest parties, e.g., with respect to graph connectivity. There are three possibilities for modelling the network adversary:

- A *static* adversary is required to decide / commit to the set of graphs beforehand i.e. before the protocol begins.
- An (non-rushing) *adaptive* adversary can choose the graph for round $r+1$ at the end of round r.
- A *rushing adaptive* adversary can first wait to see through what network edges the messages were attempted to be sent during round r, before determining the graph for round r.

In this paper, we consider a *rushing adaptive* network adversary.

We assume that honest parties are *oblivious* on the communication graph of a given round. That is, they only know the overall family $\mathcal{G}$, but not the actual G_r. So, in a protocol they may *attempt* to use any channel in their neighborhood of $\bar{\mathsf{G}} = \bigcup_{\mathsf{G}\in\mathcal{G}} \mathsf{G}$ whereas only channels in the unknown G_r will actually transmit. Honest parties do not learn which of their outgoing transmissions were successful.

Communication Complexity. In the setting with oblivious honest parties, every *attempted* use of a communication channel will count towards the communication complexity even if the actual transmission fails. For example, if an honest party tries to send a bit to every other party, this will count as n bits of communication irrespective of outcome.

Future Work: Other Models. For static or adaptive adversaries we could also consider *aware* honest parties, who are given G_r at the beginning of the round. We therefore assume that they will only use channels within G_r which are guaranteed to work. Aware honest parties will not attempt to send messages on channels which are not available, leading to better communication complextiy.

In the rushing adaptive setting, we could also consider looking at *retroactive awareness*, where honest parties are notified about successful transmissions.

2.2 Building Blocks

In this section we define the building blocks necessary for our protocols.

Threshold Secret Sharing Scheme A t-out-of-n secret sharing scheme allows a party to "split" a secret into n shares that can be distributed among different parties. To reconstruct the original secret x at least $t+1$ shares need to be used.

Definition 1 (Secret Sharing). *A t-out-of-n secret sharing scheme is a tuple of efficient algorithms* $(\texttt{share}, \texttt{reconstruct})$ *defined as follows.*

- *The randomized algorithm* `share` *takes as input a secret $x \in \mathbb{F}$ and outputs a set of n share, i.e.,*

$$(s_1, \ldots, s_n) \in \mathbb{F}^n \xleftarrow{\$} \texttt{share}(x).$$

- *The reconstruct algorithm* `reconstruct` *takes as input a vector of at least $t+1$ shares and outputs either the secret x, or outputs $\perp$, i.e.,*

$$\{x, \perp\} \leftarrow \texttt{reconstruct}\left(\{s_i\}_{i\in S\subseteq[n],|S|>t}\right).$$

We require the following properties of a t-out-of-n secret sharing scheme:

Perfect Correctness. *The perfect correctness property requires that the shares of a secret x should always reconstruct to x. More formally, a secret sharing scheme is* perfectly correct *if for any secret x, for any subset $S \subseteq [n], |S| > t$,*

$$\mathsf{P}\left[x = x' \;:\; \begin{array}{c}(s_1, \ldots, s_n) \xleftarrow{\$} \texttt{share}(x),\\ x' \leftarrow \texttt{reconstruct}(\{s_i\}_{i\in S})\end{array}\right] = 1,$$

where the probability is taken over the random coins of `share`*. Moreover, if a negligible error probability is allowed, we simply say that the scheme is correct.*

Privacy: *The privacy property requires that any combination of up to t shares should leak no information about the secret x. More formally, we say that a secret sharing scheme is* private *if for any x, and for any set $\mathbb{A} \subseteq \{1, \ldots, n\}$, $|\mathbb{A}| \leq t$ the distribution of the set $\{s_i\}_{i\in\mathbb{A}}$ is statistically independent of x.*

Instantiation. In our constructions, we use the Shamir's threshold secret sharing scheme [Sha79]. We give a brief description of this scheme below. Informally, the shares output by the algorithm `share` correspond to the set of evaluations on n different points of a t-degree polynomial (whose coefficients are chosen at random from a finite field $\mathbb{F}$) with constant term as its secret. The algorithm `reconstruct` uses Lagrange interpolation to identify the t-degree polynomial that is consistent with the set of shares in order to return the constant term as the secret ($\perp$ is returned if no such polynomial exists).

3 Reliable Communication

In this section we describe protocols for reliable communication in our model, which we summarize in Table 1.

3.1 Passive Corruptions

In the passive case a simple flooding protocol where parties echo the first received message can be used.

Protocol 1: $\mathsf{Flood}(\mathcal{G}, \mathsf{S}, \mathsf{R}, m)$

Let $\bar{\mathsf{G}} = \bigcup_{\mathsf{G} \in \mathcal{G}} \mathsf{G}$. The sender S has message m as input. The protocol runs for n rounds:

- In each round the sender S sends m to all neighbors in $\bar{\mathsf{G}}$.
- Once a party $\mathsf{P} \in \mathcal{P} \setminus \{\mathsf{S}, \mathsf{R}\}$ receives the first message m' from any neighbor, it will send m' to all neighbors in $\bar{\mathsf{G}}$ in all subsequent rounds.
- The sender will output the first received message m' at the end of the protocol.

Fig. 1: Simple flooding protocol, secure against $t < n$ passive corruptions in a connected network.

