Practical Provably Secure Flooding for Blockchains

Chen-Da Liu-Zhang[1][0000−0002−0349−3838], Christian Matt[2][0000−0001−5900−336X], Ueli Maurer[3], Guilherme Rito[3][0000−0002−0080−8670], and Søren Eller Thomsen[4][0000−0002−6931−4740]

[1] NTT Research, USA
chen-da.liuzhang@ntt-research.com
[2] Concordium, Zurich, Switzerland
cm@concordium.com
[3] Department of Computer Science, ETH Zurich, Switzerland
{maurer, gteixeir}@inf.ethz.ch
[4] Concordium Blockchain Research Center, Aarhus University, Denmark
sethomsen@cs.au.dk

Abstract. In recent years, permisionless blockchains have received a lot of attention both from industry and academia, where substantial effort has been spent to develop consensus protocols that are secure under the assumption that less than half (or a third) of a given resource (e.g., stake or computing power) is controlled by corrupted parties. The security proofs of these consensus protocols usually assume the availability of a network functionality guaranteeing that a block sent by an honest party is received by all honest parties within some bounded time. To obtain an overall protocol that is secure under the same corruption assumption, it is therefore necessary to combine the consensus protocol with a network protocol that achieves this property under that assumption. In practice, however, the underlying network is typically implemented by flooding protocols that are not proven to be secure in the setting where a fraction of the considered total weight can be corrupted. This has led to many so-called eclipse attacks on existing protocols and tailor-made fixes against specific attacks.

To close this apparent gap, we present the first practical flooding protocol that provably delivers sent messages to all honest parties after a logarithmic number of steps. We prove security in the setting where all parties are publicly assigned a positive weight and the adversary can corrupt parties accumulating up to a constant fraction of the total weight. This can directly be used in the proof-of-stake setting, but is not limited to it. To prove the security of our protocol, we combine known results about the diameter of Erdős–Rényi graphs with reductions between different types of random graphs. We further show that the efficiency of our protocol is asymptotically optimal.

The practicality of our protocol is supported by extensive simulations for different numbers of parties, weight distributions, and corruption strategies. The simulations confirm our theoretical results and show that messages are delivered quickly regardless of the weight distribution,

whereas protocols that are oblivious of the parties' weights completely fail if the weights are unevenly distributed. Furthermore, the average message complexity per party of our protocol is within a small constant factor of such a protocol.

Keywords: flooding networks · peer-to-peer networks · blockchain · network layer · multicast

1 Introduction

1.1 Motivation

Since Nakamoto proposed the first decentralized permisionless blockchain protocol [32], a significant line of works has been done. In such protocols, one considers a setting where different parties are weighted according to how much of a resource they own (mining power, stake, space, etc.), and security relies on the fact that a certain fraction of the total weight (typically more than the majority, or two thirds) is owned by the honest parties.

Current blockchain protocols typically are proven secure assuming the availability of a *multicast* network, which allows each party to distribute a value among the parties within some delivery time Δ (see e.g. [3, 12, 14–16, 20, 34, 36]). However, very little attention has been devoted to the construction of provably secure multicast networks themselves.

In practice, the multicast network is typically implemented via a message-diffusion mechanism, where in order for a party P to distribute a message, P sends the message to a subset of its neighbors, who then forward the message to their neighbors and so on. The idea is that if the graph induced by the honest parties is connected, the message will reach all the honest parties, and if the graph has low diameter, it will reach all honest parties after only a few iterations. Indeed, there have been works that study how to randomly select the neighbors so that the induced graph remains connected with small diameter after removing corrupted nodes (see e.g. [24, 30, 37]).

Unfortunately, to the best of our knowledge, currently analyzed diffusion mechanisms do not consider weighted parties, and therefore can only be proven secure when a certain constant fraction of the parties is honest (in particular it is not enough to assume a fraction of the total weight is owned by honest parties). This means that when such a message diffusion mechanism is used to build a blockchain, the overall protocol relies on *both* the constant-honest-fraction-of-weight assumption *and* the constant-honest-fraction-of-parties assumption.

Note that for a fixed weight distribution, a bound on the corrupted weight also implies a bound on the number of parties that can be corrupted, where this maximum is achieved by greedily corrupting parties with the least weight first. Hence, current multicast protocols could in principle also be used assuming only a bound on the corrupted weight. However, the message complexity of such protocols is inversely proportional to the guaranteed honesty ratio. That is, to

