Sub-Linear Secure Broadcast and Applications

Yuval Gelles[1], Ilan Komargodski[1], and Merav Parter[2]

[1]Hebrew University of Jerusalem
[2]Weizmann Institute of Science

June 3, 2026

Abstract

We present improved distributed broadcast and MST algorithms that are unconditionally secure against an eavesdropper controlling a fixed set of at most f edges in an n-node m-edge D-diameter graph. We strive for secure algorithms with sublinear round and subquadratic message complexities (in n) for any f. This is in contrast to the exponential or polynomial dependence on f in prior works. Our main results are:

Secure broadcast algorithm, for sending an $O(\log n)$-bit message, that runs in $\tilde{O}(D + \sqrt{n})$ rounds and $\tilde{O}(n^{3/2})$ messages. This matches the state-of-the-art bounds for *insecure* broadcast by [Ghaffari and Kuhn, and Gmyr and Pandurangan, DISC 2018]. Our bounds also improve over the $\tilde{O}(D + \sqrt{fn})$-round complexity and $\tilde{O}(\sqrt{fn} \cdot m)$ message complexity of secure broadcast by [Hitron, Parter and Yogev, DISC 2022].

Secure MST algorithm with sublinear round and subcubic message complexities that improve over the algorithm by [Hitron, Parter and Yogev, ITCS 2023] in the entire regime. In particular, when $f = \Theta(n)$, we improve the round complexity from $\tilde{O}(n^{3/2})$ to $\tilde{O}(n^{2/3})$, and the message complexity from $\tilde{O}(n^3)$ to $\tilde{O}(n^{7/3})$.

Our algorithms are randomized and their correctness and (statistical) security hold with high probability. The algorithms are based on a combination of techniques: Karger's sampling, tree packing and sparse recovery sketches.

Contents

1 Introduction **1**
1.1 Our Contributions . . . 2
1.2 Open Questions . . . 5

2 Preliminaries **5**
2.1 The Communication and Adversarial Models . . . 5
2.2 Graph Theoretic Definitions and Insecure Distributed Tools . . . 6
2.3 Secure Distributed Primitives . . . 8

3 Technical Overview **9**

4 Maximal Forest of One-Sided Sampled Subgraph **12**

5 Secure Broadcast **16**
5.1 Step 1: Trusted Tree Collection . . . 16
5.2 Step 2: Broadcast over Trusted Tree Collection . . . 17
5.2.1 Broadcast on Trusted Tree . . . 17
5.2.2 Sublinear and Sub-quadratic Secure Broadcast . . . 18
5.3 Faster Broadcast for Dense Graph . . . 19

6 Applications **21**
6.1 Aggregate Functions . . . 21
6.2 Secure MST and Low-Congestion Compiler . . . 22

References **27**

A Basic Tools and Proofs **27**
A.1 Secret Sharing Scheme . . . 27
A.2 Sparse Recovery Linear Sketching Scheme . . . 27
A.3 Proof of Lemma 2.4 . . . 30
A.4 Proof of Lemma 5.5 . . . 30
A.5 Proof of Claim 4.6 . . . 30

B Fractional Tree Packing - Revisited **31**

1 Introduction

As modern networks continue to grow in scale and complexity, the private information of their users is under a constant threat of leakage to untrusted parties. We consider secure message passing algorithms against eavesdropper adversaries (a.k.a passive wiretappers) who can eavesdrop on a bounded number of communication links in the given network (whose identity is unknown to the network's participants). Our objective is to provide round and communication efficient distributed algorithms that ensure that the eavesdropper obtains no non-trivial information about the private inputs of the users. As we discuss soon, existing secure algorithms scale poorly with the corruption budget of the eavesdropper.

We consider the CONGEST model [Pel00], where in each round of communication, a vertex can exchange $O(\log n)$-bit messages with each of its neighbors. The communication network is abstracted by an n-vertex m-edge D-diameter graph $G = (V, E)$ with unique vertex identifiers of $O(\log n)$ bits. Communication proceeds in synchronous rounds. We want to protect the users of the network from an eavesdropper who has full information on the graph topology and the protocols executed by the vertices. This adversary is computationally unbounded and can eavesdrop on at most f edges selected at the start of the execution. The adversary, however, cannot see neither nodes' private inputs nor randomness.

Traditionally, the literature on distributed graph algorithms focuses on understanding the graph structure of a given network and utilize it for providing efficient distributed solutions that meet various communication bottlenecks, induced by the network structure. Most often, this comes at the cost of sacrificing privacy of users. Indeed, most existing distributed algorithms are highly insecure, as throughout their executions the nodes gain information on the input and output of their neighbors and beyond.

Obtaining security for various distributed algorithmic tasks in the CONGEST model is a topic that has received a lot of attention recently. The works that initiated this field [PY19a, PY19b, PY19c] focus on the case where $f = 1$ and provide generic compilers that take any "natural" distributed algorithm[1] and turn it into a secure one (again, assuming only one corruption). Their round overhead is $\tilde{O}(D \cdot \mathsf{poly}(\Delta))$, where D is the diameter of the graph and Δ is the maximum degree.[2] The extension of their approach to handling f corruptions leads to an overhead of $(\Delta D)^{\Theta(f)}$ for $\Theta(f)$-edge connected graphs. This is prohibitive already for $f \approx \log n$. Hitron and Parter [HP21] showed that a large class of algorithms (so-called "routing-based" algorithms) inherently suffer from exponential blowup in f. For the fundamental problems of broadcast and MST, Hitron, Parter and Yogev [HPY22, HPY23] bypassed the exponential in f barrier by employing a coding based approach, resulting in a polynomial in f overhead. Furthermore, using techniques from coding-based distributed algorithms, Parter [Par23] managed to bypass the exponential in f barrier, showing a compiler with overhead $\mathsf{poly}(\Delta, f) \cdot \tilde{O}(D + \sqrt{n})$.

These results raise the following very interesting question.

Is polynomial dependence on f in the round complexity inherent? Is it possible to come up with compilers whose overhead is independent of f?

Even for particular distributed tasks of interest it is not known whether the dependence on f is necessary. Central tasks that have been studied include *broadcast* and a class of global graph optimization problems like *minimum spanning tree* (MST), *minimum cut*, *shortest paths*, and more. In broadcast, the goal is to deliver all the vertices a given message M, while leaking no information

[1] Whose local computation is polynomial.

[2] The notation $\tilde{O}(\cdot)$ hides poly-logarithmic factors.

neither about M nor about the sender to an eavesdropper.[3] In graph optimization problems, we consider graphs with weights on their edges and the goal is to hide from the adversary the weights. For broadcast, the state of the art is a secure algorithm with $\tilde{O}(D + \sqrt{fn})$ rounds [HPY22]. For the global graph optimization problems, like MST, the state of the art is a secure algorithm with $\tilde{O}(D + f\sqrt{n})$ rounds [HPY23]. In both cases, there is a polynomial dependence on f.

Is there a way to avoid the dependence on f for broadcast and/or MST?

Notably, broadcast is an important building block in many privacy-preserving applications such as secure multiparty computation, distributed auctions, and even global graph optimization problems (such as MST). Therefore, often any improvement in the efficiency of broadcast directly leads to improvements in its applications.

One task where a secure algorithm exists with runtime independent of f, is a significant relaxation of broadcast called *unicast*, wherein a pair of vertices s and t wish to exchange a message [DDWY93]. Jain's protocol [Jai04] (see also [FMSS04, GI99, HPY22]) requires for every graph G only $O(D)$ rounds and works as long as s and t are connected in $G \setminus F$, where F is the set of edges controlled by an adversary. A naive approach for solving secure broadcast is by running n application of unicast with n pairs s, t for any $t \in V$. This, however, leads to a round complexity of $O(n)$.

Sub-linear algorithms. Ignoring security considerations, one central holy grail in distributed computing is obtaining sub-linear algorithms; i.e., ones that require sub-linear in n number of rounds and total sublinear in n^2 communication bits. For many years, it was considered impossible and that a round complexity of $\Omega(n)$ is optimal, because there are graphs of diameter $D = \Omega(n)$ on which one cannot do better. Analogously, the pioneering work of Awerbuch et al. [AGPV90] provided an $\Omega(m)$ message lower bound for broadcast algorithms that are either deterministic, or randomized comparison-based, or when assuming that vertices do not know the identifiers of their neighbors.

Garay, Kutten and Peleg [GKP98, KP98] were the first to realize that sublinear time algorithms exist in graphs of sublinear diameter. The first message sub-linear algorithm was given by King, Kutten, and Thorup [KKT15]. A long line of work improved either the round or communication complexity of various distributed tasks [GKK$^+$18, Gha14, GL14, BFKL21, HKN21, Nan14, LP13, NS14, GK13, GK18, GP18, SHK$^+$12]. The current state of the art algorithms are not only sub-linear in rounds but also in communication! Indeed, Ghaffari and Kuhn [GK18] and Gmyr and Pandurangan [GP18] developed a broadcast and MST algorithms that complete in $\tilde{O}(D + \sqrt{n})$ rounds and use $\tilde{O}(\min\{m, n^{3/2}\})$ total communication bits.

No analog results are known for the secure counterparts of broadcast or MST. Indeed, the sublinear time algorithms for broadcast and MST suffer from super linear (in n^2) total message complexity (even when $f = o(n)$, but even more so when $f = \Omega(n)$). This general direction is summarized in the following question.

Is there a way to design secure algorithms with sublinear time and subquadratic communication?

1.1 Our Contributions

We answer the above two questions affirmatively, presenting the first *sub-linear-round*, *sub-quadratic-message* broadcast algorithm secure against eavesdropping adversaries. Our protocols provide information-theoretic security while significantly reducing round and communication overhead

[3]The abstraction that hides both the broadcasted message and the broadcaster is sometimes referred to as *anonymous* broadcast. We shall refer to this task merely as broadcast.

compared to prior work. In many regimes of parameters, our protocols are either optimal or optimal given the state of the art results in non-secure algorithms. We elaborate on these next.

We present two protocols, each with a different tradeoff between rounds and messages. Our first protocol's complexity is completely independent of the number of faulty edges and achieves sub-linear round complexity, and sub-quadratic number of total transmitted bits. This protocol is the first ever to achieve either of these properties, let alone simultaneously. Our second protocol is optimized for dense graphs and achieves even lower round complexity in such a case. All of our protocols require randomness and their correctness and security guarantees hold with probability $1 - n^{\Theta(\log n)}$.

Let $G = (V, E)$ be a D-diameter λ-edge connected[4] graph with $|V| = n$ and $|E| = m$. Suppose a computationally unbounded adversary that can statically corrupt and eavesdrop on up to f edges, where $f \leq \lambda/(2 + \epsilon)$ for any constant $\epsilon > 0$. We note that the requirement of a graph being $(f + 1)$-edge connected is necessary for handling f faulty edges, and our connectivity threshold is slightly larger. All presented algorithms are randomized and their correctness and security hold w.h.p.

Theorem 1.1 (Message-efficient secure broadcast; Informal)**.** *There exists a secure broadcast protocol for a b-bit message with the following properties:*

- ***Round complexity**: $\tilde{O}(D + b + \sqrt{bn})$.*
- ***Total message complexity**: $\tilde{O}(bn + \sqrt{b} \cdot n^{3/2})$.*

For $b \geq n$, the above complexities simplify to $\tilde{O}(b)$ round complexity and $\tilde{O}(bn)$ total messages. The total number of messages is almost optimal as in order to transmit a b-bit message to n parties, each one of the n parties needs to receive at least b bits. For $b < n$, the complexities simplify to $\tilde{O}(D + \sqrt{bn})$ rounds and $\tilde{O}(\sqrt{b} \cdot n^{3/2})$ messages. Together, these parameters are very close to the best known complexity of broadcast *without security* ($\tilde{O}(D + \sqrt{bn})$ rounds and $\tilde{O}(\min\{bm, \sqrt{b} \cdot n^{3/2}\})$ messages [GK18, GP18]).

Our second algorithm works best for dense graphs and is able to further parallelize communication, yielding the following improvement.

Theorem 1.2 (Round-efficient secure broadcast; Informal)**.** *Assuming $f^2 b \geq n$, there exists a secure broadcast protocol for a b-bit message with the following properties:*

- ***Round complexity**: $\tilde{O}(D + \sqrt{n} + \sqrt[3]{b/f} \cdot n^{2/3})$.*
- ***Total message complexity**: $\tilde{O}(m + \sqrt[3]{fb^2} \cdot n^{4/3})$.*

Applications beyond broadcast. As mentioned, broadcast is a fundamental primitive in many applications, and as such, our improved broadcast algorithms imply improved state of the art applications. For instance, we get the best known secure MST algorithm as a direct corollary of [HPY23].

[4] A λ-edge connected graph is a graph where the removal of any $\lambda - 1$ edged does not disconnect the graph.

Theorem 1.3 (Secure MST; informal)**.** *There exists a (statistically) secure MST algorithm requiring $\tilde{O}(D + \min\{n^{2/3}, \sqrt{fn}\})$ rounds, and $\tilde{O}(m + f \cdot n^{4/3} + \sqrt{f} \cdot n^{3/2}) = \tilde{O}(n^{7/3})$ total messages.*

This improves over the round complexity of $\tilde{O}(D+f\sqrt{n})$ rounds and $\tilde{O}(m+f^2n+fn^{1.5})$ message complexity of [HPY23], which is $\tilde{O}(n^3)$ for some parameters. In particular, our MST algorithm is the first sublinear-round construction!