Lemma 1. *Protocol 1 allows S to reliably send a message m to R in the presence of a rushing adaptive network adversary that passively corrupts $t < n$ parties. The protocol runs for n rounds and in each round a party sends $\mathcal{O}(n|m|)$ bits.*

Proof. Correctness. Corruption is passive, so all parties which got a message actually get the sender's message m. Let H denote the set of nodes who have already heard the message, and D denote the set that have not. Since the actual communication graph chosen by the adversary is connected, there must be at least one edge between H and D. Any party (apart from the receiver) that knows the message will try to send it to all potential neighbors in every round. In a given round, either the receiver already knows the message, or some party will learn the message in this round; after at most n rounds, the message must have reached the receiver.

Complexity. In each round a party sends at most one message of size $|m|$ to at most n. The round complexity follows from the protocol description. □

3.2 Active Corruptions

For the active corruption setting, we assume that the honest parties' knowledge about $\mathcal{G}$ comes in the form of a set Paths of possible paths between sender S and receiver R. We assume that the paths have length of at most L and $|\mathsf{Paths}| = M$.

In Protocol 2 we adapt the peer-to-peer protocol from [WW20] to allow for reliable communication in the presence of a rushing adaptive adversary.

Protocol 2: AuthenticatedP2P(Paths, S, R, m)

The set of possible paths Paths from S to R is public knowledge. The sender S has message m as input. The protocol runs for LM rounds:

- In each round S sends (m, S) to each neighbor that is on a path in Paths.
- Upon receiving (m', π) from a neighbor P_j, party $\mathsf{P}_i \in \mathcal{P} \setminus \{\mathsf{S}, \mathsf{R}\}$ does the following:
  1. If $i \in \pi$ or $j \notin \pi$, P_i ignores the message.
  2. Otherwise the party will in all subsequent rounds send $(m, \pi \cup \{i\})$ to all neighbors that are next nodes on paths in Paths for which $\pi \cup \{i\}$ forms a prefix.
- Upon receiving (m', π) from a neighbor P_j, R stores (m', π) unless $\mathsf{R} \in \pi$ or $j \notin \pi$.

At the protocol end, if party R has more than t stored values (m', π) with disjoint paths in Paths, R outputs m'.

Fig. 2: The modified authenticated P2P protocol of Wang and Wattenhofer [WW20]

Theorem 1. *The authenticated P2P protocol of Wang and Wattenhofer [WW20]* modified to only accept messages received over $t+1$ disjoint paths *as described in Protocol 2 achieves reliable communication between* S *and* R *in the presence of a rushing adaptive network adversary that actively corrupts at most t parties as long as the network has connectivity $k > 2t$ in every round. The protocol runs for LM rounds.*

We approach Theorem 1 by first introducing Lemma 2 and Lemma 3.

Lemma 2. *If the network has connectivity k in every round, then there are k disjoint paths from the sender to the receiver* over time.

Proof (of Lemma 2). Since there are a finite number of sets of k disjoint paths from sender S to receiver R, and the adversary has to choose one such path set in every round, it follows that after a finite number of rounds one path set will have been chosen sufficiently many times that the message had the opportunity to traverse all of its paths. □

The following lemma is useful for analyzing the round complexity of our constructions.

Lemma 3. *Consider a path set* Paths *which is k-connected and a sender* S *who sends a message along each path of* Paths*. Then, the receiver* R *would receive the messages from a set of k disjoint paths in at most LM rounds, where L is the maximal length of paths in* Paths *and M is the cardinality of* Paths*.*

Proof (of Lemma 3). Let $\mathcal{C}_r$ be the set of paths along which S's message has reached R at round r, and let Paths_r be the set of paths between S and R chosen in that round. In the first round, the message advances along at least k edges. In every subsequent round, if $\mathcal{C}_r$ contains a set of k disjoint paths, then we are done. Otherwise, it must be that $|\mathsf{Paths}_r \setminus \mathcal{C}_r| > 0$, so the message must advance along at least one edge it has not advanced along before. Since the total number of edges across all paths in Paths is at most LM, we can be certain that we are done once the message advances along all of them, we must be done after at most LM rounds. □

We can now give the proof for Theorem 1.

Proof (of Theorem 1). Correctness Assume S and R are honest.

First consider the network without the corrupted nodes. This network is guaranteed to be $t+1$ connected in every round (the overall network is at least $2t+1$ connected). It follows from Lemma 2 and Lemma 3

that the message m sent by the sender will arrive at the receiver R in at most LM rounds via at least $t+1$ disjoint paths. This makes m a valid output.

Next, consider any message $m' \neq m$. If R gets a tuple (m', π') there must be a corrupt P_j that sent (m', π'') for $\pi'' \subset \pi'$ to an honest party. Honest parties only forward a tuple if the sender of the tuple is in the path information. This implies that $j \in \pi''$ and thus $j \in \pi'$. There are at most t corrupted parties, so R will receive m' on at most t disjoint paths. Hence m' can never be a valid output. This means m is the unique output for R after LM rounds.

Round complexity follows from the protocol description. □

Analyzing the communication complexity of Protocol 2. We observe that the maximum communication complexity of any party in a round r is $\mathcal{O}(2^n n^2 |m|)$. This is because a party in round $r-1$ may have received messages from various paths represented by different sets (which we refer to as metadata). Since there could be 2^{n-1} such sets (all possible sets that exclude this party), we can infer that a party has to communicate a message of size $|m|$ and metadata of size at most $\mathcal{O}(n2^n)$ (n bits are sufficient to represent one set) to each of her neighbours (which are at most $\mathcal{O}(n)$), which adds up to a communication complexity of $\mathcal{O}(2^n n(n+|m|))$.