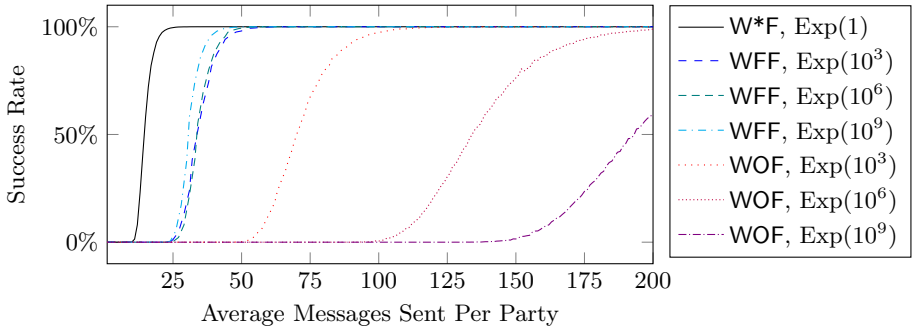

Fig. 1. Comparison of our WFF protocol with a weight oblivious protocol WOF that chooses a fix number of random neighbors independently of their weight. The simulations are for $n = 1024$ parties with exponential weight distributions $\mathrm{Exp}(r)$ where the heaviest party's weight is r times the lightest party's weight. Note that for $\mathrm{Exp}(1)$, WFF and WOF are identical. For each setting, we consider a 50% corruption threshold, and a greedy corruption strategy where lighter parties are corrupted first. Each simulation was repeated 10 000 times and the success rate measures how often all parties received a single sent message.

still guarantee security under more corrupted parties, the remaining parties have to send to more neighbors. In particular, this means that in many of the current weight distributions where there are very few people owning a large fraction of the total weight, but thousands of parties owning a tiny little fraction of the weight, the incurred concrete message complexity to achieve security significantly blows up (see an example in Fig. 1, where even for large sizes of neighborhood sizes, the protocol fails).

The need for a practically efficient multicast network secure solely relying on the constant-honest-fraction-of-weight assumption is therefore apparent.

1.2 Our Contributions

In this work, we investigate provably secure protocols that implement a multicast network for the weighted setting, relying solely on the constant-honest-fraction-of-weight assumption. Additionally, we are interested in protocols that are concretely efficient. In short, we explore the following natural questions:

> *Is there a provably secure multicast protocol in the weighted setting, assuming only a constant fraction of honest weight? And if so, is there a practically efficient one?*

We answer both of these questions in the affirmative by presenting the first multicast protocol WFF (weighted fan-out flooding) that relies solely on the constant-honest-fraction-of-weight assumption, and evaluate its practical efficiency by performing various simulations. More concretely, we prove the following theorem:

Theorem 1 (Informal). *Let κ be a security parameter, n be the number of parties, and $\gamma \in [0,1]$ be the fraction of the total weight that is guaranteed to belong to honest parties. Further, let $\delta_{Channel}$ be an upper bound on the delays of the underlying point-to-point channels. Then, WFF is a secure flooding protocol with maximal delay $\Delta \coloneqq \left(7 \cdot \log\left(\frac{6n}{\log(n)+\kappa}\right)+2\right) \cdot \delta_{Channel}$ and message complexity $2n \cdot \frac{\log(n)+\kappa}{\gamma}$.*

Note that the maximum delay and the message complexity in the theorem are independent of the weight distribution. By naturally assigning the weights to corresponding stake quantities, the achieved guarantees match those required in previous proof-of-stake blockchain protocols (see e.g. [12,15,16]), and therefore our protocol can be used to build a blockchain protocol from point-to-point channels without the need for any additional assumption apart from those needed in the blockchain protocol itself.

Asymptotic optimality and practicality. Our protocol has the property that 1) parties accumulating large amounts of weight need to send to more parties, and 2) the number of parties that each party sends to increases logarithmically in the total number of parties. We prove that both properties are inherent for secure flooding protocols, meaning that Theorem 1 is asymptotically optimal. Concretely, for the first point, if a small set S (say, of constant size) accumulates more than a γ-fraction of the weight, then this set necessarily needs to send at least to a linear number $\Theta(n)$ of parties.

This means it is undesirable to have parties with very small weight and also to have parties with a huge weight. A simple way to mitigate this in practice is to exclude parties with less than $W_{\min}$ weight and cap the maximal weight to $W_{\max}$. This means if we use the flooding for a proof-of-stake blockchain, that parties with a huge amount of stake need to split their stake over several nodes such that none has more than $W_{\max}$ weight. Parties with very little stake can still obtain data from other nodes by requesting data from them periodically. We discuss this further below.

Simulations. We use simulations to evaluate the practicality of our provably secure protocol. The simulations confirm our theoretical results and also show that our protocol is practical: Messages are diffused quickly to all parties with high success probability even when weights are unevenly distributed. On the other hand, as our simulations also show, prior protocols—oblivious of the parties' weights—fail completely for neighborhood sizes for which our provably secure protocol succeeds (see Fig. 1). This in particular means that our protocol achieves the necessary security guarantees considerably fewer number of messages than current (weight-oblivious) protocols.