Additionally, our techniques extend to protocols beyond broadcast. Essentially, our techniques naturally extend to give a CONGEST algorithm for securely computing aggregate functions (e.g., sum, max, min, etc.). An *aggregate function* is a function that summarizes a dataset distributed across multiple nodes by computing partial results locally and combining them efficiently. Let $\{x_1, x_2, \ldots, x_n\} \subseteq X^n$ be a dataset partitioned across n nodes, such that the i-th node holds $x_i \in X$, where X is a finite domain. An aggregate function $\mathcal{A} : X^n \to Y$ is a function of the form

$$\mathcal{A}(x_1, \ldots, x_n) = f(h(g(x_1), h(g(x_2), \ldots, g(x_n)) \ldots)),$$

where $g\colon X \to Z$, $h\colon Z \times Z \to Z$, and $f\colon Z \to Y$, and where Z is a finite domain. Further, the functions g and h satisfy associativity ($h(h(a,b),c) = h(a,h(b,c))$), commutativity ($h(a,b) = h(b,a)$), and have an identity element.

Theorem 1.4 (Message-efficient secure aggregation; Informal)**.** *For every Z such that each element can be described using* polylog(n) *bits, there exists a secure aggregation protocol with the following properties:*

- ***Round complexity**: $\tilde{O}(D + \sqrt{n})$.*
- ***Total message complexity**: $\tilde{O}(n^{1.5})$.*

Lastly, we give a generic compiler for (insecure) protocols with "light" congestion. The *congestion* of an algorithm is the maximum number of messages sent through a single edge. This result is obtained by using our broadcast protocol within the protocol of [HPY23].

Theorem 1.5 (Generic compiler; informal)**.** *Given an (insecure) algorithm with congestion k, there exists a compiler that outputs a secure version of the algorithm, with the following overhead:*

- ***Round complexity**: Additional $\tilde{O}(D + \min\{\sqrt[3]{k} \cdot n^{2/3}, \sqrt{kfn}\})$ rounds.*
- ***Total message complexity**: Additional $\tilde{O}(m + fk^{2/3} \cdot n^{4/3})$ messages if $n < kf^3$ (i.e., if $\sqrt[3]{k} \cdot n^{2/3} < \sqrt{kfn}$), and $\tilde{O}(\sqrt{kf^3} \cdot n^{3/2})$ otherwise.*

On the necessity of statistical security. Our algorithms (as well as the efficient secure algorithms of previous works, e.g., [HPY22]) achieve statistical security.[5] In statistical security

[5]We note that by turning the algorithm of [HPY22] into a Las-Vegas rather than Monta-Carlo algorithm, it can achieve perfect security (but with a running time bound that only holds w.h.p.)

there is a small, typically negligible, probability that security does not hold (the probability is over the internal randomness of the protocol). One may ask whether *perfect* security, where privacy is maintained *no matter what*, is possible. We show that statistical security is actually inherent for sublinear algorithms.

Observation 1.6. Perfectly *secure broadcast algorithms require* $\Omega(fn)$ *messages.*

Here is the proof. Assume, towards a contradiction, that such a protocol Π exists. Let p_i be the probability that v_i will communicate through at most f edges. Since there are $o(fn)$ messages, there exists a v_i with $p_i > 0$. The adversary $\mathcal{A}$ can guess the identity of v_i and will eavesdrop over a randomly chosen f edges incident to v_i. With some non-zero probability $\mathcal{A}$ will learn the entire communication of a single vertex and will deduce the broadcast message, which is a contradiction to perfect security.

1.2 Open Questions

Our work leaves several directions for future work. First, can the constraint $f \leq \lambda/(2+\varepsilon)$ be relaxed? In our analysis, this bound follows from the Nash–Williams tree-packing theorem (Lemma 2.2, or more precisely, the fraction version thereof that is easier to use in distributed settings [CHGK14, Theorem 1.3]; see Lemma 2.3). Second, can our techniques be extended to adaptive adversaries? Sampling-based approaches like ours fail when an adversary can defer corruption decisions until after observing the sample. Finally, establishing nontrivial lower bounds on the round or message complexities remain an intriguing open problem.

2 Preliminaries

We assume a connected graph $G = (V, E)$ with $|V| = n$ and $|E| = m$. For every $p \in [0, 1]$, let $G[p] = (V, E[p])$ denote an instance of a graph obtained by including each edge of G independently with probability p. We further consider a model where each vertex makes independent sampling decisions about its incident edges. Specifically, a vertex includes each edge with probability $\hat{p} = 1 - \sqrt{1-p}$. This choice ensures that the probability that an edge is included in the graph is exactly p (since $2\hat{p} - \hat{p}^2 = p$).

We assume the existence of a universal security parameter $\kappa = \kappa(n) \in \Omega(1) \cap O(\log n)$, where the goal is to achieve $n^{-\Omega(\kappa)}$ probability of error.[6]

2.1 The Communication and Adversarial Models

We consider the standard CONGEST model where vertices are assigned unique IDs of $O(\log n)$ bits and initially, each vertex knows only the identifiers of its neighbors (and thus the identifiers of its incident edges). At every round each vertex can send a possibly distinct $O(\log n)$-bit message over each of its incident edges. The local computation in our algorithms is polynomial. For simplicity we assume that vertices can exchange (polylog(n))-bit messages over each edge in a given round, which incurs a polylog(n) blow-up in the round and message complexities. The main complexity measures of a distributed algorithm is the round complexity and message complexity measured by the total number of bits exchanged throughout the entire algorithm. The *congestion* of a distributed algorithm is the largest number of $O(\log n)$-bit messages exchanged over a fixed edge in the graph.

[6] We require that $\kappa \in O(\log n)$ to have at most logarithmic overhead. However, for (sub)-exponentially small error one can ignore this requirement.

We consider an eavesdropper adversary that selects a fixed edge-set F to eavesdrop before the algorithm begins (but after it is specified). The adversary is passive, i.e., it cannot alter messages or control randomness, but it observes all communication over the chosen edges in an attempt to infer private information. We assume that the adversary knows the graph structure, the algorithm, and it has unbounded computation power. Additionally, the adversary might be centralized and collect the messages that pass through all the corrupted edges. The only information hidden from the adversary is the private inputs and (non-shared) random coins of the vertices.

Security definitions. Let $\Pi\colon X^n \to Y^n$ be a randomized distributed algorithm running on an n-vertex graph $G = (V, E)$, where X and Y are finite sets. For every edge $e \in E$ let $\Pi_e^{\bar{x}} \in \{0,1\}^*$ be the random variable specifying all messages sent through e over the execution of Π with inputs $\bar{x} \in X^n$. For $F \subseteq E$, let $\Pi_F^{\bar{x}} = \{\Pi_e^{\bar{x}}\}_{e \in F}$.

Definition 2.1. For a graph $G = (V, E)$, an algorithm Π is (f, ϵ)-secure if for every (computationally unbounded) machine $\mathcal{A}$, all pairs of inputs $\bar{x}_0, \bar{x}_1 \in X^n$, and any subset $F \subseteq E$ of size f,

$$\left| \Pr\left[\mathcal{A}(\Pi_F^{\bar{x}_0}, G, \bar{x}_0, \bar{x}_1) = 1\right] - \Pr\left[\mathcal{A}(\Pi_F^{\bar{x}_1}, G, \bar{x}_0, \bar{x}_1) = 1\right] \right| \leq \epsilon,$$

where the probability is over the coins tossed by the vertices during the execution of Π. In other words, an all-powerful machine $\mathcal{A}$ selects a subset $F \subseteq E$, observes all messages sent over the edges in F, and can perform arbitrary computations in an attempt to distinguish between the underlying inputs.

An algorithm called *perfectly* f-secure if it is $(f, 0)$-secure, and *statistically* f-secure if there exists a positive constant c, such that for every positive integer $\kappa = \kappa(n)$ the algorithm is $(f, n^{-c \cdot \kappa})$-secure.

2.2 Graph Theoretic Definitions and Insecure Distributed Tools

Edge connectivity and tree packing. A classical results of Tutte [Tut61] and Nash-Williams [NW61] shows that λ-edge connected graphs contains a tree collection (known as *tree packing*) of $\Theta(\lambda)$ edge-disjoint trees. Formally,

Lemma 2.2. *For every λ-connected graph there are at least $\frac{\lambda-1}{2}$ disjoint spanning trees.*

In the distributed setting it is more convenient to consider a fractional relaxation of the above mentioned (integral) tree packing result, which admits an efficient construction in the CONGEST model. A k-size fractional tree packing is a collection of spanning trees $\mathcal{T} = \{T_1, \ldots, T_\ell\}$ and (possibly fractional) weights $w_1, \ldots, w_\ell$. Each tree T_i has a weight w_i such that total weight $\chi(\mathcal{T}) \triangleq \sum w_i \geq k$ and for each edge e, it holds that $\sum_{T_i \in \mathcal{T}, e \in T_i} w_i \leq 1$.

Lemma 2.3 (Theorem 1.3 [CHGK14])**.** *Given a D-diameter λ-connected n-vertex graph, and an integer parameter $\lambda' \leq \frac{\lambda-1}{2}(1 - o(1))$, one can compute in $\tilde{O}(D + \sqrt{\lambda \cdot n})$ rounds, a fractional tree packing $\mathcal{T}$ and a weight function $w\colon \mathcal{T} \to (0, 1]$, such that:*

1. *for every $T_i \in \mathcal{T}$, $w(T_i) = \frac{n_i}{\lceil \log^8 n \rceil}$ for some positive integer $n_i \geq 1$,*
2. *$\chi(\mathcal{T}) \in [\lambda', \lambda'(1 + o(1))]$.*

Karger's sampling. For every $p \in [0, 1]$, let $G[p] = (V, E[p])$ denote an instance of a graph obtained by including each edge of G independently with probability p. The sampling can be implemented locally (with no communication) by letting each vertex including each of its incident

edges independently with probability $\hat{p} = 1 - \sqrt{1-p}$. This choice ensures that the probability that an edge is included in the graph is exactly p (since $2\hat{p} - \hat{p}^2 = p$). We use the following seminal graph sparsification result by Karger [Kar94].

Lemma 2.4. *[Slight adaptation of Karger [Kar94]] Suppose that $G = (V, E)$ is λ-connected and that $p \in (0, 1]$, and $\kappa = \kappa(n) \geq 10$. If $p \geq \frac{\kappa \log n}{\lambda}$, then with probability at least $1 - n^{-\Omega(\kappa)}$, the sampled graph $G[p]$ is $((1 - \kappa^{-0.1})p\lambda)$-connected, and for a subset $F \subseteq E$, $|F[p]| \leq (1 + \kappa^{-0.1})|F|p$. See Appendix A.3 for the proof.*

Chandra et al. [CCD+24] recently showed that the diameter of Karger's sampled graph $G[p]$ can be bounded by $\tilde{O}(np)$.

Lemma 2.5. *[Lemma 5 in [CCD+24]] The graph $G[p]$ for $p = \kappa \cdot \log n/\lambda$ has diameter of $\tilde{O}(n/\lambda)$ with probability $1 - n^{-\Omega(\kappa)}$.*

Dominating Sets. Given a graph $G = (V, E)$, a subset $S \subseteq V$ called *k-dominating set*, if for every $v \in V$, there exists $u \in S$ at distance at most k from v. We use the following distributed construction by Kutten and Peleg [KP95].

Lemma 2.6. *[[KP95]] Given a graph $G = (V, E)$, there is an $O(k \log^* n)$-round distributed algorithm that computes k-dominating set of size at most n/k.*

(Insecure) broadcast and MST in sublinear round and message complexity. Obviously, the straightforward way to (insecurely) broadcast a $\log n$-bit message is by running a BFS and then to broadcast the message down the BFS tree. This approach, however, requires $\Omega(m)$ total messages which might be $\Omega(n^2)$. Ghaffari and Kuhn [GK18] presented the following improvement.

Lemma 2.7. *[Theorems 1 and 2 in [GK18]] Given a D-diameter graph $G = (V, E)$, there is an $\tilde{O}(D+\sqrt{n})$-round distributed algorithm that computes a spanning tree with diameter $\tilde{O}(D+\sqrt{n})$ using $O(\min\{m, n^{3/2}\})$ total messages. Moreover, one can compute an MST and implement broadcast of an $O(\log n)$-bit message within the same round and message complexities.*

Sparse recovery sketches for detecting outgoing edges. Consider the following task: Given a graph $G = (V, E)$ with a connected component $C \subseteq V$ spanned by a D-diameter tree $T \subseteq G$. The component C has exactly one outgoing edge $e = (u, v)$ (i.e., an edge (u, v) with $u \in C$ and $v \notin C$). Vertices are allowed to communicate only on edges of T. The goal is for all vertices in C to learn the identity of e. The naive solution is to propagate the knowledge of all the component members which allows u to determine that e is an outgoing edge and to announce this fact to all vertices in C; however, this requires $O(D + |C|)$ rounds, which might be inefficient.

Instead, we use the linearity of graph sketches. By broadcasting a poly-logarithmic seed to all vertices in C, each vertex can locally compress its edges to poly-logarithmic number of bits such that the ID of the outgoing edge can be extracted from the bitwise-XOR (exclusive or) of the values of all vertices in C. Specifically, since an edge between two vertices in C will be XORed twice, the XORed value will be exactly the desired edge. Since XOR is a linear function, it can be implemented in $\tilde{O}(D)$ rounds. By using $\tilde{O}(k)$-size hash-table, this approach can be extended to extract k outgoing edges. See Appendix A.2 for a formal definition and proof. Here, we include the lemma, restricted to our use.