Lastly, we point that the computation done by the receiver R to check if she has received values along $t+1$ disjoint paths would involve $\mathcal{O}(LMn)$ computation per round. This can be done by the receiver R as follows: consider the graph $\mathcal{C}_r$ formed by the set of paths containing the same message along which S's message has reached R at round r. Apply the Ford-Fulkerson algorithm [FF56] (with complexity $O(|E|n)$, where $|E|$ denotes the number of edges and n denotes the number of nodes) on $\mathcal{C}_r$ to check if it is $t+1$ connected. If yes, output this common message. Else, try with set of paths containing a different message. If none of them succeed, try again in the next round.

Efficiency In this section, we discuss a special case when this authenticated communication protocol becomes *efficient* (has complexity polynomial in n). For this, suppose there are k *short* paths from sender to receiver at each round. We let c denote the upper bound on the length of such a "short" path. First, we analyze the case where c is any constant (Lemma 4). Finally, we argue that for sufficiently large k, we are guaranteed to have many short paths (Lemma 5).

Lemma 4. *If S is connected to R via k disjoint paths of length at most c in every round (for a constant c), then the authenticated P2P protocol of Wang and Wattenhofer [WW20] described in Figure 2 runs in at most a polynomial (in n) number of rounds.*

Proof (of Lemma 4). Let Paths' be a set of paths with length at most c. Protocol 2 on Paths' runs in $c|\mathsf{Paths}'|$ rounds as shown in *Theorem* 1. So it remains to analyze the size of $|\mathsf{Paths}'|$. Let $|\mathsf{Paths}_\ell|$ denote how many paths of length at most ℓ exist from S to R in Paths'. We have $|\mathsf{Paths}_1| \leq 1$ and for $\ell > 1$, $|\mathsf{Paths}_\ell| \leq |\mathsf{Paths}_{\ell-1}| + \frac{(n-2)!}{((n-2)-(\ell-1))!}$[5]. We have $|\mathsf{Paths}'| = |\mathsf{Paths}_c|$ which is polynomial in n for constant c. We can thus conclude that the protocol terminates in at most a polynomial (in n) number of rounds. □

Ensuring Enough Paths of Constant Length We have shown that if we have k disjoint paths of constant length c at every timestep, S's message will reach R along k disjoint paths in polynomial time. Of course, k-connectivity *over constant length paths* is in general a much stronger assumption than k-connectivity. However, we show here that (loosely speaking) if we have many disjoint paths, this implies that at least some of them must be short, and in particular, if we have k-connectivity for k linear in n, this implies we have k'-connectivity over constant length paths, where k' can be very close to k.

Lemma 5. *Say we have k disjoint paths in a graph on n nodes. For any L, at least $k - \frac{n}{L+1}$ of these have length at most L.*

[5] where the latter term is the number of ways of choosing $\ell - 1$ intermediate nodes among $n-2$ nodes (excluding S and R).

Proof. Let $k_{\leq L}$ and $k_{>L}$ be the number of paths of length at most L and greater than L, respectively, among the k given ones. The subset of paths of length greater than L contain at least $(L+1)k_{>L}$ distinct nodes, so we have $(L+1)k_{>L} \leq n$, implying $k_{>L} \leq \frac{n}{L+1}$. Since clearly $k = k_{\leq L} + k_{>L}$, the lemma follows. □

An immediate consequence of this is that if $k = dn$ for a constant fraction d, we are guaranteed to have at least $n(d - \frac{1}{L+1})$ disjoint paths of length L. By choosing a large enough but constant L, we can have $\Omega(n)$-connectivity over constant length paths; in fact that underlying constant can be chosen arbitrarily close to d.

In particular, assume we want to tolerate a constant fraction of corrupted players, as is standard in MPC. We know that for t active corruptions and even for a static network, we must always have connectivity at least $2t + 1 = cn$ (for a constant c); otherwise broadcast is impossible. Therefore, in the dynamic case, if we ask for slightly larger connectivity, namely $k = dn$ for any $d > c$, the above lemma allows us to assume $2t + 1$-connectivity over constant length paths, implying that our protocols will run in polynomial time.

We state the formal theorem below.

Theorem 2. *The modified authenticated P2P protocol of Wang and Wattenhofer [WW20] as described in Protocol 2 is an* efficient reliable communication protocol *(i.e. runs in polynomial time and with polynomial complexity) in the presence of t active corruptions as long as the network has connectivity $k = dn$ for any $d > c$ (where c is a constant) in every round (using the set of all paths of length at most c as* Paths*).*

4 Private Communication

In this section, we look at the feasibility of establishing a secure channel between sender S and receiver R. The knowledge on $\mathcal{G}$ is given as a set Paths of bidirectional paths between S and R. The set is of size M and paths in the set are of length of at most L.

As a starting point, assume that S and R somehow have shared secret randomness o. Given the results from the previous section, they could establish a reliable channel to securely transmit message m as $c = m+o$. This reduces the problem of secure communication to establishing shared randomness between S and R. At a first glance this seems as difficult as the original problem. However, we note that there is a slight difference i.e. this value o (unlike m) need not be a 'fixed' value pre-determined by S but can be dynamically determined during the protocol.

This is exactly what we exploit in our upper bounds that have the following common approach: S chooses a set of random values, one for each path in Paths. Next, R upon receiving 'sufficiently many' random values reports back to S which paths she received information from. For this, R acts as a sender and can rely on a reliable communication protocol. This is because while we may want to hide the random values along paths that the adversary does not have access to (i.e. the paths that comprise of only honest nodes), there is no harm in revealing to the adversary the identity of the paths R received information from (as these paths were in fact determined by the dynamic adversary). Given this path information both S and R can compute o from the randomness sent along those paths. Finally, S can mask the actual m with o and send it over a reliable channel.