1.3 Model and Assumptions

Network and corruption model. We assume all parties have access to an underlying network that allows them to establish point-to-point channels to other parties.

We further assume each party p is publicly assigned a weight $W_p > 0$ and an adversary can corrupt parties accumulating at most a constant fraction $1 - \gamma$ of the total weight. For simplicity, we consider static corruptions in our proofs, but using the techniques from [30], all our results can be extended to security against delayed adaptive adversaries (adversaries for which there is a delay from the time they decide to corrupt a party until the party is effectively controlled by the adversary). Intuitively, if a corruption takes longer than the duration from the earliest point in time an adversary can learn the neighbors of a party till the neighbors are guaranteed to have resent the message to other parties, then adaptivity does not help the adversary prevent delivery of any messages. However, there is significant overhead involved in proving this because the adversary can still dynamically decide how many parties are left to guarantee delivery for. This is why we only present proofs for a static adversary and refer to [30] for techniques for how to prove such statement.

Realising public weights from resource assumptions. Proof-of-stake blockchains rely on a constant fraction of the stake being honest (typically more than $1/2$ [14, 15] or more than $2/3$ [12]). Furthermore, a blockchain itself provides a ledger accessible by all parties describing how much stake each party owns. Hence, it is immediate how to assign weights to parties by simply accessing the ledger in order to instantiate the weights for our protocols.

To achieve a weight distribution for blockchain protocols that rely on a constant fraction of the computational resources being honest [20, 34–36] one can make use of the techniques for committee selection for such setting [35, 36]. The idea behind this is that for long fragments of a chain with high chain-quality, the distribution of block creators is similar to the distribution of computational resources among parties. Hence, this distribution translates directly to a weight distribution publicly available to all parties. For techniques to achieve a high chain-quality, see [34].

Delivery to zero-weight parties. While we assume that all parties have positive weight and parties with zero weight cannot contribute to the security of the protocol, it is still desirable in practice to allow such parties to obtain the state of the system. This can be achieved, e.g., by letting such parties fetch missing data from other nodes. We discuss some options in the full version of this work [26].

Static versus dynamic weight. For simplicity, we consider for this paper the static-weight setting, in which the weight of all parties remains fixed. When weight is instantiated with the stake in a proof-of-stake, this might appear unrealistic. This is, however, not a real limitation of our protocol when combined with such a blockchain. For example in [15], to prove their protocol secure for a dynamic stake, the authors divide time into epochs where the stake used for producing blocks remains unchanged and additionally make assumptions on the speed that stake can between epochs. In their proofs, they note that all parties agree on the stake distribution in a previous epoch. We note that our proofs only rely on the

weight being static for the propagation of a single message, and the time it takes to propagate a message is very small compared to such epochs.

Practicality of complete network. We note that the assumption that any two parties can establish a point-to-point connection between each other is indeed a reasonable assumption, e.g., in the proof-of-stake setting: Parties who want to participate in the protocol first need to register their node, see, e.g., [6]. This registration process can include the node's IP address and further required information that allows other nodes to establish a connection with that node.

1.4 Technical Overview

Flooding protocol skeleton. Our protocol follows the basic structure of previous flooding protocols: When a party p receive a message m for the first time, p samples a set of neighbors N from the party set $\mathcal{P}$, according to some probability distribution $\mathcal{N}_p$. The party then forwards the message m to all parties in N. The crucial variable of this protocol is the distribution $\mathcal{N}_p$, i.e., how parties select their peers.

Remark 1. In most practical blockchain implementations, parties do not resample their peers for every message, but keep the connections over an extended period of time [22, 29]. We note that our protocol can also be used in such a fashion and all our results can be translated to such a setting. The reason for resampling peers often is that against a delayed adaptive adversary [30], security can only be guaranteed if the corruption delay is longer than the time peers keep their connections. Hence, resampling more often provides better security guarantees.

Dependency of neighborhood selection on weight distribution. It is clear that to achieve efficient results, one must make use of the overall weight distribution to decide whether a party p_i forwards the message to party p_j. What is perhaps less clear, is what the required *amount* of dependency is. We here argue intuitively that the neighborhood selection must depend (at least) on both the weights of p_i and p_j: Consider a weight distribution where p_i's weight is overwhelming, and there are many parties with very little weight (including p_j). In this case, the adversary has corruption budget to corrupt all parties except for p_i and p_j. Therefore, in order to guarantee that an honest p_j receives the message, p_i must send to that party with probability 1. Consequently, the neighborhood selection distribution $\mathcal{N}_{p_i}$ must depend on p_i's weight. It follows via an analogous argument that p_i must send to p_j if the latter's weight is overwhelming. Hence, the probability to choose p_j in $\mathcal{N}_p$ must also depend on p_j's weight.