Lemma 2.8 (Sparse Recovery Sketches for Detecting Outgoing Edges)**.** *For a given integer k, given an $\tilde{O}(k)$-bit random seed, there are* sketch *and* recovery *algorithms such that the sketching algorithm receives a set of edge IDs and outputs an $\tilde{O}(k)$ size value. The* recovery *algorithm receives a XORed*

value of any number of sketches, and outputs a set of $\tilde{O}(k)$ edge IDs, where if the number of IDs that are XORed an odd number of times is at most k, with probability $1 - n^{-\omega(1)}$, it outputs all of them. Note that there is no guarantee about the output of the recovery algorithm in the complement case.

In our applications, $k = O(\sqrt{n})$. We also provide a direct corollary by applying these algorithms to find all the outgoing edges in a component.

Corollary 2.9. *Fix an n-vertex graph $G = (V, E)$, a D-diameter connected component $C \subseteq V$, and a parameter $\kappa = \kappa(n) \geq 10$. There is an $\tilde{O}(D + \sqrt{n})$-round $O(1)$-congestion algorithm that outputs a subset of edges $E' \subset E$ of size $O(\kappa \cdot \sqrt{n})$. If there are at most $\sqrt{n}$ outgoing edges, with probability $1 - n^{-\Omega(\kappa)}$, all of them will be part of E'.*

The sketching technique for identifying outgoing edges requires that both endpoints of any internal edge $e = (u, v)$ (i.e., with $u, v \in C$) know the ID of e. This is an obstacle that our algorithms need to overcome. Also, note that sketching can be simulated by using a small random seed. This is in contrast to Karger's sampling, which, to the best of our knowledge, cannot be implemented with a sublinear number of bits.

2.3 Secure Distributed Primitives

We make an extensive use of the following secure primitives from [HPY22].

Secure unicast. In the unicast task, a source vertex s is required to send a b-bit message to a designated target t. The target is known by all vertices while the source s can be anonymous (i.e., the vertices do not know the identifier of s). Jain [Jai04] showed that if the graph G contains some trusted s-t path (that is *not* required to be known by the vertices), then s can securely send the desired message to t in time that depends on the (untrusted) graph diameter while having no dependency on the possibly long trusted s-t path. The algorithm of Jain sends only $O(1)$ messages over each edge in the graph and hence can be parallelized when given k distinct instances of the problem.

Lemma 2.10. *[[Jai04], Lemma 3 in [HPY22], [FP23]] Given a D-diameter graph $G = (V, E)$, there is an $O(D)$-round distributed perfectly f-secure algorithm for the unicast task, provided that s and t are connected in $G \setminus F$, where F is the corrupted set. The congestion of the algorithm is $O(1)$. Moreover, k instances of the unicast task (with pairs s_i, t_i) can be solved in $O(D + k)$ rounds.*

Secret sharing. In a secret sharing scheme, a secret message is split into multiple parts, called *shares* with the following property: knowing all shares uniquely restores the message, and knowing all but one share reveals no information (in the information theoretic sense) on the original message.

Definition 2.11 (Secret Sharing [Sha79a])**.** Given a bound on the message size B, a message $x \in \{0,1\}^B$ and a parameter k, the secret share $\mathsf{SecretShare}(x, k, B)$ is composed of k uniformly randomly chosen strings $x_1, \ldots, x_k \in \{0,1\}^B$ called *shares*, conditioned on $x = \bigoplus_{j=1}^{k} x_j$.

When the message x represents an integer number bounded by some integer q, the integer-variant, denoted as $\mathsf{SecretShare}_{\text{int}}(m, k, q)$, splits x into k randomly chosen integer shares $x_1, \ldots, x_k \in \mathbb{Z}_q$ such that $x = (\sum_{j=1}^{k} x_j) \bmod q$.

See Lemma A.4 for a more detailed statement.

3 Technical Overview

Throughout, we assume the KT1 model where each vertex knows the identifiers and the weights of its incident edges in G [AGPV90, GK18, GP18]. A spanning tree T is said to be *trusted* if it contains no faulty edge, i.e., $T \cap F = \emptyset$. In the following we present our take on the algorithm of [HPY22], where we divide the algorithm into two steps. The first computes a collection of trees such that one of them is guaranteed to be trusted, the second step uses this tree collection to securely send the broadcast message. In Section 5.2.1, we prove the following useful tool used implicitly in [HPY22]:

Lemma 3.1. *Let T be a d-diameter spanning tree of G, and let $\widehat{T}$ be a trusted spanning tree of G. Then, there exists a perfectly f-secure algorithm* BroadcastOnTrustedTree *that given a distributed knowledge of these trees, broadcasts a b-bit message in $\tilde{O}(d+\sqrt{bn})$ rounds and $\tilde{O}(\sqrt{b}\cdot n^{1.5})$ messages.*

In our actual algorithms the vertices do not know their incident edges in the trusted tree, and instead are provided with a collection of trees with the guarantee that *one* of them is trusted. The main goal is to compute a small collection of ℓ spanning trees $\mathcal{T}^*$ that contains: (i) a tree T of depth $O(D+\sqrt{n})$ and (ii) a trusted tree $\widehat{T}$ (whose identity is unknown to the vertices). We call such a tree collection a *trusted tree packing*. One can then apply the algorithm of Lemma 3.1 on each tree in that collection resulting in a secure broadcast algorithm with $O(\ell \cdot (D+\sqrt{n}))$ rounds and $O(\ell \cdot n^{3/2})$ messages. [HPY22] computed such a collection with $\ell = \Theta(f)$, and ignoring message complexity aspects. Our main result is in providing such a collection with $\ell = \text{polylog}(n)$ and extending the sublinear broadcast algorithm of [GK18, GP18] to optimize also the message complexity. From that point on, we focus on the computation of the tree collection $\mathcal{T}^*$ that contains a trusted tree. We then explain how to complete the broadcast task using this tree collection.

Computing a Trusted Tree Packing. The approach taken in [HPY22] is based on computing a fractional tree packing $\mathcal{T}$ of $\ell = \Theta(f)$ trees using Lemma 2.3. Already, this step takes $\tilde{O}(D+\sqrt{fn})$ rounds, which dominates their final round complexity. By the properties of this tree packing, we observe (see Appendix B) that one of the trees in $\mathcal{T}$ must be trusted. The round complexity of $\tilde{O}(D+\sqrt{fn})$ shown in [HPY22] is obtained by using a careful fragmentation of the ℓ trees.

To eliminate the dependency in f in the round complexity, our idea is to compute the above mentioned tree packing (which contains also a trusted tree) in Karger's subgraph rather than in G. Specifically, by sampling each edge independently with probability of $p = \Theta(\log n/f)$, Lemma 2.5 shows that the sampled subgraph contains at most $\ell = O(\log n)$ faulty edges and in addition, it is $\Omega(\ell)$ connected. This effectively reduces the actual number of faults from f to $O(\log n)$. Combining this with the algorithm of [HPY22] leads to the desired round complexity of $\tilde{O}(D+\sqrt{n})$ but with possibly quadratic message complexity. As we discuss next, the major barrier for reducing the number of messages is the density of Karger's sampled subgraph. Specifically, recall that we aim at matching the round and message complexities obtained for the insecure broadcast task by [GK18, GP18], namely, $\tilde{O}(D+\sqrt{n})$ rounds and $\tilde{O}(n^{3/2})$ messages. This seems to be challenging already for $f = o(\sqrt{n})$ as the sampled subgraph H might have $\omega(n^{3/2})$ edges. In the distributed implementation of the sampling, each vertex locally samples its incident edges and notify the other endpoints of its successfully sampled edges, which requires $|H|$ messages. This seems to be inevitable as to this date it is unclear how to implement Karger's sampling with a sublinear (in n) number of random coins.

Our approach is based on computing a tree packing in the possibly one-sided known subgraph H (namely, where for each edge in H, possibly only one of the endpoints knows that the edge is in H). Using straightforward reductions, the task of computing the tree packing in Karger's subgraph H

boils down into MST computation in that subgraph. For simplicity, we focus here on an even simpler task of merely computing a spanning tree in H.

Key task: sublinear spanning tree algorithm in one-sided sampled subgraph. Our starting observation is that since the graph H is known in a one-sided manner, we cannot employ the sublinear spanning tree algorithm of [GKK$^+$18] (of Lemma 2.7). The reason is that their algorithm is heavily based on the tool of graph sketching[7] which requires both endpoints of each edge $e = (u, v)$ to know that $e \in H$. In the work of [GKK$^+$18], $H = G$ and hence this requirement holds for free. Computing over one-sided known subgraph calls for a new approach. The key result is:

Theorem 3.2 (Sublinear Spanning Tree in One-Sided Sampled Subgraph)**.** *Let G be a D-diameter n-vertex weighted graph and let $H = G[p]$ be a one-sided sampled G-subgraph. One can compute a spanning tree of H in $\tilde{O}(D + \sqrt{n})$ rounds and $\tilde{O}(n^{3/2})$ messages.*

Note that the key task is concerned with providing an *insecure* graph algorithm that eventually leads to the computation of tree packing. The latter is used by the *secure* broadcast algorithm.

We now sketch the proof. Similarly to [GK18] we distinguish between high and low-degree vertices using a degree threshold of $\sqrt{n}$. Despite the fact that our spanning tree should be a subgraph of H, the classification into high and low degree vertices is based on the G-degrees. This leads to some complications due to the possibly lower degree of these vertices in H.

We implement a variant of Boruvka's algorithm with star contractions. The algorithm has $O(\log n)$ phases. In each phase, given is a collection of vertex-disjoint components $C_1, \ldots, C_\ell$ where each C_i is a spanned by tree $T_i \subseteq H$ rooted at a center s_i. We need to handle two challenges that are not encountered in the setting of [GK18].

The diameter challenge. In the algorithm of [GK18], one first covers all high-degree vertices by a collection of $\tilde{O}(\sqrt{n})$ stars. This is done by sampling $\tilde{O}(\sqrt{n})$ star centers, and letting each high-degree vertex join the star of one of its sampled neighbors; the latter exists w.h.p. Boruvka's algorithm then merges these stars in $O(\log n)$ steps. Since there are at most $\tilde{O}(\sqrt{n})$ stars (of diameter 2), the diameter of each component is bounded by $\tilde{O}(\sqrt{n})$.

In our case, the H-degree of a high-degree vertex might be $O(\sqrt{n}/f)$ and hence we need to sample $\tilde{O}(f\sqrt{n})$ vertices to hit the degrees of all high-degree vertices, leading to components with potentially large diameter. To circumvent this problem, we use the notion of low-congestion shortcuts [GH16] and augment each component with additional G-edges to reduce the diameter of each component to $O(\sqrt{n})$. Those G-edges are only used for the sake of the efficiency of computing a H-subgraph, and are not part of the final spanning tree. The shortcuts exploits the fact that the G-degree of each vertex is $\Omega(\sqrt{n})$.

The one-sided knowledge challenge. Our main goal is for each component C of heavy vertices to learn one outgoing H-edge of C (if such exists). By our shortcut scheme, we can assume that the diameter of each component C is $\tilde{O}(\sqrt{n})$. Our approach is based on distinguishing between two types of components: dense and sparse. A component C is *sparse* if it has at most $\sqrt{n}$ outgoing edges in G (and hence also in H). Otherwise, the component is *dense*. Assume first for simplicity that all components are sparse and with at least one outgoing edge in H (i.e., the components are not isolated in H). We use the notion of k sparse recovery sketches for $k = \sqrt{n}$. By broadcasting a $\tilde{O}(k)$-length random seed over the components, each vertex can locally compute its sketch, and by

[7] This tool is used in [GKK$^+$18] to detect an outgoing edge of MST fragments. The linearity of the sketching cancels internal edges (with both endpoints in the same MST fragment) that are added to the sketch by both of their endpoints.

aggregating the sketches over the component, the collection of all outgoing edges E' in G can be recovered. By communicating over the E' edges, their endpoints can learn if these edges are in H and one such edge is selected by a simple upcast over the component.

Next, assume the other extreme case where all component are dense. In this case, we apply a more aggressive sampling in a way that guarantees that at least one outgoing H-edge survives while keeping the total number of sampled edges to be sufficiently small. Specifically, each edge of $G[C_i]$ is sampled independently probability of $q = \Theta(\log n/\sqrt{n})$. As the number of sampled edges is $\tilde{O}(n^{3/2})$ w.h.p., the vertices can communicate over these edges and deduce those that are outgoing in H. Then an arbitrary selected such edge can be computed by a simple upcast operation using $O(\sqrt{n})$ rounds, by our bound on the diameter of each component. In the actual implementation some of the components might be sparse and some are dense. The algorithm then applies the dense and sparse routines consecutively. The analysis shows that every non-isolated component in H detects some outgoing H-edge at the end of these routines.

Putting everything together: secure broadcast given a trusted tree packing. So far, we sketched the ideas for computing a collection of $\ell = O(\log n)$ tree $\mathcal{T}$ such that one of the trees is trusted. The secure broadcast task is completed as follows. The anonymous source vertex s locally computes ℓ secret shares of its broadcast message M given by $M_1, \ldots, M_\ell$ where $\sum_{i=1}^{\ell} M_i = M$. The objective is to propagate each M_i over T_i for every $T_i \in \mathcal{T}$. This is done by applying Lemma 3.1 on each tree T_i with the message M_i and the $O(D + \sqrt{n})$-depth tree T. This step takes $\tilde{O}(D + \sqrt{n})$ rounds, as desired. The algorithm can be summarized as follows:

Algorithm SecureBroadcast

Input: Broadcast message M.