For simplicity, we assume in the following that $m \in \mathbb{F}$.

4.1 Passive Corruption

In this section, we present a protocol that constructs a secure channel given that the graph is at least $t + 1$-connected.

Consider plugging in the above common approach in a network with connectivity $k > t$ where the dynamic adversary corrupts up to t nodes passively. We are guaranteed that R would receive random values from $(t+1)$ disjoint paths, among which the adversary has access to at most t of them (because in the worst case, there could be one corrupt node in each of the t disjoint paths). These $(t + 1)$ random values could simply be viewed as an additive sharing of the shared randomness o that remains private from this passive adversary. Tying this with the above outlined approach, R would reliably communicate the identity of these

Protocol 3: $\Pi^{\mathsf{prv}}_{\mathsf{perf,sh}}(\mathsf{Paths}, \mathsf{S}, \mathsf{R}, m)$

The set of possible paths Paths is public knowledge. The sender S has message m as private input.

Randomness Generation For ML rounds the parties do the following:

- For each path $p \in \mathsf{Paths}$ the sender S:
  1. Sample randomness $r_p \in \mathbb{F}$.
  2. In each time step send $m_p = (p, r_p)$ to the first node on the path p until the phase is complete.
- Once intermediate node P_i receives the first message $m_p = (p, r_p)$ on path p, it will echo m_p to the next node on the path in each of the subsequent time steps until the phase is complete.
- The receiver node R initializes sets $\mathsf{RecPaths} = \emptyset$ and sets $\mathsf{RecPaths} = \mathsf{RecPaths} \cup \{p\}$ upon receiving $m_p = (p, r_p)$ for any $p \in \mathsf{Paths}$.

Afterwards, the receiver define $\mathsf{GoodPaths} \subseteq \mathsf{RecPaths}$ as a set of $t+1$ *disjoint* paths and send $\mathsf{GoodPaths}$ to S using an instance of Protocol 1. This concludes the randomness generation phase.

Secure Communication The parties do the following:

1. Both S and R (locally) compute $o = \Sigma_{p \in \mathsf{GoodPaths}} r_p$.
2. S sends $c = m + o$ to R using an instance of Protocol 1.
3. R outputs $m = c - o$.

Fig. 3: Perfectly-secure private communication protocol against $t < n$ passive corruptions in a network with connectivity $k > t$.

$(t+1)$ disjoint paths to S, allowing S to compute o and reliably communicate the masked secret $c = m + o$. The formal description of the protocol is given as Protocol 3.

Theorem 3. *Protocol 3 is perfectly secure protocol that allows* S *to securely send* m *to* R *in the presence of a rushing adaptive adversary that passively corrupts at most* t *parties, as long as the network has connectivity* $k > t$*. The protocol runs for* $LM + 2n$ *rounds and in each round a party sends* $\mathcal{O}(Mn^2(\log M + |m|))$ *bits.*

Proof. Correctness: As the graph is at least $t+1$ connected, receiver R will have received randomness over at least $t+1$ disjoint paths after LM rounds (cf. Lemma 3). This makes $\mathsf{GoodPaths}$ well defined. Correctness therefore follows by the correctness of Protocol 1 (cf. Lemma 1) and the correctness of one time pad encryption.

Privacy: To argue privacy, we note that the adversary has access to the random values corresponding to at most t paths among the $(t+1)$ disjoint paths constituting $\mathsf{GoodPaths}$. The property of additive secret sharing guarantees that o remains perfectly hidden from the adversary. It now directly follows from the security of the one-time pad encryption the adversary does not learn m from c alone.

Complexity: Protocol 1 has a round complexity of n. This implies a round complexity of $LM + 2n$. Lastly, we analyze the communication complexity. While sending the randomness forward, the complexity per party in each round is $\mathcal{O}(M(\log M + |m|))$. Observe that $\mathsf{GoodPaths}$ can be encoded in $\mathcal{O}(n \log M)$ bits, so the per party per round complexity of both instances of Protocol 1 is bounded by $\mathcal{O}(n^2(\log M + |m|))$. This gives an overall (loose) bound of $\mathcal{O}(Mn^2(\log M + |m|))$. □

4.2 Active Security

In this section, we provide protocols for secure communication in the presence of an adversary that actively corrupts parties.

Statistical Security with $k > 2t$ The above construction (Protocol 3) does not withstand active corruptions as the active adversary could tamper with the random values along the paths in $\mathsf{GoodPaths}$ where there is an actively corrupt node. This would lead to R determining an incorrect random one-time pad (o) i.e. different than the one computed by S; resulting in R obtaining the wrong secret. Further, the adversary could also tamper with the path information (i.e. the sequence of nodes forming the path); this may potentially lead R to wrongly believe that certain values are coming from 'disjoint' paths when in fact they are not.

To detect cheating of the above type in the statistical setting, one could authenticate the random values (using information theoretic mac) and ensure that R verifies each potentially tampered random value accompanied by its mac using the corresponding verification key determined by the honest S. However, for this to work, we should make sure that the verification key remains unknown to the adversary and untampered until it reaches R! We resolve this seemingly circular issue in the following way: To authenticate a random value, say r_p along a path p, S generates a mac using a different verification key for every path disjoint from p. R accepts r_p only if its macs verify against at least t paths that are mutually disjoint and also disjoint to p. The idea is that at least one of these t paths would comprise of only honest nodes and the verification keys sent along this all-honest path would be untampered and unknown to the adversary. (Note that if the value along p is tampered, then there must be at least one corrupt node already in p; so there can be at most $t-1$ paths disjoint to p that could be influenced by the adversary.) This completes the high-level overview of the protocol, whose formal details are described in Protocol 4.