A simple inefficient solution. From the above observations, we see that the neighborhood distribution must depend on both the weights W_i of p_i and W_j of p_j. A simple idea is to let each party p_i internally emulate W_{p_i} parties, and then run a traditional unweighted flooding protocol among $W = \sum_p W_p$ nodes, where two nodes are connected with some probability ρ. By properties of Erdős–Rényi

graphs, this leads to a secure flooding protocol [30]. Note that the probability that a node from p_i is connected to a node from p_j depends on both weights W_{p_i} and W_{p_j}, namely $1-(1-\rho)^{W_{p_i}\cdot W_{p_j}}$.

However, the resulting protocol is highly inefficient, since it has a message complexity that depends on the total sum of the weights W, rather than the number of parties. Note that in current proof-of-stake systems, the total stake is in the order of billions, so any dependency on the total weight is highly undesirable.

Scaling invariance. The simple protocol from above not only is inefficient if the total weight is large, it also has the undesirable property that the efficiency depends on the "unit" of the weight: If we multiply everybody's weight by 100, the overall number of messages increases substantially, even though this scaling has no effect on the possible corruptions. We thus postulate that practical protocols should be invariant under such weight scalings.

A simple fix seems to be to normalize the weight distribution by dividing every party's weight by the weight of the lightest party. This, however, introduces two issues: First, since the number of internally emulated nodes must be an integer, this division leads to rounding issues, with implications for the security argument. Secondly, introducing an additional extremely light party now has a massive impact on the efficiency, even though this additional party does not substantially change the possible corruptions.

A first theoretical protocol. Our first technical theoretical contribution is a new simple way to choose the neighbors in the flooding protocol. More precisely, we generalize the approach above and show that it is actually enough to emulate a number of nodes that is proportional to the total number of parties (rather than the total weight).

For that, we introduce the notion of an emulation-function $\mathtt{E}:\mathcal{P}\to\mathbb{N}\setminus\{0\}$. According to the emulation function, we let each party p internally emulate $\mathtt{E}(p)\geq 1$ different nodes, in a graph consisting of $n_{\mathtt{E}}:=\sum_p \mathtt{E}(p)$ nodes. As explained above, the basic idea is to create an Erdős–Rényi graph on the emulated graph with $n_{\mathtt{E}}$ nodes and edge-probability ρ. Then, we say that a party p_i forwards the message to p_j if *any* of the emulated nodes from p_i is connected to *any* of the emulated nodes from p_j. This means that the probability that p_i forwards the message to p_j is $1-(1-\rho)^{\mathtt{E}(p_i)\cdot\mathtt{E}(p_j)}$.

We then consider the emulation function $\mathtt{E}(p)=\lceil\alpha_p\cdot n\rceil$, where α_p is p's fraction of the total weight. That is, we let each party emulate a number of nodes proportional to the number of parties scaled by the party's relative weight. Note that the ceiling ensure that each party emulates at least one node. We then prove that by choosing ρ appropriately such that the unweighted subgraph emulated by honest parties remains connected with low diameter, we obtain a flooding protocol with message complexity $O((\log(n)+\kappa)\cdot n\cdot\gamma^{-1})$ and time complexity $O(\log(n)\cdot\delta_{\text{Channel}})$.

A practical protocol. Although the method described above is intuitive and gives us asymptotically good complexities, it is very far from being practical. In

particular, the protocol requires every party to locally flip $\Omega(n)$ coins for each message. Similar to current protocols deployed in practice, we would like to have a protocol that instead chooses a *fixed* set of neighbors (possibly dependent on the weight distribution, but nothing else), and provide provable security for it.

We propose a protocol where each party p chooses to send to $K = k \cdot \mathtt{E}(p) = k \cdot \lceil \alpha_p \cdot n \rceil$ distinct parties (for a parameter k), according to a weighted sampling without replacement [7]. More precisely, p chooses K parties, where the probability to choose a certain tuple[5] of parties $(q_1, \ldots, q_K)$ (among the set of parties $\mathcal{P} \setminus \{p\}$) is

$$\Pr[(q_1, \ldots, q_K)] = \prod_{i=1}^{K} \frac{\mathtt{E}(q_i)}{n_{\mathtt{E}} - \mathtt{E}(p) - \mathtt{E}(q_1) - \cdots - \mathtt{E}(q_{i-1})}.$$

We show that this practical protocol has the same asymptotic guarantees as the first protocol above.

Importance of emulation function. Even though that this protocol is so simple that it can be described in a few lines, it is by no means trivial. In fact, it is crucial for the correctness of the protocol that the emulation function is used to determine both the number of neighbors *and* the distribution of these neighbors.