- **Step (S1): Tree Packing in One-Sided Karger's Subgraph**
  - Each vertex locally samples each incident edge into H with probability of $p = \Theta(\log n/\lambda)$.
  - Compute a collection of $\ell = O(\log^2 n)$ trees $\mathcal{T}$ in H such that one of the trees is trusted, w.h.p.
- **Step (S2): Broadcast on Trees**
  - Compute a $\tilde{O}(D + \sqrt{n})$-depth spanning tree T_0 by applying Lemma 2.7.
  - Let $M_1, \ldots, M_\ell$ be Shamir's secret share of M.
  - Apply Algorithm BroadcastOnTrustedTree of Lemma 3.1 on each T_i, T^* and M_i for each $T_i \in \mathcal{T}$.

Remark 3.3. For $f \in \Omega(\sqrt{n})$, there is a considerably simpler solution: one can apply Karger's sampling, inform the neighbors on the selected edges, and apply the algorithm of [HPY22]. From Lemma 2.5 the diameter of the sampled graph is $O(\sqrt{n})$ and the number of corrupted edges is $O(\log n)$, resulting is round complexity of $\tilde{O}(\sqrt{n})$.

Secure MST by Broadcasting Longer Messages. [HPY23] presented a secure MST algorithm that works in $\tilde{O}(D + \sqrt{n})$ rounds given that all the vertices hold a secret message of $\tilde{O}(f)$ bits. In [HPY23], securely sending $\tilde{O}(f)$ bits takes $\tilde{O}(D + f\sqrt{n})$ rounds, which dominates their round complexity.

We now focus on providing improved algorithms for securely broadcasting longer messages of b bits (for the MST application, $b = \tilde{O}(f)$). The trivial solution (sketched below) introduces a factor of b in the round and message complexities of our secure broadcast algorithm. We show how to improve the dependency to $\sqrt{b}$, as well as, optimizing the bounds for larger connectivity values. Altogether, this approach provides the first secure MST whose round and message complexities are independent in f.

The naive solution for sending the b-bit message is to perform the previous secure broadcast algorithm b times, resulting in $\tilde{O}(D+b\sqrt{n})$ rounds and $\tilde{O}(b\cdot n^{1.5})$ messages. We use two methods for improving the round and message complexities. The first method uses the algorithm of Lemma 3.1: for a given trusted tree, the algorithm first partitions it into $\sqrt{n/b}$ MST fragments each of size $\sqrt{bn}$, each fragment has some leader vertex (which represents the fragment). By $O(b\cdot\sqrt{n/b})$ applications of the secure unicast procedure (Lemma 2.10), the source s sends the b-bit message to the $\sqrt{n/b}$ leaders. This is done in $O(D+\sqrt{bn})$ rounds. In the second step, each leader sends the secret broadcast message to its fragment over the trusted tree, which takes $\tilde{O}(\sqrt{bn})$ rounds. Using our approach, we get round complexity of $\tilde{O}(D+\sqrt{bn})$ and message complexity[8] of $\tilde{O}(\sqrt{b}\cdot n^{1.5})$.

The second method is favorable for dense graphs. We use the result of [CCD+24] that shows that the diameter of Karger's sampled subgraph can be bounded by $\tilde{O}(n/f)$ (Lemma 2.5). For $b = f$, we show that the method yields round complexity of $\tilde{O}(D+\min\{\sqrt{fn}, n^{2/3}\})$ and $O(n^{7/3})$ messages. The idea is to sample $k = \tilde{O}(f/n^{1/3})$ independent subgraphs $H_1,\ldots,H_k$, where each H_i is a Karger's subgraph. W.h.p., each subgraph is $\Omega(\log n)$-edge connected and contains $O(\log n)$ fault edges. Moreover, the subgraphs are nearly edge-disjoint. We will employ the first method to securely send $b' = n^{1/3}$ bits in each subgraph, in *parallel*. Since the diameter of each H_i is $\tilde{O}(n/f)$, this can be done in $\tilde{O}(n/f+n^{2/3})$ rounds. The total message complexity is $\tilde{O}(\sqrt{b'}\cdot k\cdot n^{3/2}) = \tilde{O}(n^{7/3})$.

4 Maximal Forest of One-Sided Sampled Subgraph

Our main technical contribution is an algorithm that for a (not necessarily connected) graph[9] G and one-sided known subgraph $H = G[p]$ (both possibly disconnected), efficiently computes a graph $\mathbf{F}$ that contains a maximal forest of H that has $\tilde{O}(n^{1.5})$ edges. Looking forward, we will use two simple reductions:

1. Computing a collection of trees, where one of them is free of corrupted edges. This can be done by using poly-logarithmic many executions of an MST algorithm over one-sided known graphs with poly-logarithmic number of weight classes.

2. Computing an MST over one-sided known graphs with poly-logarithmic number of weight classes can be done by using the MST algorithm of [GK18] over the union of set of graphs, one for each weight class, such that each one has at most $\tilde{O}(n^{1.5})$ edges, and it contains a maximal forest of all the edges with the same weight.

In this section, we prove the following lemma.

Lemma 4.1. *Let $G = (V,E)$ be a graph and $H = G[p] = (V,E')$ be a one-sided known sampled graph. There exists an algorithm that computes a subset $\mathbf{F} \subseteq E'$ containing a maximal forest of H, where $|\mathbf{F}| \in \tilde{O}(n^{1.5})$. The round complexity of the algorithm is $\tilde{O}(\sqrt{n})$ and the message complexity is $\tilde{O}(n^{1.5})$.*

[8] We get this message complexity by running Jain's algorithm on a graph containing $\tilde{O}(n)$ edges: the union of all $O(\log n)$ trees in the tree packing and the $O(D+\sqrt{n})$-depth tree from Lemma 2.7.

[9] In our applications of this algorithm, we employ it on subgraphs of the input graph which are not necessarily connected.

Key Task: Maximal Forest of High-Degree Vertices. A vertex $v \in V$ is called *heavy* if its degree in G exceeds $\sqrt{n}$, and *light* otherwise. Our main contribution is in showing the following lemma.

Lemma 4.2. *Fix a graph $G = (V, E)$, a one-sided known sampled graph $H = G[p] = (V, E')$, and a set of heavy vertices S (in G). There is an $\tilde{O}(\sqrt{n})$-round $\tilde{O}(n^{1.5})$-message algorithm that outputs a maximal forest of $H[S]$, the graph induced by the heavy vertices.*

The round complexity is achieved by reducing the diameter of each heavy component of heavy vertices to $O(\sqrt{n})$. This is achieved by using the notion of low-congestion shortcuts [GH16], as we explain next. For a heavy vertex u, let $N_{\sqrt{n}}(u)$ be the first $\sqrt{n}$ neighbors of u in G (based on node IDs). Define:

$$E_{\sqrt{n}}(S) = \{(u, v) \mid u \in S, v \in N_{\sqrt{n}}(u)\}. \tag{1}$$

For a connected component S, the edges of $E_{\sqrt{n}}(S)$ play the role of low-congestion shortcuts for $G[S]$. In particular, for a collection of vertex-disjoint connected components $S_1, \ldots, S_k$ of heavy vertices, one can send a message over each S_i, in parallel, using $O(\sqrt{n})$ rounds. This is based on the following observation.

Observation 4.3. *Any n-vertex graph with minimum degree δ has diameter $O(n/\delta)$.*

Claim 4.4. *Let $S_1, \ldots, S_k$ be a collection of vertex-disjoint connected components where all vertices in $\cup S_i$ are heavy. Let T_i be any spanning tree of S_i and let $H_i = T_i \cup E_{\sqrt{n}}(S_i)$. Then, the diameter of H_i is $O(\sqrt{n})$ and each edge appears on at most two H_i subgraphs.*

Proof. Suppose, for contradiction, that the diameter of H_i is at least $12\sqrt{n}$ and let P be the longest shortest path. Let $B_i = V(H_i) \setminus S_i$. Since H_i does not include edges in $B_i \times B_i$, any H_i-path of length at least 2 must contain an S_i-vertex. Hence, there are at least $6\sqrt{n}$ heavy vertices on P. Since P is a shortest path in H_i, the H_i-neighborhoods of heavy vertices at distance at least 3 on P must be disjoint. Since each heavy vertex has $\sqrt{n}$ neighbors in H_i, we get a contradiction that the number of vertices is n.

For any edge (u, v), an edge can appear in at most two subgraphs: one for the component S_u containing u and the other for the component S_v that contains v.

□

Our algorithm combines these low-congestion shortcuts with the well-known star contraction of connected components. Each heavy vertex starts as a component, and at each one of the $O(\log n)$ iterations the members of each component calculate an outgoing edge to another component, using sparse recovery sketches, and merge into this component. Note that, to forbid long sequence of components that want to merge each other, at each iteration, a component is randomly assigned as a *center* or a *satellite*, where a "center" component connects only to a "satellite" one. Throughout, we use the low-congestion shortcut of Claim 4.4 to send messages over a component.

A component is *dense* if it has at least $\sqrt{n}$ outgoing edges in G, and otherwise it is *sparse*. To detect an outgoing edge of a sparse component, we use the sparse recovery procedure of Corollary 2.9 which recovers all the outgoing edges of the component and allows the vertices to pick an outgoing edge that belongs to H. For the dense components, the vertices sample edges with smaller probability to guarantee that only $\tilde{O}(n^{3/2})$ edges are sampled, this allows the vertices to directly communicate over the sampled edges, and detect an outgoing edge in H, as well. See below for a formal description.

Maximal Spanning Forest of Heavy Vertices

Setup Phase:

1. Each light vertex informs all its neighbors that it is a light vertex. Let $V^* \subseteq V$ be the set of heavy vertices and $G^* = G[V^*]$ be the subgraph induced by the heavy vertices.

2. Each heavy vertex $v_i \in V^*$ considers itself the leader of the component C_i and set the communication tree $T_i = (C_i, \emptyset)$.

Star Contraction Phase:

3. Repeat the following for $\kappa \log n$ iterations:

   (a) Each component leader v_i tosses a fair coin to decide whether it is a "center" or "satellite".

   (b) If v_i is a center, it broadcasts "center" to all vertices in C_i over T_i.

   (c) If v_i is a satellite:

   Handling Sparse Components:

   i. The vertices in C_i run the sparse-recovery algorithm of Corollary 2.9 over the edge space, to extract a subset of edge IDs E_{C_i}.

   ii. All vertices in C_i communicate with the endpoints of E_{C_i} to find an edge in H leading to a center component; they select an arbitrary such edge (if one exists) and add it to $\mathbf{F}$.

   Handling Dense Components:

   iii. If no edge to a center component is found, the vertices of C_i sample each edge in G^* independently with probability $\frac{\kappa \log n}{p\sqrt{n}}$. Suppose the edge is also in H. In that case, they check whether it connects to a center component and, collectively, add a single such outgoing edge to $\mathbf{F}$.[a]

   (d) If an edge to center component is found, the satellite component members update their component ID to the ID of the center component. Note that merging into center component invalidating it communication tree.

   (e) For each component C_i in $\mathbf{F}$, update the communication tree T_i to be a BFS tree of $\mathbf{F}[C_i] \cup E_{\sqrt{n}}(C_i)$.

[a] Since the number of selected edges can be large with some small probability, to bound the number of messages each vertex selects at most $\kappa \log^3 n$ edges, leading to an additional $n^{-\omega(1)}$ term in the probability of error.

The proof of Lemma 4.2 proceeds as follows: we first establish correctness (i.e., that $\mathbf{F}$ is a maximal forest) and then analyze the complexities. We start by showing that in each iteration of the star contracting phase, a satellite component will fail to merge if and only if it has no outgoing edge in H to a center component.

Claim 4.5. *In step 3, with probability $1 - n^{-\Omega(\kappa)}$, if C_i is a non-isolated satellite component, it will find an edge to another component.*

Proof. Let $\hat{E}$ denote the set of edges incident to at least one vertex in C_i. We consider two cases:

- **Dense Component:** We analyze the probability that an outgoing edge $\{v,u\}$ is present in both H and $\hat{E}\left[\frac{\kappa \log n}{p\sqrt{n}}\right]$. Since these are independent events,

$$\Pr\left[\{v,u\} \in H \cap \hat{E}\left[\frac{\kappa \log n}{p\sqrt{n}}\right]\right] = \Pr\left[\{v,u\} \in H\right] \cdot \Pr\left[\{v,u\} \in \hat{E}\left[\frac{\kappa \log n}{p\sqrt{n}}\right]\right].$$

  Since H is a one-sided known sampled graph:

$$\Pr\left[\{v,u\} \in H\right] \cdot \Pr\left[\{v,u\} \in \hat{E}\left[\frac{\kappa \log n}{p\sqrt{n}}\right]\right] =$$
$$(1-\sqrt{1-p}) \cdot \frac{\kappa \log n}{p\sqrt{n}} \geq \frac{p}{2} \cdot \frac{\kappa \log n}{p\sqrt{n}} = \frac{\kappa \log n}{2\sqrt{n}}.$$

  Therefore, the probability that no outgoing edge is in $H \cap \hat{E}\left[\frac{\kappa \log n}{p\sqrt{n}}\right]$ is at most

$$\left(1-\frac{\kappa \log n}{2\sqrt{n}}\right)^{\#\text{outgoing edges}} \leq \left(1-\frac{\kappa \log n}{2\sqrt{n}}\right)^{\sqrt{n}} \leq e^{-\frac{\kappa \log n \sqrt{n}}{2\sqrt{n}}} = n^{-\frac{\kappa}{2}}.$$

- **Sparse Component:** Since we use a $\sqrt{n}$-sparse-recovery sketching scheme, by the union bound, with probability $1-n^{-\Omega(\kappa)}$, E_{C_i} will contain all outgoing edges of C_i in G, and therefore, also in H.

□

To complete the correctness argument, we show that $\kappa \log n$ iterations are sufficient to merge all possible components.[10]

Claim 4.6. *After $\kappa \log n$ iterations of step 3, with probability $1-n^{-\Omega(\kappa)}$, all components in $\mathbf{F}$ are isolated in H.*

See Appendix A.5 for the proof.