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

The set of possible paths Paths is public knowledge. The sender S has message m as private input.

Randomness Generation For ML rounds the parties do the following:

- For each path $p_i \in \mathsf{Paths}$ the sender S:
  1. Sample randomness $r_{p_i} \in \mathbb{F}$ and for each disjoint path p_j sample key o_{p_i,p_j} and corresponding mac mac_{p_i,p_j}. Let $\mathcal{K}_{p_i} = \{o_{p_j,p_i}\}$ and $\mathcal{M}_{p_i} = \{\mathsf{mac}_{p_i,p_j}\}$ the set of macs.
  2. In each time step send $m_{p_i} = (p_i, r_{p_i}, \mathcal{M}_{p_i}, \mathcal{K}_{p_i})$ to the first node on the path p_i until the phase is complete.
- Once intermediate node P receives the first message $m_p = (p, r_p, \mathcal{M}_p, \mathcal{K}_p)$ on path p, it will echo m_p to the next node on the path in each of the subsequent time steps until the phase is complete.
- The receiver node R initializes sets $\mathsf{RecPaths} = \emptyset$ and sets $\mathsf{RecPaths} = \mathsf{RecPaths} \cup \{p\}$ upon receiving $m_p = (p, r_p, \mathcal{M}_p, \mathcal{K}_p)$.

Afterwards the receiver initializes $\mathsf{GoodPaths} = \emptyset$ and does the following for each $p_i \in \mathsf{RecPaths}$:

1. Check if $\mathsf{RecPaths}$ contains a set Paths' of at least t path disjoint from p_i such that for each $p_j \in \mathsf{Paths}'$ the key o_{p_i,p_j} (received via p_j) confirms the mac mac_{p_i,p_j} on r_{p_i} (both received via p_i).
2. If the above check holds (and $|\mathsf{GoodPaths}| < t$) set $\mathsf{GoodPaths} = \mathsf{GoodPaths} \cup \{p\}$.

Then R sends $\mathsf{GoodPaths}$ to S using an instance of Protocol 2. This concludes the randomness generation phase.

Secure Communication As in Protocol 3 the message is sent one-time padded except that Protocol 2 is used for the reliable communication channel.

Fig. 4: Statistically-secure private communication protocol against $t < n$ active corruptions in a network with connectivity $k > 2t$.

Theorem 4. *Protocol 4 is a statistically-secure protocol that allows* S *to securely send* m *to* R *in the presence of a rushing adaptive adversary that actively corrupts at most* t *parties, as long as the network has connectivity*

$k > 2t$. *The protocol runs for* $3LM$ *rounds and in each round a party sends* $\mathcal{O}(n^3 2^n(|m| + \log M) + M(|m| + \lambda \cdot M))$ *bits.*

Proof. Correctness: By Theorem 1 sender and receiver will agree on $\mathsf{GoodPaths}$ and the receiver will receive the sender's c. So correctness follows if sender and receiver agree on o. This is the case if R actually got the sender's randomness for each path in $\mathsf{GoodPaths}$. So assume there must exist at least one path, say p_i such that R obtained $r'_{p_i} \neq r_{p_i}$ but $p_i \in \mathsf{GoodPaths}$. This can occur only if p_i contains at least one corrupt node on behalf of which the adversary tampered with the random value and potentially its set of accompanying macs. However, since $p_i \in \mathsf{GoodPaths}$, it must hold that the macs verified against verification keys received along t other disjoint paths, which must include at least one verification key that was untampered and unknown to the adversary. This is because there can be at most $t-1$ corrupt nodes along these t disjoint paths; therefore there must be a path containing all honest nodes along which the verification key was correct (i.e. the same as chosen by S). It now follows directly from the unforgeability property of the mac that the (potentially) adversarial chosen mac can verify against this verification key successfully for $r'_{p_i} \neq r_{p_i}$ only with negligible probability. This implies statistical correctness.

Privacy: Privacy follows by the same argument as in the proof of Theorem 3.

Complexity: Protocol 2 has a round complexity of LM. This implies a round complexity of $3LM$. Lastly, we analyze the communication complexity. The two instances of Protocol 2 have a communication complexity of $\mathcal{O}(n^3 2^n(|m| + \log M))$. The messages m_p have a complexity of $\mathcal{O}(|m| + \lambda \cdot M)$ as the mac and key sets are $\mathcal{O}(\lambda \cdot M)$. So in the first LM rounds a party communicates $\mathcal{O}(M(|m| + \lambda \cdot M))$ bits per round. This totals upto communication complexity of $\mathcal{O}(n^3 2^n(|m| + \log M) + M(|m| + \lambda \cdot M))$. □

Perfect Security with $k > 3t$. In the above construction, settling for statistical security allowed us to use authentication tools to detect cheating. We now analyze how to achieve perfect security, where it becomes more challenging to deal with the adversary tampering with the random values sent across the paths.