To see that it is crucial to use the emulation function to decide how many neighbors each party should choose, consider a small change to the protocol, namely send to $K = k \cdot \alpha_p \cdot n$ parties (instead of $K = k \cdot \mathtt{E}(p)$). Now, consider a sender p with a small fraction of the total weight α_p, and let us estimate the parameter k to ensure that this p sends to at least one honest party. As any party potentially could be corrupt it must be that p sends to more than just one neighbor. Hence, it must be that $k > \frac{1}{\alpha_p \cdot n}$, just to ensure this very minimal requirement. A rough bound on the message complexity of such protocol would be $\sum_{p'} k \cdot \alpha_{p'} \cdot n > \frac{1}{\alpha_p}$, which is impractical if α_p is small.

To see that it is crucial to weigh the selection of neighbors with the emulation function, we consider another small change to the protocol, namely to select parties weighted by their weight instead of the emulation function. Now, consider a weight distribution where just one party p has a very small fraction of the total weight and all others having roughly equal weight. Note, that for any party choosing less than n neighbors the probability that p is chosen as a neighbor becomes arbitrarily small for a decreasing α_p. Hence, to ensure that p receives a message this would induce a quadratic message complexity which is impractical.

Security proof. Proving security of such a protocol in the weighted setting directly is non-trivial for two reasons: First, the choices of whether to send to a neighbor or not are not independent. Secondly, the fact that the choices are according to an arbitrary weight distribution makes the analysis considerably harder than traditional graph-theoretic results that consider the non-weighted setting. Instead of providing a direct graph-theoretic analysis, we give a security proof via a

[5] The probability to choose the unordered neighborhood set $N = \{q_1, \ldots, q_K\}$ is the sum over the probabilities of all permuted tuples.

sequence of intermediate protocols, essentially relating the success probability of the first protocol above based on Erdős–Rényi graphs to the practical protocol. This leads to Theorem 1. Due to space constraints, many proofs are left out of this version. We refer to the full version of this paper [26] for these.

1.5 Current State of the Art and Related Work

Flooding networks in a Byzantine setting. [24] was the first to relate probabilistic gossiping to the connectivity of the induced graph. They considered $(1-\gamma)\cdot n$ out of n parties failing and showed that each party needs to forward a message with probability $\rho > \frac{\log(n)+\kappa}{\gamma \cdot n}$ to any other party to ensure that messages are delivered to all non-failing parties with a probability overwhelming in κ.

[30] observed that against an adversary capable of adaptively corrupting up to t parties, any flooding network where each party sends to less than t neighbors is inherently insecure (an adversary can simply corrupt all neighbors of a sender). To mitigate this problem and achieve a protocol secure against a Byzantine adaptive adversary, [30] formalized the notion of a delayed adversary (informally introduced by [35]) for which there is a delay from the time the adversary decides to corrupt a party until the party is effectively controlled by the adversary. In this setting, they showed that against an adversary delayed for the time it takes to send a message plus the time it takes to resend a message, it is sufficient to on average send to $\Omega((\log(n)+\kappa)\cdot\gamma^{-1})$ neighbors to achieve a flooding protocol that with an overwhelming probability in κ has $O(\log(n))$ round complexity for n parties with at most $(1-\gamma)\cdot n$ of the parties being corrupted. In this work, we match the theoretical performance of their flooding protocol with a practical protocol that only relies a γ fraction of the weight remaining honest, which is more relevant in the blockchain setting.

Kadcast [37] is a recent flooding protocol specifically designed for blockchains. Interestingly, they claim that structured networks are inherently more efficient than unstructured networks and propose a structured protocol with $O(\log n)$ neighbors and $O(\log n)$ steps to propagate a message, which is similar to what we achieve using an unstructured network. It is unclear how their protocol performs under Byzantine failures. Further, we note that structured networks are inherently vulnerable to attacks by adaptive adversaries.

A different line of work [27, 28, 31] considers how to propagate updates in a database using gossip where at most t of the processors may be corrupted. The setting is however different from ours as they assume that at least t honest parties get the update as input initially, and only updates input to some honest processor can be accepted by the other processors.

Probabilistic communication have also been used to improve the communication complexity for both multi-party-computation (MPC) [9] and Byzantine broadcast [39]. In [9], communication between honest parties is assumed to be hidden from the adversary. This is exploited by constructing a random communication network with an average polylogarithmic degree based on Erdős–Rényi graphs. They thereby achieve a MPC protocol with low communication locality that is secure against a fully adaptive adversary. [39] combines the classic broadcast

protocol by Dolev and Strong [17] with gossiping based upon Erdős–Rényi-graphs to obtain the first broadcast algorithm with a sub-cubic communication complexity for a dishonest majority. Using similar techniques and assuming a trusted setup they also achieve an asymptotically optimal communication complexity for parallel broadcast.