Efficiency analysis. To analyze the algorithm's complexities, we first bound the diameter of each component (including the shortcuts). We analyze each phase separately:

- **Setup Phase:** Each light vertex has at most $\sqrt{n}$ neighbors in G, and each heavy vertex has $\sqrt{n}$ neighbors in the low-congestion subgraph of its component. Therefore, the total number of messages in this step is $\tilde{O}(n^{1.5})$.
- **Merging Components Phase**: By Claim 4.4, each broadcast, aggregation, or BFS is performed on a tree of depth $O(\sqrt{n})$.
  - BFS: Requires $\tilde{O}(\sqrt{n})$ rounds with edge congestion $O(1)$. Since $\mathbf{F} \cup E_{\sqrt{n}}(V^*)$ has $\tilde{O}(n^{1.5})$ edges, the total number of messages is $\tilde{O}(n^{1.5})$.
  - Broadcast: Requires $\tilde{O}(\sqrt{n})$ rounds with edge congestion $O(1)$ over a BFS tree, so the total number of messages is $\tilde{O}(n)$.

[10] Traditionally, star-merge requires $O(\log n)$ rounds; here, we allow κ to be super-constant when negligible (sub-inverse-polynomial) error is required.

– Aggregation: By the properties of the linear sketch (see Definition A.5), each sketch size is $\tilde{O}(\sqrt{n})$ bits, and aggregating (using bitwise XOR) k-bit values on a tree of depth at most D requires $\tilde{O}(k+D)$ rounds and $O(k)$ edge congestion. Since the aggregation is over a tree, the number of messages is $\tilde{O}(n^{1.5})$ and the number of rounds is $\tilde{O}(\sqrt{n})$. See Lemma 4.4 for details.

- **Handling Sparse Components:** By Corollary 2.9, each $|E_{C_i}| \in \tilde{O}(\sqrt{n})$ and there are at most n components. Thus, the number of messages is $\tilde{O}(n^{1.5})$.
- **Handling Dense Components:** By Chernoff's bound, with high probability the number of edges that are both in H and $\hat{E}\left[\frac{\kappa \log n}{p\sqrt{n}}\right]$ is $O\left(m \cdot \frac{\kappa \log n}{\sqrt{n}}\right) = \tilde{O}(n^{1.5})$, which is the total number of messages.

Putting all together (Lemma 4.1). Recall that $G^* = G[V^*]$ is the subgraph induced by the heavy vertices. Then, $H \setminus G^*$ is the sampled graph containing all edges incident to at least one light vertex. We define the desired graph as $\mathbf{F} \cup (H \setminus G^*)$. Since $G \setminus G^*$ contains at most $n^{1.5}$ edges, two neighbors in $G \setminus G^*$ can inform each other if the connecting edge belongs to H or not.

Suppose, towards contradiction, that there exists a component in H that is divided into at least two components, and let $e \in H$ be an edge between such components. If e is incident to a light vertex, then it must be in $H \setminus G^*$. Otherwise, since $\mathbf{F}$ is a maximal forest of H, both heavy vertices must be in the same component, contradicting the assumption. Therefore, $\mathbf{F} \cup (H \setminus G^*)$ must contain a maximal forest of H.

5 Secure Broadcast

In this section we provide two b-bit secure broadcast algorithms, the first runs in $\tilde{O}(D+\sqrt{bn})$ rounds and has message complexity $\tilde{O}(\sqrt{b} \cdot n^{3/2})$. The second algorithm runs in $\tilde{O}(D+\sqrt{n}+\sqrt[3]{b/f} \cdot n^{2/3})$ rounds. While the latter is faster in some non-trivial cases, its message complexity is $\Omega(m)$.

Theorem 5.1. *Given a D-diameter $(2+\epsilon)f$-connected graph $G=(V,E)$, for any positive constant ϵ and a positive integer $b \leq n$. There exists a statistically f-secure $\tilde{O}(D+\sqrt{bn})$-round distributed b-bit broadcast algorithm with message complexity $\tilde{O}(\sqrt{b} \cdot n^{3/2})$.*

Theorem 5.2. *Given a D-diameter $(2+\epsilon)f$-connected graph $G=(V,E)$, for any positive constant ϵ and a positive integer $b \leq n$. If $f^2 b \geq n$, then there exists an statistically f-secure $\tilde{O}(D+\sqrt{n}+\sqrt[3]{bn^2/f})$-round distributed b-bit broadcast algorithm with message complexity $\tilde{O}(m+\sqrt[3]{fb^2n^4})$.*

5.1 Step 1: Trusted Tree Collection

First, we formally define the notion of *trusted* collection of trees.

Definition 5.3. A collection of trees $\mathcal{T}$ is called *f-trusted* (with respect to a graph G) if for every subset of edges $F \subset E$ of size f, there exists $T_i \in \mathcal{T}$ such that $T \cap F = \emptyset$. Additionally, we say that the collection has diameter D if there exists $T_j \in \mathcal{T}$ with diameter D.

This definition is a restricted version of *fractional tree packing* (see Lemma 2.3). When $f \in O(\kappa \log n)$, the algorithm of [CHGK14] for constructing f-trusted collection consists of poly-logarithmic many executions of an MST algorithm. Combined with Lemma 2.7, we achieve the following reduction:

Lemma 5.4. *Fix a D-diameter n-vertex graph G, a one-sided known $(2+\epsilon)\hat{f}$-connected graph $H = G[p]$, and an MST algorithm for one-sided known subgraph. For a constant $\epsilon \in (0,1]$ and $\hat{f} \in O(\kappa \log n)$, there is a distributed algorithm that computes a $O(\kappa \log^2 n)$ size $\hat{f}$-*trusted *collection $\mathcal{T}$ of diameter $\tilde{O}(D + \sqrt{n})$. Additionally, the complexities of the algorithm are the same as the complexities of the MST algorithm, up to poly-logarithmic factors.*

We also provide a simpler proof in Appendix B.

From Maximal Forest to MST. The MST algorithm of [GK18] computes a minimum spanning tree (MST) in $\tilde{O}(D + \sqrt{n})$ rounds using $\tilde{O}(n^{1.5})$ messages for any graph. However, in randomly sampled graphs, the communication cost of informing neighbors about selected edges can be prohibitive when $f \ll \sqrt{n}$. To address this, we state a straightforward lemma in graph theory that provides a reduction from MST to maximal spanning forest.

Lemma 5.5. *Let $G_w = (V, E, w\colon V \to W)$ be a weighted graph, and let $E_c = \{e \in E \mid w(e) = c\}$. Suppose $\mathcal{F} = \{\mathbf{F}_c \subseteq E_c\}_{c \in W}$, where each $\mathbf{F}_c$ contains a maximal forest of $G_c = (V, E_c)$. Then, $\hat{\mathbf{F}} = \bigcup_{\mathbf{F}_c \in \mathcal{F}} \mathbf{F}_c$ contains an MST of G_w. See Appendix A.4 for formal proof.*

We also state the following result.

Lemma 5.6 (Combining Lemmas 6 and 7 of [GK18])**.** *Given a D-diameter n-vertex m-edge graph G and another graph G' over same vertices. There is an $\tilde{O}(D + \sqrt{n})$-round algorithm that computes an MST of G', where the message complexity is $\tilde{O}(\min\{m, n^{1.5}\})$.*

In conclusion, using the algorithm of Lemma 4.1, we got the following result.

Theorem 5.7. *Given a D-diameter n-vertex graph G, a one-sided known $(2+\epsilon)\hat{f}$-connected graph $H = G[p]$. For any constant $\epsilon \in (0,1]$, and $\hat{f} \in O(\kappa \log n)$, there is an $\tilde{O}(D + \sqrt{n})$-round $\tilde{O}(n^{1.5})$-messages algorithm that computes $O(\kappa \log^2 n)$ size $\hat{f}$-*trusted *collection $\mathcal{T}$ of diameter $\tilde{O}(D + \sqrt{n})$.*

5.2 Step 2: Broadcast over Trusted Tree Collection

A naive attempt first samples $G[p]$ using Lemma 2.4, reducing the number of corrupted edges to approximately $\kappa \log n$ (see Lemma 2.4), and then runs the protocol of [HPY22] with $f = (1 + \epsilon/10) \cdot \kappa \log n$. Unfortunately, the diameter of $G[p]$ might be considerably larger than D, which leads to poor round complexity.

5.2.1 Broadcast on Trusted Tree

Like the broadcast algorithm of [HPY22], our protocol uses f-trusted collection of trees to broadcast a message. We first present the BroadcastOnTrustedTree algorithm—the algorithm that proves Lemma 3.1.

BroadcastOnTrustedTree

Input: A b-bit message M_j, a trusted tree T_j, and a D-diameter tree T_0.

1. Run the algorithm of Lemma 2.6 over T_j to compute a $\sqrt{bn}$-dominating set L_j. At this point, each vertex knows its own parent in the spanning forest with $|L_j| \leq \sqrt{n/b}$ trees, where the depth of each tree is at most $\sqrt{bn}$.
2. Each vertex $v \in L_j$ informs v_1 of its identity through T_0.
3. Run $|L_j|$ instances of unicast algorithm over $T_j \cup T_0$ to forward s_j from v_1 to L_j.
4. Each $v_\ell \in L_j$ broadcasts M_j over T_j to all vertices within distance at most $\sqrt{bn}$.

Efficiency. We consider the steps of the algorithm one by one and analyze their complexity.

Step 1: By Lemma 2.6 this step requires $\tilde{O}(\sqrt{bn})$ rounds and $|L_j| \leq \sqrt{n/b}$.

Step 2: Since $|L_j| \leq \sqrt{n/b}$ and the diameter of T_0 is $\tilde{O}(D + \sqrt{n})$, then this step requires $\tilde{O}(D + \sqrt{n} + |L_j|) = \tilde{O}(D + \sqrt{n})$ rounds and $O\left(n^{1.5}/\sqrt{b}\right)$ messages.

Step 3: By Lemma 2.10 this step requires $\tilde{O}(D + \sqrt{n} + b|L_j|) = \tilde{O}(D + \sqrt{bn})$ rounds and $O(\sqrt{bn^3})$ messages.

Step 4: This step requires $O(\sqrt{bn})$ rounds and $O(n)$ messages.

Summing over all steps, the algorithm requires $\tilde{O}(D + \sqrt{bn})$ rounds and the total message complexity is $\tilde{O}(\sqrt{b} \cdot n^{3/2})$.

Security. The security of the algorithm comes directly from the fact that each message is forwarded on the trusted tree, or via the perfectly secure unicast algorithm of [Jai04].

5.2.2 Sublinear and Sub-quadratic Secure Broadcast

We give the precise version of the algorithm from Section 3, which proves Theorem 5.1.

Sublinear and Subquadratic Secure Broadcast

Let v_{i^*} be the vertex that wants to broadcast $M \in \{0,1\}^b$.

Generate Trusted Collection.

1. For every edge $\{u, v\} \in E$, u and v keep the edge with probability p where $2p - p^2 = \frac{\kappa \log n}{f}$. The one-sided known subgraph is denoted by $H = G[p] = (V, E')$. Note that, since both u and v keep the edge independently, the probability that an edge is in H is $\frac{\kappa \log n}{f}$.
2. Generate $((2 + \epsilon/2)\kappa \log n)$-trusted collection with diameter $\tilde{O}(D + \sqrt{n})$ using the algorithm of Lemma 5.4 over G and H, where T_0 is the $\tilde{O}(D + \sqrt{n})$-diameter tree.

Broadcast on Trusted Collection.

3. Run unicast algorithm (Lemma 2.10) over $T = \bigcup_{T_j \in \mathcal{T}} T_j$ to forward x from v_{i^*} to v_1.

4. v_1 create $|\mathcal{T}|$ shares of M using a linear $(|\mathcal{T}|, |\mathcal{T}|-1)$ secret sharing scheme (see Lemma A.4), denote them with $\{M_j\}_{j\in[|\mathcal{T}|]}$.
5. For each tree $T_j \in \mathcal{T}$: broadcast M_j using BroadcastOnTrustedTree algorithm over T_j and T_0.
6. Each vertex reconstructs M from $\{M_j\}_{j\in[|\mathcal{T}|]}$.

Efficiency. We consider the steps of the algorithm one by one and analyze their complexity.

Step 1: This step, consisting of only local computation, does not require any communication.

Step 2: By Lemma 5.4 this step requires $\tilde{O}(D+\sqrt{n})$ rounds and $\tilde{O}\left(n^{1.5}\right)$ messages.

Step 3: Since the number of edges in each tree is $n-1$, and there are $O(\kappa \log^2 n)$ trees, by Lemma 2.10 this step requires $\tilde{O}(D+b+\sqrt{n})$ rounds and $\tilde{O}(bn)$ messages.

Step 4: This step, consisting of only local computation, does not require any communication.

Step 5: By Lemma 5.4 $|\mathcal{T}| \in$ polylog(n) and from Lemma 3.1 it required $\tilde{O}(D+\sqrt{bn})$ rounds and $\tilde{O}(\sqrt{bn^3})$ messages.

Step 6: This step, consisting of only local computation, does not require any communication.

Summing over all steps, the algorithm requires $\tilde{O}(D+\sqrt{bn})$ rounds and the total message complexity is $\tilde{O}(\sqrt{bn^3})$.