To handle this, we make S send redundant information in a way that allows R to detect such misbehaviour. Instead of using sum sharing, we rely on threshold sharing in the following way: We let R report back $(2t+1)$ disjoint paths (instead of $t+1$ as before). Next, we make S compute a threshold sharing (with threshold t) of the message m and mask the shares (instead of the message directly) using the $(2t+1)$ random values corresponding to paths that were reported by R. Once these $(2t+1)$ masked shares are reliably communicated to R, R can retrieve the shares and attempt to reconstruct the secret. The main idea is that, unlike before, R can now check that she has the 'correct' secret by checking that all the $(2t+1)$ shares lie on a t-degree polynomial. This is because, the set of shares will comprise of $(t+1)$ untampered shares (that were masked with random values along the paths that comprised only of honest nodes) which suffice to uniquely determine the t degree polynomial. If the check fails, we make R retry with another set of $(2t+1)$ disjoint paths until she finds one that verifies. To accommodate for this, we assume larger connectivity i.e. $k > 3t$. This completes the high-level idea of the protocol, which is formally described in Protocol 5. We state the formal theorem below.

Theorem 5. *Protocol 5 is a perfectly-secure protocol that allows* S *to securely send* m *to* R *in the presence of a rushing adaptive adversary that actively corrupts at most* t *parties, as long as the network has connectivity* $k > 3t$. *The protocol runs for* $3LM$ *rounds and in each round a party sends* $\mathcal{O}(nM^n(M + |m|))$ *bits.*

Proof. Correctness: By Theorem 1 sender and receiver will agree on $\mathsf{GoodPaths}$ and thus on $\mathcal{W}$. This also hold for the set of one-time padded shares $\{c_{p_i}^{\mathsf{Paths}'}\}$.

By Lemma 2 the set $\mathcal{W}$ must contain at least $2t+1$ disjoint paths that comprise of only honest nodes. So, there must exist at least one candidate set Paths' which will lead would lead to the correct output of m. On the other hand any set Paths' that leads R to output m' contains at least $t+1$ honest paths (This is because at most t paths in Paths' could be such that it has a corrupt node.). The adversary cannot tamper with the random values or path information across these $(t+1)$ paths, therefore we are guaranteed that $(t+1)$ among the $(2t+1)$ shares retrieved by R must indeed be correct. Since these suffice to uniquely determine the t-degree polynomial that S used for Shamir sharing, it follows that the message reconstructed by R must be correct.

Protocol 5: $\Pi^{\mathsf{prv}}_{\mathsf{perf,mal}}(\mathsf{Paths}, \mathsf{S}, \mathsf{R}, m)$

The set of possible paths Paths is public knowledge. The sender S has message m as private input.

Randomness Generation For ML rounds the parties do the following:

- The sender S does the following:
  1. For each set $\mathsf{Paths}' \subset \mathsf{Paths}$ such that the paths in Paths' are disjoint and $|\mathsf{Paths}'| = 2t+1$ the receiver samples for each $p \in \mathsf{Paths}'$ random value $r_p^{\mathsf{Paths}'}$ [a]. For any $p \in \mathsf{Paths}$ denote by $r_p = \{r_p^{\mathsf{Paths}'}\}$ the set of all sampled random values.
  2. In each time step send $m_p = (p, r_p)$ to the first node on the path p until the phase is complete.
- Once intermediate node P_i receives the first message $m_p = (p, r_p)$ on path p, it will echo m_p to the next node on the path in each of the subsequent time steps until the phase is complete.
- The receiver node R initializes sets $\mathsf{RecPaths} = \emptyset$ and sets $\mathsf{RecPaths} = \mathsf{RecPaths} \cup \{p\}$ upon receiving $m_p = (p, r_p)$ for any $p \in \mathsf{Paths}$.

Afterwards, the receiver R sets $\mathsf{GoodPaths}$ to the maximal set of pairwise disjoint paths in $\mathsf{RecPaths}$. This can be done using the Ford-Fulkerson Algorithm. Then R sends $\mathsf{GoodPaths}$ to S using an instance of Protocol 2. This concludes the randomness generation phase.

Secure Communication The message m is sent as follows.

1. Let $\mathcal{W}$ denote the set of all subsets $\mathsf{Paths}' \subseteq \mathsf{GoodPaths}$ such that $|\mathsf{Paths}'| = 2t+1$ For each $\mathsf{Paths}' \in \mathcal{W}$:
   - The sender computes shamir-sharing of the message m with threshold t as $(s_1^{\mathsf{Paths}'}, \ldots, s_{2t+1}^{\mathsf{Paths}'}) \leftarrow \texttt{Shamir.share}(m)$.
   - Set $c_{p_i}^{\mathsf{Paths}'} = s_i^{\mathsf{Paths}'} + r_{p_i}^{\mathsf{Paths}'}$ for each $p_i \in \mathsf{Paths}'$, where $r_{p_i}^{\mathsf{Paths}'}$ is the appropriate random mask (i.e. the random value that was chosen for path p_i corresponding to set Paths').
2. Sender S sends $\{c_{p_i}^{\mathsf{Paths}'}\}_{p_i \in \mathsf{Paths}', \mathsf{Paths}' \in \mathcal{W}}$ to R using an instance of Protocol 2.
3. For each $\mathsf{Paths}' \in \mathcal{W}$ the receiver R:
   - Recovers the shares as $s_i = c_{p_i}^{\mathsf{Paths}'} - r_{p_i}^{\mathsf{Paths}'}$ for $p_i \in \mathsf{Paths}'$
   - Checks if $m' \leftarrow \texttt{reconstruct}(\{s_i\}_{i \in [2t+1]})$ results in $m' \neq \perp$ (this step essentially checks if all the shares lie on a t degree polynomial). If yes, the receiver outputs m'.

[a] We assume that all possible sets Paths' are lexicographically ordered. Abusing notation, when Paths' is used in superscript, it refers to the appropriate index of Paths' based on this ordering.