A different line of work considers the problems of MPC and Agreement on incomplete communication networks [10, 11, 18, 21, 23, 25, 40]. To circumvent complexity bounds for fully adaptive adversaries, the seminal work of [18] introduced the problem of *almost-everywhere agreement* as a relaxation of agreement where not all nodes are required to be consistent, but a small number of nodes are allowed to be inconsistent. Since then, the relaxation has also been extended to MPC [21], and different aspects of solutions to this problem have been continuously improved [10, 11, 23, 25, 40]. Notably, [25] used probabilistic communication to increase the number of consistent parties, and [11] used Erdős–Rényi graphs with a diameter of 2 to obtain a construction secure not only against adaptive corruptions but also an adversary allowed to adaptively remove some communication links. In our work nodes are also of bounded degree, but contrary to this line of work we work in a slightly weaker adversarial model which allows us to ensure correctness for *all* parties.

Attacks on the network layers of blockchains. Attacks on network layers of blockchains are not only a theoretical concern. In fact, several works [5, 22, 29, 38] have shown that it has been possible to launch eclipsing attacks[6] against nodes in the Bitcoin network and the Ethereum network.

Bitcoin's peer-to-peer network works by letting each node in the network maintain 8 outgoing connections and up to 117 incoming connections. This is clearly insecure when considering a resource-constrained adversary instead of a traditional adversary (as the probability of only connecting to adversarial nodes can be arbitrarily high). Additional to this inherent insecurity, [22] showed how to eclipse a node that is already a part of an existing honest network by exploiting a bias in the way a peer selects its outgoing connections. They launched such an attack with only 4600 bots and achieved 85% success probability to actually eclipse a targeted node.

By default, a node in the Ethereum peer-to-peer network selects 13 outgoing connections contrary to the 8 that is the default in Bitcoin. Hence, one might be led to believe that it is more difficult to eclipse an Ethereum node than a Bitcoin node. However, in a Ethereum neighbors are selected using a distance measure that is based on nodes' public keys. Exploiting that in a prior version of the Ethereum client a single computer was allowed to run several nodes, [29] showed that just a single computer can be used to mount an attack by creating multiple carefully selected public keys.

[6] An attack where an adversary tricks an honest party into talking only with adversarial parties. It is thereby possible for the adversary to manipulate the honest node in various ways.

[5] showed that BGP-Hijacking can also be used to eclipse Bitcoin nodes. However, we note that such attack is immediately observable as an adversary will need to announce a false BGP prefix publicly. In [38], it was shown that a stealthier version of such an attack in can also be launched against a Bitcoin node by additionally influencing how a bitcoin node selects its outgoing connections. We note that such attacks are attacks on the infrastructure of the internet, and therefore fall outside the scope of our model.

We note that the attacks presented in [22, 29, 38] all rely on exploiting the heuristics used to select outgoing connections for nodes in the peer-to-peer network. Hence, such attacks would not have been possible if, instead of heuristics a provably secure protocol (such as the one presented in this work) had been deployed.

Detecting eclipse attacks. As a way of mitigating attacks on the network layer a line of work considers the possibility of detecting eclipse attacks [4,41,44]. [41] provide a method for using supervised learning to detect eclipsing attacks based on the metadata in packages. We note that this method is only as good as its data set for training, and hence cannot be used to detect attacks in general. A different approach is to try to detect eclipse attacks based on the absence of new blocks [4,44]. However, this method has the drawback that it becomes arbitrarily slow as the fraction of resources controlled by an adversary approaches 50%, and even for small values, it takes upwards of 3 hours to detect. Finally, it has been considered to detect eclipse attacks using an additional overlay gossip protocol [4]. However, contrary to this work this is not *proven* to work but rather demonstrated to work empirically.

Consequences of eclipse attacks. If a party is eclipsed it is immediate that security proofs that rely on guaranteed message delivery no longer apply. Several works have shown that eclipse attacks do not only invalidate the security proofs but actually invalidate the actual security of blockchain protocols [22,33,43]. Eclipsing can be used to invalidate the total order that blockchain provides and thereby allow double-spend attacks [22], amplify the rewards from selfish mining [33], and dramatically speed up "stake-bleeding"-attacks [43].

The Generals' Scuttlebutt: Byzantine-Resilient Gossip Protocols [13]. Concurrent with and independent of our work, [13] considered the problem of designing a message diffusion mechanism based on the majority of honest stake assumption. The main focus of that paper is to design a network protocol specifically for the Ouroboros Praos consensus protocol [15]. To mitigate a specific denial-of-service attack possible in that protocol (and related proof-of-stake protocols), the authors propose a mechanism that relies on long-lived connections between parties to synchronize chains instead of generically diffusing messages. A consequence of these long-lived connections between parties is that an adaptive adversary can eclipse a set of honest parties. Because their ideal functionality allows such eclipsing, the functionality is different from the assumed functionality of [15] (and thereby the functionality implemented in this work), and the authors argue

in [13] that security of [15] can be proven using this new functionality. In contrast to that, the focus of our work is to realize the flooding functionality without eclipsing assumed by most existing blockchain protocols. Hence, while some techniques are similar, the results of [13] are mostly orthogonal to our work.