Correctness. We show that with probability $1-n^{-\Omega(\kappa)}$ each vertex learns x. All the algorithm's steps succeed with probability 1, except that the connectivity of H might be less than $((2+\epsilon/2)\kappa \log n)$ connected. Since $p = (\kappa \log n)/f$, by Lemma 2.4, $H = G[p]$ is $((2+\epsilon-\kappa^{-0.1})\kappa \log n)$-connected with probability at least $1-n^{-\Omega(\kappa)}$, and therefore, for every constant ϵ there is a constant κ such that H is $((2+\epsilon/2)\cdot \kappa \log n)$-connected.

Security. Let F be the set of eavesdropped edges, an immediate consequence of Lemma 5.4 is the existence of a tree $T^* \in \mathcal{T}$ such that $T^* \cap F = \emptyset$. Therefore, by Lemma 2.10 applied on $T^* \cup T$, any eavesdropper adversary that listens to the edges in F learns nothing from an execution of the multicast algorithm. Combining with the security of the secret sharing scheme (see Definition A.3), we conclude that the protocol is statistically f-secure. Additionally, the anonymity guarantee of the unicast algorithm also hides the broadcaster's identity.

5.3 Faster Broadcast for Dense Graph

In this part we utilize the fact that the diameter of dense graphs is low, that is, for a λ-connected graph, the diameter is $O(n/\lambda)$. Therefore, by sampling a collection of $G[1/k]$ graphs with size k, we obtain nearly disjoint graphs, where the diameter of each of them is $O(kn/\lambda)$.

The following algorithm and the rest of the section prove Theorem 5.2.

Faster Broadcast for Dense Graph

1. Partition x into δ parts $M = (M_1, \ldots, M_k)$, each of size $\frac{b}{k}$, where $\delta = \sqrt[3]{\frac{f^2 b}{n}}$.
2. Sample $G_i = G\left[\frac{1}{\delta}\right]$ for $i \in [k]$.
3. For every $i \in [\delta]$, in parallel, broadcast x_i using broadcast algorithm from Theorem 5.1.

Efficiency. We start by showing that each edge is only part of a small number of sampled graphs. Let $I_e = \{i \in [\delta] \mid e \in G_i\}$. Since each edge is sampled independently in every iteration, $\mathbb{E}[|I_e|] = 1$. By Chernoff's bound (see Lemma A.1),

$$\Pr[|I_e| \geq \kappa \log n + 1] \leq e^{-\frac{\kappa^2 \log^2 n}{2+\kappa \log n}}.$$

By the union bound,

$$\Pr[\exists e \in E \colon |I_e| \geq \kappa \log n + 1] \leq n^2 \cdot e^{-\frac{\kappa^2 \log^2 n}{2+\kappa \log n}} = n^{2-\frac{\kappa^2 \log n}{2+\kappa \log n}} \in n^{-\Omega(\kappa)}.$$

Additionally, since G is $((2+\epsilon)f)$-connected, by Lemma 2.4, for every $i \in [\delta]$

$$\Pr\left[G_i \text{ is not } \left((2+\epsilon/2)\sqrt[3]{\frac{fn}{b}}\right)\text{-connected}\right] \leq n^{-\Omega(\kappa)}.$$

By another union bound,

$$\Pr\left[\exists i \in [\delta] \colon G_i \text{ is not } \left((2+\epsilon/2)\sqrt[3]{\frac{fn}{b}}\right)\text{-connected}\right] \leq \delta \cdot n^{-\Omega(\kappa)} \in n^{-\Omega(\kappa)}.$$

Let $f_i = |F \cap G_i|$. An additional property that is achieved by Lemma 2.4 is that for every $i \in [\delta]$, $f_i \leq (1+o(1))\sqrt[3]{\frac{fn}{b}}$, and we conclude that each G_i is $((2+\epsilon')f_i)$-connected, for some constant $\epsilon' > 0$.

Finally, we bound the round complexity of each (parallelized) doubly sublinear algorithm. Since the diameter of λ-connected graph is $O(n/\lambda)$, and on each graph we broadcast b/k size message, the round complexity of the algorithm is,

$$\tilde{O}\left(D(G_i) + \sqrt{\frac{bn}{\delta}}\right) = \tilde{O}\left(\frac{n}{(2+\epsilon/2)\sqrt[3]{\frac{fn}{b}}} + \sqrt{\frac{bn}{\delta}}\right) = \tilde{O}\left(\sqrt[3]{\frac{bn^2}{f}} + \sqrt{\frac{bn}{\delta}}\right).$$

Replacing δ with $\sqrt[3]{\frac{f^2 b}{n}}$, we get

$$\tilde{O}\left(D(G_i) + \sqrt{\frac{bn}{\delta}}\right) = \tilde{O}\left(\sqrt[3]{\frac{bn^2}{f}}\right).$$

6 Applications

6.1 Aggregate Functions

An aggregate function is a function that can be computed distributively without carry information about previous computation steps (e.g., sum, average, max, argmax, ...). From another perspective, a set, multiset, or collection is aggregate under a function g if g is defined for that type of input and produces a valid single output summarizing the collection. In this section we use the trusted collection to secure the computing of any aggregate function. We completeness we repeat the formal definition,

Definition 6.1. Let $\{x_1, x_2, \ldots, x_n\} \subseteq X^n$ be a dataset partitioned across n nodes, such that the i-th node holds $x_i \in X$, where X is a finite domain. An aggregate function $\mathcal{A} : X^n \to Y$ is a function of the form

$$\mathcal{A}(x_1, \ldots, x_n) = f(h(g(x_1), h(g(x_2), \ldots, g(x_n)) \ldots)),$$

where $g: X \to Z$, $h: Z \times Z \to Z$, and $f: Z \to Y$, and where Z is a finite domain. Further, the functions g and h satisfy associativity ($h(h(a, b), c) = h(a, h(b, c))$), commutativity ($h(a, b) = h(b, a)$), and have an identity element.

Theorem 6.2. *Given a D-diameter $((2+\epsilon)f)$-connected graph $G = (V, E)$. There is a statistically f-secure $\tilde{O}(D + \sqrt{n})$-round distributed algorithm that compute any aggregate function with message complexity $\tilde{O}(n^{1.5})$.*

Note that, while linear functions (e.g. sum) can be securely computed using Jain's algorithm [Jai04] and broadcast in a black-box manner, functions of compare-and-choose type (e.g. min/max) are less obvious. The following algorithm provesnzun Theorem 6.2:

Aggregation Algorithm

1. Run the broadcast algorithm from Section 5.2.2, to the trusted collection $\mathcal{T}$, and the dominating set L_1 of $T_1 \in \mathcal{T}$, where the broadcast message is a degree-$2\kappa \log n$ random polynomial over $\mathbb{F}_q$, for the smallest prime $q > n^2$, generated by v_1, and denoted by h.

2. For each $v_\ell \in L_1$, aggregate x's over the (disjoint) subtree of v_ℓ in T_1 as follows: when vertex v send a message x to vertex u, it sends $x + h(u, v)$, and u recovers x be subtracting $h(u, v)$. Let $\{y_\ell\}_{\ell \in |L_1|}$ be the aggregated values held by the members of L_1.

3. Run unicast algorithm (see Lemma 2.10) over $\bigcup_{T_j \in \mathcal{T}} T_j$ to forward $\{y_\ell\}_{\ell \in |L_1|}$ from L_1 to v_1.

4. v_r computes the aggregated value y of $\{y_\ell\}_{\ell \in |L_1|}$, and broadcasts it like in the broadcast algorithm from Section 5.2.2

Efficiency and correctness. By Bertrand-Chebyshev's theorem (see Lemma A.2), $q \leq 2n^2$ and so it can be represented using $2 \log n + 1$ bits. The rest of the analysis and the correctness is reminiscent of the broadcast algorithm analysis. For completeness, it is given below.

Security. By Lemma 2.4, with probability $1 - n^{-\Omega(\kappa)}$, there are fewer than $2\kappa \log n$ corrupted edges in T_1. Since a degree $2\kappa \log n$ random polynomial is a generator of $(2\kappa \log n)$-wise independent random variable, and for every k-wise independent random variable, and any set of k edges $\{\{u_i, v_i\}\}_{i\in[k]}$, and all sets of k pairs $\{(x_i, z_i) \in Z^2\}_{i\in[k]}$,

$$\forall i \in [k]: \qquad \Pr\left[x_i + h(u_i, v_i) = z_i\right] = \frac{1}{|Z|}, \qquad \text{independently.}$$

Therefore, the algorithm is statistically f-secure. The rest of the security comes from Lemma 2.10 and Theorem 5.1.

6.2 Secure MST and Low-Congestion Compiler

Hitron et al. [HPY23] introduced a general compiler that translates any k-congestion insecure algorithm to a secure one using broadcast of roughly kf random bits, and additionally, showed (in a white-box manner) how to securely generate an MST. First, we combine our broadcast theorems for kf-bit messages:

Theorem 6.3 (Combination of Theorems 5.1 and 5.2 for $b = kf$)**.** *Given a D-diameter $(2+\epsilon)f$-connected graph $G = (V, E)$. For any positive constant ϵ, and a positive integer $k \leq n/f$, there exists a statistically f-secure $\tilde{O}(D + \min\{\sqrt[3]{kn^2}, \sqrt{kfn}\})$-round distributed (kf)-bit broadcast algorithm.*

Note that, for $k \in \text{polylog}(n)$, the round complexity is $\tilde{O}(n^{2/3})$, which is the first sublinear time f-bit secure broadcast algorithm. Combining our broadcast result with [HPY23]'s work, we state the general compiler result,

Lemma 6.4 (Low-congestion compiler)**.** *Given a D-diameter $(2+\epsilon)f$-connected graph $G = (V, E)$, and any k-congestion insecure algorithm. For any positive constant ϵ there exists a compiler that output a secure version of the algorithm, with additional $\tilde{O}(D + \min\{\sqrt[3]{kn^2}, \sqrt{kfn}\})$ rounds, and messages complexity $\tilde{O}(km + \sqrt[3]{f^3k^2n^4} + \sqrt{kfn^3})$.*

Proof. Some vertex, let say v_1, generate a $(k \log n)$-size set of f-degree random polynomial $\{P_{i,j}\}_{i\in[k],j\in[\log n]}$ (over $\mathbb{F}_q$, for some prime $n^2 < q < 2n^2$), and broadcasts the $\tilde{O}(kf)$-bit description of this set.

Now, each round in the original algorithm is extended into $\log n$ rounds, when the ith message M should be sent through an edge $e \in [n]^2$ on the original algorithm, the messages that replace M are $P_{i,1}(e) \oplus M, \ldots P_{i,\log n}(e) \oplus M$, and in all other edges uniformly random message is sent.

Correctness. To decrypt the incoming messages, for every chunk of $\log n$ messages $M_1, \ldots M_{\log n}$ through an edge e a vertex v computes $P_{i,1}(e) \oplus M_1, \ldots P_{i,\log n}(e) \oplus M_{\log n}$. If $M_1 = M_2 = \cdots = M_{\log n}$, it increases i by 1 and keeps the message; otherwise the message ignored. From the construction, when a meaningful message is sent, it will decrypt correctly with probability 1. And if the messages are uniformly random all of them will be same with probability $n^{-\Omega(\log n)}$.

Security. Since the adversary controls up to f edges, for every i, the set $P_{i,1}(e), \ldots P_{i,\log n}(e)$ is uniformly random, and therefore, $P_{i,1}(e) \oplus M, \ldots P_{i,\log n}(e) \oplus M$ perfectly hides the message. □

Next, we use secure broadcast to achieve a secure MST, which is the first sublinear-time and sub-cubic-message secure MST algorithm.

Lemma 6.5 (Secure MST)**.** *Fix a D-diameter $((2+\epsilon)f)$-connected graph $G=(V,E)$. For any constant $\epsilon>0$, there exists a secure $\tilde{O}(D+\min\{n^{2/3},\sqrt{fn}\})$-round MST algorithm with message complexity $\tilde{O}(m+\sqrt[3]{f^3n^4}+\sqrt{fn^3})=\tilde{O}(n^{7/3})$.*

Proof. The distributed version of each one of the $O(\log n)$ iterations in Boruvka's algorithm [NMN01] consists with the following phases:

- **Local phase.** An $\tilde{O}(1)$-congestion $\tilde{(}\sqrt{n})$-round algorithm (computing aggregate functions in vertex disjoint connected components of diameter $O(\sqrt{n})$). This phase can be secured by using our low-congestion compiler from Lemma 6.4. This requires $\tilde{O}(D+\min\{n^{2/3},\sqrt{fn}\})$ rounds and $\tilde{O}(m+\sqrt[3]{f^3n^4}+\sqrt{fn^3})$ messages.
- **Global phase.** $\tilde{O}(\sqrt{n})$ applications of unicast algorithm. This phase can be secured by using Jain's unicast algorithm from Lemma 2.10 in an $\tilde{O}(n)$-size subgraph containing the union of the $O(\kappa\log^2 n)$-size trusted collection of trees from Lemma 5.4 along with a BFS tree. This requires $\tilde{O}(D+\sqrt{n})$ rounds and $\tilde{O}(n^{1.5})$ messages.

□

Acknowledgments

This project is partially funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement No. 949083. Yuval Gelles and Ilan Komargodski are supported in part by the European Union (ERC, SCALE, 101162665). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

[AGPV90] Baruch Awerbuch, Oded Goldreich, David Peleg, and Ronen Vainish. A trade-off between information and communication in broadcast protocols. *J. ACM*, 37(2):238–256, 1990.

[BFKL21] Ruben Becker, Sebastian Forster, Andreas Karrenbauer, and Christoph Lenzen. Near-optimal approximate shortest paths and transshipment in distributed and streaming models. *SIAM J. Comput.*, 50(3):815–856, 2021.