Fig. 5: Perfectly-secure private communication protocol against $t < n$ active corruptions in a network with connectivity $k > 3t$.

Privacy: Note that for each candidate set $\mathsf{Paths}' \subseteq \mathcal{W}$, the adversary knows at most t shares, namely, those that were masked using random values corresponding to the (at most) t paths in Paths' that the adversarial nodes were a part of. Privacy of the secret now follows directly from the privacy guarantee of threshold sharing and the security of one-time pad encryption (which holds as each random value is used for encryption at most once).

Complexity: The round complexity follows analogous to the proof of Theorem 4. Lastly, we analyze the communication complexity. The complexity is dominated by randomness generation phase. There are fewer than M^{2t+1} sets Paths' such that $|\mathsf{Paths}'| = 2t+1$. Since there are $2t+1$ paths p in each Paths', the sender picks fewer than $(2t+1)M^{2t+1}$ random values $r_p^{\mathsf{Paths}'}$. Each path can be represented using M bits, and each random value $r_p^{\mathsf{Paths}'}$ can be represented using $|m|$ bits; so, the communication per round per party is $\mathcal{O}((2t+1)M^{2t+1}(M+|m|) = \mathcal{O}(nM^n(M+|m|)$. □

Improving the expected complexity: In the above construction, the primary communication bottleneck is due to the fact that S has to account for all possible path sets of size $2t+1$. We propose the following modification to improve the expected communication complexity: Similar to Protocol 4, S could choose just one random value per path, and augment these values with information-theoretic MACs. Accordingly R could include a path p in $\mathsf{GoodPaths}$ only if t of the MACs have verified successfully using verification keys sent across t disjoint paths (that are disjoint from p as well). Once $\mathsf{GoodPaths}$ comprises of $2t+1$ disjoint paths, these are reported by R to S, who sends masked threshold shares corresponding to just this one subset. However, the check in the communication phase still remains the same i.e. R continues to check if *all* the shares lie on a t-degree polynomial (which maintains that the protocol is perfectly-secure). If this check fails, then we re-run the protocol beginning with the randomness generation phase.

While the communication complexity of the above modified protocol (which we refer to as the communication-efficient variant of Protocol 5) is same as Protocol 5 in the worst case (which occurs when the all-honest subset is the last one to be tried by R), the modified protocol has running time $\mathsf{poly}(ML)$. This is because the security of the MAC guarantees that the adversary can make an iteration fail only with negligible probability, say μ. The probability that the protocol does not terminate in γ iterations is therefore μ^γ.

We remark that the MACs help only to improve the expected communication complexity, but the protocol is still perfectly secure (due to the verification done by the receiver R before accepting the output).

Perfect Security with $k > 4t$. In the above construction (Protocol 5), we make S account for all possible disjoint path sets of size $2t+1$ as she does not know in advance which set of $2t+1$ all-honest disjoint paths will eventually reach R. We observe that this can be avoided by assuming a larger connectivity of $k > 4t$. Higher connectivity allows transferring more redundant information, enabling R to recover from the incorrect information rather than simply detect it. We tweak the above construction to let R report back $(3t+1)$ disjoint paths instead. Now upon receiving $(3t+1)$ shares among which at most t could be incorrect, R can use error-correction techniques (such as Reed-Solomon error correction) to reconstruct the correct t-degree polynomial despite the errors. This completes the high-level description of the protocol, which is formally described in Protocol 6.

Theorem 6. *Protocol 6 is a perfectly-secure protocol that allows S to securely send m to R in the presence of a rushing adaptive adversary that actively corrupts at most t parties, as long as the network has connectivity $k > 4t$. The protocol runs for $3LM$ rounds and in each round a party sends $\mathcal{O}(Mn^3 2^n(|m| + \log M))$ bits.*

Proof. Correctness: First, we note that Lemma 2 and the presence of at most t active corruptions imply that $\mathsf{GoodPaths}$ must consist of at least $3t+1$ disjoint paths, therefore $|W| > 3t$ holds. Since the quantity $\frac{|W|-t-1}{2} \geq t$, error-correction would definitely be successful as long as there are at most t errors. This is indeed true in our case as only the shares that were masked using random values sent along paths in $\mathsf{GoodPaths}$ where there was at least one corrupt node could be incorrect; and there could be at most t such paths. Correctness now follows directly from the correctness of the error correction algorithm `RSDec` and the correctness of the reliable communication protocols.

Protocol 6: $\Pi^{\mathsf{prv},k>4t}_{\mathsf{perf},\mathsf{mal}}(\mathsf{Paths},\mathsf{S},\mathsf{R},m)$

The set of possible paths Paths is public knowledge. The sender S has message m as private input. We assume parties have access to decoding algorithm $\Pi_{\mathtt{RSDec}}(W,t)$ that takes as input a vector W of shamir shares with threshold t (viewed as a Reed-Solomon codeword) where some of these may be incorrect, and either removes the errors and returns the correct secret if there are at most $\frac{|W|-t-1}{2}$ of them, or produces $\perp$ if there are more than $\frac{|W|-t-1}{2}$ errors. This can be instantiated for instance by Berlekamp-Welch algorithm [BW86].

Randomness Generation Same as in Protocol 3 where S sends one random value per path $p \in \mathsf{Paths}$. The set $\mathsf{GoodPaths}$ is selected to contain at least $3t+1$ disjoint paths and sent back to S using Protocol 2.

Secure Communication The message m is sent as follows.