2 Notation and Model

2.1 Notation

We will use κ to denote the security parameter of our protocols. We will write $A \xleftarrow{\$} \mathcal{D}$ to sample the value A from the distribution $\mathcal{D}$ and use the infix notation $\sim$ to denote that two random variables are distributed identically. We will let $\mathcal{B}(n, \rho)$ denote the binomial distribution with parameters n and ρ, and $\mathcal{U}(A)$ denote the uniform distribution on a set A. We denote by $\log x$ the natural logarithm of x. In our proofs we will write RHS and LHS to refer to respectively the right hand side and left hand side of (in)equalities.

Graphs. We use standard notation for graphs and let $G = (\mathcal{V}, E)$ be a graph with nodes $\mathcal{V}$ and edges E. An edge can be either directed in which case we will write (v, z) to denote the edge from v to z, or undirected in which case we will write $\{v, z\}$ to denote the edge between the two nodes. We write $\texttt{dist}(v, z)$ to denote the shortest distance between two nodes v and z. Further, we use the shorthand notation $\texttt{MaxDist}(G, v) \triangleq \max_{z \in \mathcal{V}} \texttt{dist}(v, z)$ for the maximum distance from v to any node in a graph $G = (\mathcal{V}, E)$, and the following notation $\texttt{Diam}(G) \triangleq \max_{v \in \mathcal{V}} \texttt{MaxDist}(G, v)$ for the diameter of a graph G.

We also define Erdős–Rényi graphs and digraphs.

Definition 1 (Erdős–Rényi (di)graphs). *An Erdős–Rényi (di)graph is an (di)graph $G = (\mathcal{V}, E)$ where all possible edges are present with an independent probability ρ. That is for any $v, z \in \mathcal{V}$, we have $\Pr[\{v, z\} \in E] = \rho$ for Erdős–Rényi graphs and $\Pr[(v, z) \in E] = \rho$ for digraphs. To sample such a graph G with $|\mathcal{V}| = \eta$, we write $G \xleftarrow{\$} \mathcal{G}_{\mathrm{ER}}(\eta, \rho)$ and for the directed case $G \xleftarrow{\$} \mathcal{G}_{\overrightarrow{\mathrm{ER}}}(\eta, \rho)$.*

2.2 Parties, Weight, Adversary and Communication Network

We let $\mathcal{P}$ denote the static set of parties for which our protocols will work. For convenience we let $n \coloneqq |\mathcal{P}|$ and let $\mathcal{H} \subseteq \mathcal{P}$ be the set of parties that are honest.

We assume that a public weight is assigned to each party. We let W_p denote the weight assigned to party p, and let $\alpha_p \coloneqq \frac{W_p}{\sum_{p \in \mathcal{P}} W_p}$ i.e., the fraction of the total weight assigned to party p.

We allow an adversary to corrupt any subset of the parties such that the remaining set of honest parties together constitutes more than a $\gamma \in (0, 1]$ fraction of the total weight. Formally, we assume that $\sum_{p \in \mathcal{H}} \alpha_p \geq \gamma$, and that all parties

have a non-zero positive weight i.e. $\forall p \in \mathcal{P}, W_p > 0$.[7] We will refer to this assumption as the honest weight assumption. For simplicity, we consider a static adversary, although our results also hold against a so-called delayed-adaptive adversary [30], where the corruptions can be adaptively chosen but only happen after a certain amount of time.

Parties $\mathcal{P}$ have access to a complete network of point-to-point authenticated channels that guarantee delivery within a bounded delay. Concretely, we assume that all channels ensure delivery within δ_{Channel} time.

3 Weighted Flooding

In this section we present a practical and provably secure flooding protocol WFF (weighted fan-out flooding) that only relies on the honest weight assumption. Before doing so we first present our definition of a flooding protocol in Section 3.1. Then, in Section 3.2 we present a generic skeleton for flooding protocols that is parameterized by the way parties selects their neighbors, instantiate this skeleton in order to obtain our practical protocol (WFF), and prove that it is sufficient to consider the way neighbors are selected in order to derive security of a protocol. We use this skeleton to define our theoretical flooding protocol that is secure based upon each party emulating a number of nodes proportional to their weight in an Erdős–Rényi graph (Section 3.3). Finally, in Section 3.4 we use two intermediary protocols in order to derive the security of WFF from our theoretical protocol. All proofs can be found in the full version [26].