[CCD+24] Shashwat Chandra, Yi-Jun Chang, Michal Dory, Mohsen Ghaffari, and Dean Leitersdorf. Fast broadcast in highly connected networks. In Kunal Agrawal and Erez Petrank, editors, *Proceedings of the 36th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2024, Nantes, France, June 17-21, 2024*, pages 331–343. ACM, 2024.

[Che52] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pages 493–507, 1952.

[CHGK14] Keren Censor-Hillel, Mohsen Ghaffari, and Fabian Kuhn. Distributed connectivity decomposition. In *Proceedings of the 2014 ACM symposium on Principles of distributed computing*, pages 156–165, 2014.

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

[FMSS04] Jon Feldman, Tal Malkin, Cliff Stein, and Rocco A Servedio. On the capacity of secure network coding. In *Proc. 42nd Annual Allerton Conference on Communication, Control, and Computing*, pages 63–68. Cambridge University Press, 2004.

[FP23] Orr Fischer and Merav Parter. Distributed congest algorithms against mobile adversaries. In *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing*, pages 262–273, 2023.

[GH16] Mohsen Ghaffari and Bernhard Haeupler. Distributed algorithms for planar networks II: low-congestion shortcuts, mst, and min-cut. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 202–219. SIAM, 2016.

[Gha14] Mohsen Ghaffari. Near-optimal distributed approximation of minimum-weight connected dominating set. In *ICALP (2)*, volume 8573 of *Lecture Notes in Computer Science*, pages 483–494. Springer, 2014.

[GI99] Niv Gilboa and Yuval Ishai. Compressing cryptographic resources. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 591–608. Springer, 1999.

[GK13] Mohsen Ghaffari and Fabian Kuhn. Distributed minimum cut approximation. In *DISC*, volume 8205 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2013.

[GK18] Mohsen Ghaffari and Fabian Kuhn. Distributed mst and broadcast with fewer messages, and faster gossiping. In *32nd International Symposium on Distributed Computing (DISC 2018)*, volume 121, pages 30–1. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[GKK+18] Mohsen Ghaffari, Andreas Karrenbauer, Fabian Kuhn, Christoph Lenzen, and Boaz Patt-Shamir. Near-optimal distributed maximum flow. *SIAM J. Comput.*, 47(6):2078–2117, 2018.

[GKP98] Juan A. Garay, Shay Kutten, and David Peleg. A sublinear time distributed algorithm for minimum-weight spanning trees. *SIAM J. Comput.*, 27(1):302–316, 1998.

[GL14] Mohsen Ghaffari and Christoph Lenzen. Near-optimal distributed tree embedding. In *DISC*, volume 8784 of *Lecture Notes in Computer Science*, pages 197–211. Springer, 2014.

[GP18] Robert Gmyr and Gopal Pandurangan. Time-message trade-offs in distributed algorithms. In *DISC*, volume 121 of *LIPIcs*, pages 32:1–32:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[HKN21] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. *SIAM J. Comput.*, 50(3), 2021.

[HP21] Yael Hitron and Merav Parter. General CONGEST compilers against adversarial edges. In *DISC*, volume 209 of *LIPIcs*, pages 24:1–24:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[HPY22] Yael Hitron, Merav Parter, and Eylon Yogev. Broadcast congest algorithms against eavesdroppers. In *36th International Symposium on Distributed Computing (DISC 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[HPY23] Yael Hitron, Merav Parter, and Eylon Yogev. Secure distributed network optimization against eavesdroppers. In *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*, pages 71–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2023.

[Jai04] Kamal Jain. Security based on network topology against the wiretapping attack. *IEEE Wireless Communications*, 11(1):68–71, 2004.

[Kar94] David R Karger. Using randomized sparsi cation to approximate minimum cuts. In *Proc. 5th Symp. on Discrete Algorithms*, volume 424, page 432, 1994.

[KKT15] Valerie King, Shay Kutten, and Mikkel Thorup. Construction and impromptu repair of an MST in a distributed network with o(m) communication. In *PODC*, pages 71–80. ACM, 2015.

[KP95] Shay Kutten and David Peleg. Fast distributed construction of k-dominating sets and applications. In *Proceedings of the fourteenth annual ACM symposium on Principles of distributed computing*, pages 238–251, 1995.

[KP98] Shay Kutten and David Peleg. Fast distributed construction of small k-dominating sets and applications. *J. Algorithms*, 28(1):40–66, 1998.

[LP13] Christoph Lenzen and Boaz Patt-Shamir. Fast routing table construction using small messages: extended abstract. In *STOC*, pages 381–390. ACM, 2013.

[Nan14] Danupon Nanongkai. Distributed approximation algorithms for weighted shortest paths. In *STOC*, pages 565–573. ACM, 2014.

[NMN01] Jaroslav Nešetřil, Eva Milková, and Helena Nešetřilová. Otakar boruvka on minimum spanning tree problem translation of both the 1926 papers, comments, history. *Discrete mathematics*, 233(1-3):3–36, 2001.

[NS14] Danupon Nanongkai and Hsin-Hao Su. Almost-tight distributed minimum cut algorithms. In *DISC*, volume 8784 of *Lecture Notes in Computer Science*, pages 439–453. Springer, 2014.

[NW61] C St JA Nash-Williams. Edge-disjoint spanning trees of finite graphs. *Journal of the London Mathematical Society*, 1(1):445–450, 1961.

[Par23] Merav Parter. Secure computation meets distributed universal optimality. In *FOCS*, pages 2336–2368. IEEE, 2023.

[Pel00] David Peleg. *Distributed computing: a locality-sensitive approach*. SIAM, 2000.

[PY19a] Merav Parter and Eylon Yogev. Distributed algorithms made secure: A graph theoretic approach. In *SODA*, pages 1693–1710. SIAM, 2019.

[PY19b] Merav Parter and Eylon Yogev. Low congestion cycle covers and their applications. In *SODA*, pages 1673–1692. SIAM, 2019.

[PY19c] Merav Parter and Eylon Yogev. Secure distributed computing made (nearly) optimal. In *PODC*, pages 107–116. ACM, 2019.

[Ram19] Srinivasa Ramanujan. A proof of bertrand's postulate. *Journal of the Indian Mathematical Society*, 11(181-182):27, 1919.

[Sha79a] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Sha79b] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[SHK+12] Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM J. Comput.*, 41(5):1235–1265, 2012.

[Tut61] William Thomas Tutte. On the problem of decomposing a graph into n connected factors. *Journal of the London Mathematical Society*, 1(1):221–230, 1961.

Supplementary Material

A Basic Tools and Proofs

Lemma A.1 (Chernoff's inequalities [Che52])**.** *Suppose $X_1, \ldots, X_n$ are independent random variables taking values in $\{0,1\}$. Let X denote their sum and let $\mu = \mathbb{E}[X]$ denote the sum's expected value. Then for any $\delta > 0$,*

$$\Pr[X \geq (1+\delta)\mu] \leq e^{-\delta^2\mu/(2+\delta)}, \qquad \forall \delta \geq 0$$
$$\Pr[X \geq (1-\delta)\mu] \leq e^{-\delta^2\mu/2}, \qquad \forall 1 \geq \delta \geq 0$$
$$\Pr[|X-\mu| \geq \delta\mu] \leq 2e^{-\delta^2\mu/3}, \qquad \forall 1 \geq \delta \geq 0$$

Lemma A.2 (Bertrand-Chebyshev theorem (e.g. [Ram19]))**.** *For every $n \in N$, there is a prime number q, such that $n \leq q \leq 2n$.*

A.1 Secret Sharing Scheme

Next, we define Shamir's secret sharing scheme [Sha79b].

Definition A.3. An (n,t)-secret sharing scheme consists of the following polynomial-time algorithms:

- $\mathsf{Gen}(x,n,t) \to \{s_i\}_{i\in[n]}$, a randomized algorithm that create n shares of the value $x \in X$, where each share is of size $\text{polylog}(|X|)$.
- $\mathsf{Reconstruct}(S) \to x'$, a deterministic algorithm that reconstruct x from $t+1$ shares of the value x.

That satisfies the following requirements:

- **Correctness:** For every $n, t \in \mathbb{N}$, any $x \in X$. Let $S \leftarrow \mathsf{Gen}(x,n,t)$, then for all $S' \subseteq S$ of size $t+1$, $\Pr[\mathsf{Reconstruct}(S') = x] = 1$, where the probability is over the random coins that are tossed in Gen.
- **Security:** For every (computationally unbounded) machine $\mathcal{A}$, any $n \in \mathbb{N}$, each $t < n$, and all $x \in X$. Let $S \leftarrow \mathsf{Gen}(x,n,t)$, then for every $S' \subsetneq S$ of size at most t, $\Pr[\mathcal{A}(S', X, n) = x] = \frac{1}{|X|}$.

Lemma A.4 ([Sha79b])**.** *There is a secret sharing scheme as defined in Definition A.3.*

A.2 Sparse Recovery Linear Sketching Scheme

Consider the following task: given a graph $G = (V, E)$ and a connected component $C \subseteq V$ with diameter D, the goal is to find an edge $\{u, v\}$ where $u \in C \wedge v \in V \setminus C$, while the communication is only in C. The naive solution is just notify to all the vertices in C about all the vertices in C, unfortunately, this approach requires $\Omega(|C|+D)$ rounds which might be too much. Our goal is to solve this task w.h.p. while the round complexity is $\tilde{O}(D + \sqrt{n})$.

Definition A.5 ((n,k,ϵ)-Sparse Recovery Linear Sketching Scheme)**.** An (n,k,ϵ)-sparse recovery linear sketching scheme (SRLS) is a tuple of polynomial time algorithms as follows,

- $\mathsf{Gen}(1^n, 1^k) \to \mathsf{pp}$, a randomized algorithm that outputs public parameters.

- $\mathsf{Sketch}(\mathsf{pp}, x) \to \sigma$, a deterministic sketching algorithm that receives the public parameters and a bit vector of size n.
- $\mathsf{Recover}(\sigma') \to x'$, a deterministic recovery algorithm, that outputs a bit vector of size n.

That satisfies the following properties:

- **Sketching (compactness):** Typically $k \ll n$, and the requirement is to store roughly k bits. Formally, $|\mathsf{pp}|, |\sigma| \in k \cdot \mathrm{polylog}(n)$, and the hamming weight of $\mathsf{Recover}(\sigma)$ is in $k \cdot \mathrm{polylog}(n)$, for any possible σ
- **Bitwise linearity:** For every $x, y \in \{0,1\}^n$, $\mathsf{Sketch}(\mathsf{pp}, x) \oplus \mathsf{Sketch}(\mathsf{pp}, y) = \mathsf{Sketch}(\mathsf{pp}, x \oplus y)$.
- (k, ϵ)**-Recovery (correctness):** For every $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, if the hamming distance of x is at most k, then for all $i \in [n]$,

$$\Pr\left[x_i = 1 \wedge x'_i = 0 \,\middle|\, \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Gen}(1^n, 1^k) \\ x' = \mathsf{Recover}\left(\mathsf{Sketch}(\mathsf{pp}, x)\right) \end{array}\right] \leq \epsilon,$$

  where the probability is over the choice of pp.

Lemma A.6. *For every n and every $2 \leq k \leq n/4$ there is $(n, k, 1 - e^{-1})$-SRLS.*

Proof. Let

- $\mathsf{Gen}(1^n, 1^k)$:
  1. Find a prime $q \in [n, 2n]$ (which always exists due to Bertrand-Chebyshev theorem).
  2. Sample a random polynomial g over $\mathbb{F}_q$, where the degree of g is $k - 1$.
  3. Output $\mathsf{pp} = (g, q)$.
- $\mathsf{Sketch}(\mathsf{pp}, x = (x_1, \ldots, x_n))$:
  1. Create a $\log n$-bit vector σ of length $2k$, and fill it with zeros.
  2. For every $i \in [n]$, if $x_i = 1$, set $\sigma_{g(i)_{\mathsf{mod}2k}} = \sigma_{g(i)_{\mathsf{mod}2k}} \oplus i$.
  3. Output σ.
- $\mathsf{Recover}(\sigma)$:
  1. For every $i \in [n]$, if $i \in \sigma$ set $x_i = 1$, else, set $x_i = 0$.
  2. Output $x = (x_1, \ldots, x_n)$.

The sketching property satisfied from the construction. For bitwise linearity, let's prove by induction on i, where $\sigma_{x,i}$ is the state of σ after i iterations on the sketching procedure on input x.