1. S computes a shamir-sharing of the message m with threshold t as $(s_1,\ldots,s_{k'}) \xleftarrow{\$} \mathtt{Shamir.share}(m)$, where $k' = |\mathsf{GoodPaths}|$.
2. The sender computes $c_{p_i} = s_i + r_{p_i}$ for each $p_i \in \mathsf{GoodPaths}$.
3. The sender S sends $\{c_{p_i}\}_{p_i \in \mathsf{GoodPaths}}$ to R using an instance of Protocol 2.
4. The receiver computes the shares as $s_i = c_{p_i} - r_{p_i}$ for $p_i \in \mathsf{GoodPaths}$.
5. The sender outputs $m \leftarrow \mathtt{RSDec}(\{s_i\})_{i\in[k']}, t)$.

Fig. 6: Perfectly-secure private communication protocol against $t < n$ active corruptions in a network with connectivity $k > 4t$.

Privacy: Privacy follows by similar argument as Theorem 5. Since the adversary knows at most t shares, (namely, those that were masked using random values corresponding to the (at most) t paths in $\mathsf{GoodPaths}$ that the adversarial nodes were a part of), privacy follows directly from the privacy guarantee of threshold sharing and the security of one-time pad encryption.

Complexity: The round complexity follows analogous to the proof of Theorem 4. Lastly, we analyze the communication complexity. The two instances of the reliable communication protocol Protocol 2 have a communication complexity of $\mathcal{O}(n^3 2^n(|m|+\log M))$. The messages m_p have a complexity of $M(|m|+\log M)$. This totals upto communication complexity of $\mathcal{O}(Mn^3 2^n(|m|+\log M))$.

□

Perfect Security With $k > 2t$? In this section, we elaborate on our conjecture that it is impossible to achieve perfectly private communication against an active adversary in a dynamic network with connectivity $k > 2t$. The guarantee in such a network is that the receiver R receives values sent via a set of at least $t+1$ disjoint paths (if she waits long enough, as shown in Lemma 3). Consider any protocol which, like ours, tries to identify such a set of paths where only correct values were received, it then uses the data received on these paths and public communication to send the secret message. Of course, if there is exactly one suitable set of paths, we would be done, as this would be the all-honest set of paths. However, as we now explain, there are cases where there are multiple sets and where the receiver cannot identify the right one with certainty.

For simplicity, consider $t = 1$ and $k = 2t+1 = 3$. Say, the adversary chooses the same 3-connected graph, say $\mathcal{G}$ in all the rounds: $\mathcal{G}$ has three disjoint paths p_1, p_2 and p_3, where the corrupt node is the last node on path p_2 denoted as P_c, adjacent to the receiver P_r (see Figure 7). The adversary has the following strategy: She blocks the message along the path p_2 and sends two fabricated paths to the receiver node P_r instead. **(a)** p_1' that intersects with p_1 but is disjoint from p_3. **(b)** p_3' that intersects with p_3 but is disjoint from p_1. Note that the last node in each of these fabricated paths would be the corrupt node P_c who directly communicates them to the receiver node P_r. Now, from the perspective of the receiver, there are three disjoint sets of size

2, namely **(1)** $\{p_1, p_3\}$, **(1)** $\{p'_1, p_3\}$ and **(3)** $\{p_1, p'_3\}$. Assuming that the adversary tampers with the values associated with p'_1, p'_3, only the first set has correct values. However, there is no way for the receiver to identify $\{p_1, p_3\}$ to be the all-honest set. This is because the receiver could have the same identical view in the following different scenario – when the last node in p_2 was actually honest, the paths p'_1, p'_3 and p_3 existed and the values delivered for these paths were correct. While, on the other hand, the path p_1 and the value it carries was fabricated by a corrupt node. In this case, the receiver would have the same set of three candidate disjoint sets and values, but $\{p'_1, p_3\}$ is the correct set this time.

While authentication can be used to select the correct set with overwhelming probability as we showed earlier, this cannot give perfect security. Hence if the protocol is not allowed to abort but must continue with some set, there is a non-zero probability that a set will be chosen with one correct and one incorrect value. Intuitively, it is not possible to send a private message reliably based on such data and public communication: the output of the receiver must depend on the values received on both paths (as well as the public communication), if it only depends on one of them, this might be value the adversary knows and the message would not be private. But if it depends on both values and one is tampered with, the output will be incorrect.

This type of argument is of course not a real impossibility proof as it only considers one type of protocol, it should only be taken as evidence for our conjecture.

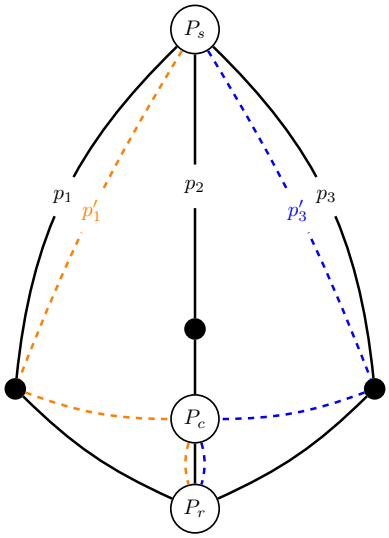

Fig. 7: 3-connected example graph.

References

BW86. E. R. Berlekamp and L. Welch. Error correction of algebraic block codes. *US Patent Number 4,633,470. Issued Dec.*, 1986.

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

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

FF56. L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

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.

Sha79. Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

WW20. Ye Wang and Roger Wattenhofer. Asynchronous byzantine agreement in incomplete networks. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, AFT '20, page 178–188, New York, NY, USA, 2020. Association for Computing Machinery.