- **Base case** $(i = 0)$: Trivially, $\sigma_{x,0} = \sigma_{y,0} = \sigma_{x \oplus y, 0} = 0^2 k$.
- **Inductive step**: Assume that $\sigma_{x,i-1} \oplus \sigma_{y,i-1} = \sigma_{x \oplus y, i-1}$, then by definition $\sigma_{x,i} \oplus \sigma_{y,i} = \sigma_{x \oplus y, i} \Leftrightarrow \sigma_{x,i,g(i)_{\mathsf{mod}2k}} \oplus \sigma_{y,i,g(i)_{\mathsf{mod}2k}} = \sigma_{x \oplus y, i, g(i)_{\mathsf{mod}2k}}$,
  - $x_i = 0, y_i = 0$:

$$\sigma_{x,i,g(i)_{\mathsf{mod}2k}} \oplus \sigma_{y,i,g(i)_{\mathsf{mod}2k}} = \sigma_{x,i-1,g(i)_{\mathsf{mod}2k}} \oplus \sigma_{y,i-1,g(i)_{\mathsf{mod}2k}} =$$
$$\sigma_{x \oplus y, i-1, g(i)_{\mathsf{mod}2k}} = \sigma_{x \oplus y, i, g(i)_{\mathsf{mod}2k}}$$

– $x_i = 0, y_i = 1$:

$$\sigma_{x,i,g(i)_{\mathsf{mod}2k}} \oplus \sigma_{y,i,g(i)_{\mathsf{mod}2k}} = \sigma_{x,i-1,g(i)_{\mathsf{mod}2k}} \oplus (\sigma_{y,i-1,g(i)_{\mathsf{mod}2k}} \oplus i) = \sigma_{x\oplus y,i-1,g(i)_{\mathsf{mod}2k}} \oplus i = \sigma_{x\oplus y,i,g(i)_{\mathsf{mod}2k}}$$

– $x_i = 1, y_i = 0$:

$$\sigma_{x,i,g(i)_{\mathsf{mod}2k}} \oplus \sigma_{y,i,g(i)_{\mathsf{mod}2k}} = (\sigma_{x,i-1,g(i)_{\mathsf{mod}2k}} \oplus i) \oplus \sigma_{y,i-1,g(i)_{\mathsf{mod}2k}} = \sigma_{x\oplus y,i-1,g(i)_{\mathsf{mod}2k}} \oplus i = \sigma_{x\oplus y,i,g(i)_{\mathsf{mod}2k}}$$

– $x_i = 1, y_i = 1$:

$$\sigma_{x,i,g(i)_{\mathsf{mod}2k}} \oplus \sigma_{y,i,g(i)_{\mathsf{mod}2k}} = (\sigma_{x,i-1,g(i)_{\mathsf{mod}2k}} \oplus i) \oplus (\sigma_{y,i-1,g(i)_{\mathsf{mod}2k}} \oplus i) = \sigma_{x\oplus y,i-1,g(i)_{\mathsf{mod}2k}} = \sigma_{x\oplus y,i,g(i)_{\mathsf{mod}2k}}$$

For the recovery property, for any i such that $x_i = 1$,

$$\Pr\left[x'_i = 1 \middle| \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Gen}(1^n, 1^k) \\ x' = \mathsf{Recover}\left(\mathsf{Sketch}(\mathsf{pp}, x)\right) \end{array}\right] \geq \Pr\left[\forall j \neq i, x_j = 0 \vee g(j)_{\mathsf{mod}2k} \neq g(i)_{\mathsf{mod}2k}\right].$$

Since g degree is $k-1$, and the hamming weight of x is at most k it means that the $g(j)$'s for all the x_j's that equal to 1 distributed independently and uniformly at random in $\mathbb{F}_q$. Therefore,

$$\Pr\left[\forall j \neq i, x_j = 0 \vee g(j)_{\mathsf{mod}2k} \neq g(i)_{\mathsf{mod}2k}\right] \geq \left(1 - \Pr\left[g(j)_{\mathsf{mod}2k} = g(i)_{\mathsf{mod}2k}\right]\right)^k.$$

Since $\Pr\left[g(i)_{\mathsf{mod}2k} = x\right] \leq \frac{1}{2k} + \frac{1}{q}$ and $k \geq 2$,

$$\left(1 - \Pr\left[g(j)_{\mathsf{mod}2k} = g(i)_{\mathsf{mod}2k}\right]\right)^k \geq \left(1 - \frac{1}{2k} - \frac{1}{q}\right)^k \geq \left(1 - \frac{1}{2k} - \frac{1}{n}\right)^k \geq \left(1 - \frac{1}{k} + \frac{1}{4k}\right)^k \geq \left(1 - \frac{1}{k} + \frac{1}{2k^2}\right)^k$$

and for every $x > 0, e^x \leq 1 - x + \frac{x^2}{2}$,

$$\Pr\left[x'_i = 1 \middle| \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Gen}(1^n, 1^k) \\ x' = \mathsf{Recover}\left(\mathsf{Sketch}(\mathsf{pp}, x)\right) \end{array}\right] \geq \left(1 - \frac{1}{k} + \frac{1}{2k^2}\right)^k \geq e^{-1}$$

□

Corollary A.7. *For every n, every $k \leq n/4$, and every $\kappa \in \Omega(1) \cap O(\log n)$ there is $(n, k, n^{-\Omega(\kappa)})$-*SRLS*.

Proof. Run the above schema $\kappa \log n$ times independently in parallel, and union the recovery results. The probability that any i such that $x_i = 1$ is not exists in all of them is $(1 - e^{-1})^{\kappa \log n} = n^{-\Omega(\kappa)}$. □

A.3 Proof of Lemma 2.4

We use the cut-counting result of Karger [Kar94].

Lemma A.8. *Given a λ-connected graph and a positive integer $k \in \mathbb{N}$, the number of cuts with at most $k \cdot \lambda$ crossing edges is at most n^{2k}.*

Given a cut $(S, V \setminus S)$, apply Chernoff's bound (Lemma A.1):

$$\Pr\left[|(S,V\setminus S)|\leq \left(1-\frac{\sqrt{2}}{\kappa^{0.1}}\right)|(S,V\setminus S)|p\right] \leq e^{-\frac{|(S,V\setminus S)|p}{\kappa^{0.2}}}$$

By Lemma A.8 and the union bound,

$$\Pr\left[\exists S\colon |(S,V\setminus S)|\leq \left(1-\frac{\sqrt{2}}{\kappa^{0.1}}\right)|(S,V\setminus S)|p\right] \leq \sum_{\alpha=1}^{n^2} n^{2\alpha}e^{-\frac{\alpha\lambda p}{\kappa^{0.2}}} \leq$$
$$\sum_{\alpha=1}^{n^2} n^{(2-\kappa^{0.8})\alpha} \leq \sum_{\alpha=1}^{n^2} n^{-\alpha\kappa^{0.6}} \leq \sum_{\alpha=1}^{\infty} n^{-\alpha\kappa^{0.6}} = \frac{1}{n^{\kappa^{0.6}}-1} \in n^{-\Omega(\kappa)}.$$

For the second result we again apply Chernoff's bound. We get

$$\Pr\left[|F[p]|\geq \left(1+\frac{\sqrt{3}}{\kappa^{0.1}}\right)|F|p\right] \leq e^{-\frac{|F|p}{\kappa^{0.9}}} \leq n^{-|F|\kappa^{0.1}} \in n^{-\Omega(1)}.$$

A.4 Proof of Lemma 5.5

To prove the lemma, we demonstrate how to construct an MST containing only edges from $\hat{\mathbf{F}}$.

Claim A.9. *Let $e_1, \ldots, e_m$ be a list of all edges sorted by w, with the following property: if $w(e) = w(e')$ and $e \in \hat{\mathbf{F}}$ while $e' \notin \hat{\mathbf{F}}$, then e precedes e' in the ordering. The MST produced by running Kruskal's algorithm on this order contains only edges from $\hat{\mathbf{F}}$.*

Proof. Let $\{u, v\} \in E \setminus \mathbf{F}_{w(u,v)}$ be an edge considered by Kruskal's algorithm for inclusion in the MST. Since $\mathbf{F}_{w(u,v)}$ contains a maximal forest, there exists a path $u - \cdots - v \in \mathbf{F}_{w(u,v)}$. Let $\{u', v'\}$ be an edge on this path. When the algorithm decides whether to include $\{u', v'\}$ in the MST, there are two possibilities:

- The algorithm includes $\{u', v'\}$ in the MST.
- $\{u', v'\}$ creates a cycle, so u' and v' are already in the same connected component.

In either case, after this step, u' and v' are in the same connected component. Since each edge in the path has a lower index in the sorted list, u and v must be in the same connected component, and thus the algorithm will reject $\{u, v\}$. □

A.5 Proof of Claim 4.6

Let X_j be the number of non-isolated components after j iterations. Each non-isolated component is a satellite with probability 1/2, and from Claim 4.5 it will find an edge to another component with probability $1 - n^{-\Omega(\kappa)}$, which is center with probability 1/2. Therefore, a non-isolated component

will merge to a center component with probability at least $1/2 \cdot 1/2 \cdot (1 - o(1)) \geq 1/5$. So we can bound the expected number of non-isolated components with respect to the previous iteration,

$$\mathbb{E}\left[X_j | X_{j-1} = k\right] \leq \frac{4k}{5}.$$

Since at the beginning each vertex may be a component, $X_0 \leq n$ and,

$$\mathbb{E}\left[X_j\right] \leq n\left(\frac{4}{5}\right)^j.$$

By Markov's inequality,

$$\Pr\left[X_{\kappa \log n} \neq 0\right] \leq \mathbb{E}\left[X_{\kappa \log n}\right] \leq n\left(\frac{4}{5}\right)^{\kappa \log n} \in n^{-\Omega(\kappa)}.$$

B Fractional Tree Packing - Revisited

In this section we revisit the result of [CHGK14] which create a collection of spanning trees that are fractionally disjoint, although our result is not novel, we provide a clearer statement and a more intuitive proof. First we repeat the definition,

Following their approach, we rely on classical results by Tutte [Tut61] and Nash-Williams [NW61], which relate edge connectivity to the number of edge-disjoint spanning trees. Specifically, any $(2+\epsilon)f$-connected graph contains at least $\left(1 + \frac{\epsilon}{3}\right) f$ edge-disjoint spanning trees. Our statement is,

Theorem B.1 (Similar to Theorem 1.3 in [CHGK14])**.** *Given a D-diameter $(2+\epsilon)f$-connected graph $G = (V, E)$, for any constant $\epsilon \in (0, 1]$. There is a distributed algorithm that computes $\mathcal{T}$ which is an f-fractionally disjoint collection of $\Theta(f \log n)$ trees. The algorithm's overall complexity matches that of executing a distributed MST algorithm on graph with $\Theta(f \log n)$ weight classes, repeated $\Theta(f \log n)$ times.*

Remark B.2. In our work we are focusing only on $f \in \operatorname{polylog} n$, but, as shown by Censor et al. [CHGK14], we can divide the graph into $\tilde{O}(f)$ subgraphs, each with $(2 + \epsilon') \log^2 n$-connectivity.

To prove Lemma 5.4 we start by defining the "load" of each edge in the collection, formally:

$$x_e = \left|\{T_i \in \mathcal{T} \mid e \in T_i\}\right|,$$

where the weight of each edge is $c_e = 2^{x_e}$. The algorithm that computes T is simple:

1. Initiate $\mathcal{T} = \emptyset$.
2. For $i \in \left[\left(1 + \frac{3}{\epsilon}\right) f \log n\right]$ times:
   (a) Run MST algorithm to compute T_i.
   (b) Update $\mathcal{T} \leftarrow \mathcal{T} \cup \{T_i\}$, and recalculate the weights.
3. Output $\mathcal{T}$.

First, we compute the minimum total weight of a subset F that intersects every tree in the collection, i.e., satisfies the condition: $\sum_{e\in F} x_e \geq \left(1+\frac{3}{\epsilon}\right) f \log n$.

$$\sum_{e\in F}\sum_{i=1}^{x_e} 2^{i-1} \geq f \sum_{i=1}^{\left(1+\frac{3}{\epsilon}\right)\log n} 2^{i-1} = f\left(n^{1+\frac{3}{\epsilon}}-1\right),$$

where the inequality holds because for integers $x > y$, $2^x + 2^y > 2^{x-1} + 2^{y+1}$, and the equation is sum of geometric series.

Next, we analyze the cumulative weights of the trees in $\mathcal{T}$. From Lemma 2.2 there is at least $(1+\epsilon/3)f$ disjoint spanning trees, meaning that the average weight of those trees is an upper bound to the MST weight. Let a_i be the weight of this disjoint trees after the ith iteration. Since for every i, we have $w(T_i) \leq \frac{a_{i-1}}{(1+\epsilon/3)f}$ and by definition $a_i \leq a_{i-1} + w(T_i)$, it follows that:

$$a_i = a_0\left(1+\frac{1}{(1+\epsilon/3)f}\right)^{i-1} \leq a_0 \cdot \exp\left(\frac{i-1}{(1+\epsilon/3)f}\right),$$

and by definition $a_0 = (n-1)(1+\epsilon/3)f$. Therefore, we can compute an upper bound to the total size of the MSTs,

$$\sum_{i=1}^{\left(1+\frac{3}{\epsilon}\right)f\log n} w(T_i) \leq \sum_{i=1}^{\left(1+\frac{3}{\epsilon}\right)f\log n} \frac{a_0}{(1+\epsilon/3)f}\cdot\exp\left(\frac{i-1}{(1+\epsilon/3)f}\right) \leq n\sum_{i=1}^{\left(1+\frac{3}{\epsilon}\right)f\log n}\exp\left(\frac{i-1}{(1+\epsilon/3)f}\right),$$

and by applying the formula for geometric series,

$$\sum_{i=1}^{\left(1+\frac{3}{\epsilon}\right)f\log n} w(T_i) \leq n\frac{\exp\left(\frac{3\log n}{\epsilon}\right)-1}{\exp\left(\frac{1}{(1+\epsilon/3)f}\right)-1} \leq (1+\epsilon/3)fn^{1+\frac{3\ln 2}{\epsilon}}.$$

And the ratio between the weight of the edges in F to the total weight is,

$$\frac{f\left(n^{1+\frac{3}{\epsilon}}-1\right)}{(1+\epsilon/3)fn^{1+\frac{3\ln 2}{\epsilon}}} \geq \frac{(1-\epsilon/6)n^{1+\frac{3}{\epsilon}}}{(1+\epsilon/3)n^{1+\frac{3\ln 2}{\epsilon}}} \geq (1-\epsilon/2)n^{\frac{3(1-\ln 2)}{\epsilon}} \in \omega(1).$$

This leads to a contradiction, implying that the condition $\sum_{e\in F} x_e \geq \left(1+\frac{3}{\epsilon}\right) f\log n$ cannot hold.