3.1 Properties of Flooding Protocols

Below we give our property based definition of a flooding protocol.[8]

Definition 2. *Let Π be a protocol executed by parties $\mathcal{P}$, where each party $p \in \mathcal{P}$ can input a message at any time, and as a consequence all parties get a message as output. We say that Π is a Δ-flooding protocol if the two properties hold with a probability overwhelming in the security parameter κ for each message m:*

1) If m is input by an honest *party for the first time at time τ, then by time $\tau + \Delta$ it is ensured that all other honest parties output m.*
2) If m is output by an honest *party at time τ, then by time $\tau + \Delta$ it is ensured that all honest parties output m.*

[7] For a discussion of the necessity of the zero-weight requirement see Section 4 and for methods to anyway achieve delivery to such zero-weight parties we refer to the full version of this work [26]. .

[8] Note that for protocols with no secrecy (each event is leaked to the adversary), and for functionalities that give the adversary full control while respecting these properties a simulation-based security notion is directly implied by the property-based definition. For flooding networks, this technique is used in the proofs in [30].

Note that this definition subsumes the assumptions that many blockchain protocols rely on [12, 15, 16, 20, 34, 36]. To the best of our knowledge only [16] relies on both Properties 1) and 2), whereas the other works only rely solely on Property 1). However, as Property 2) essentially comes for free for the type of protocols we consider (each party will forward everything they receive and thereby act as if they themselves send the message) we have chosen to include it in our definition. Furthermore, because of this structure of our protocols, it is sufficient to bound the probability of Property 1) in order to show that our protocols are in fact flooding protocols according to the definition. For our proofs and lemma statements, it is, therefore, useful to define notation for the predicate that a message input to an honest party for the first time is delivered respecting the delivery bound for a flooding protocol, which is what we encapsulate in the predicate below.

Definition 3 (Timely delivery). *For a message m that is input for the first time at an honest party at time τ we say that m is Δ-*timely-delivered *if all honest parties have output m no later than time $\tau + \Delta$. We let $\texttt{Timely}_m(\Delta)$ denote the induced predicate.*

Similarly, for a message m that is input for the first time at an honest party, we define the *message complexity* as the number of messages sent by honest parties until all honest parties output m. Looking ahead, since our protocols only consist of forwarding the initial message m, the total message complexity is simply $|m|$ times the message complexity.

Mitigating denial-of-service attacks. It is immediate that any protocol that lives up to the definition of a flooding protocol, as given above, is open to denial-of-service attacks. An adversary can simply flood arbitrary messages until the bandwidth is exceeded. This is possible because the definition requires *all* messages to be forwarded. To prevent such attacks, it is natural to consider a notion of validity and only require the delivery guarantees to apply for "valid" messages. Concretely, one could let each party $p \in \mathcal{P}$ have an updatable local predicate $\texttt{Valid}_p$ and only require that messages that are considered valid by all parties for Δ after being input/output for the first time should be propagated.

For clarity of presentation, we have left this out of our definition and protocols. However, we note that it is easy to accommodate our protocols to such notion by letting each party check if a message is valid before propagating it. We note that with such modification, all our proofs and lemmas still hold for messages that are considered valid by all parties for at least Δ after they are input/output.

3.2 A Skeleton For Flooding Protocols

We now present a skeleton for our flooding algorithm. The structure of the protocol is very similar to the protocols proposed in [30], but contrary to their protocols our protocol takes an additional parameter $\mathcal{N}$, which is an algorithm that allows each party to sample a set of neighbors. We refer to this parameter as the *neighborhood selection algorithm*.

The protocol accepts two commands: One for sending and one for checking which messages have been received. Once a send command is issued to a party, the party will forward the message to a set of neighbors that are determined using the neighborhood selection algorithm. Furthermore, once a message is received on a point-to-point channel the receiver checks if the message has already been relayed and if not it forwards the message to a set of neighbors that is again selected using the neighborhood selection algorithm.

Protocol $\pi_{\text{Flood}}(\mathcal{N})$

We use $\mathcal{N}_p$ to denote the neighborhood distribution of party p. Each party $p_i \in \mathcal{P}$ keeps track of a set of relayed messages $\texttt{Relayed}_i$ which will also be used to keep track of which messages party p_i has received.

Initialize: Initially, each party p_i sets $\texttt{Relayed}_i \coloneqq \varnothing$.

Send: When p_i receives (Send, m), they sample a set of neighbors $N \xleftarrow{\$} \mathcal{N}_p$ and forwards the message to all parties in N. Finally, they set $\texttt{Relayed}_i \coloneqq \texttt{Relayed}_i \cup \{m\}$.

Get Messages: When p_i receives $(\mathit{GetMessages})$ they return $\texttt{Relayed}_i$.

When party p_i receives message m on a point-to-point channel where $m \notin \texttt{Relayed}_i$, p_i continues as if they had received (Send, m). Otherwise, m is ignored.

Looking ahead and as an example of a neighborhood selection algorithm we present our practical and provably secure neighborhood selection algorithm.

A Practical Neighborhood Selection Algorithm Our algorithm $\mathsf{WFS}(\texttt{E}, k)$ (abbreviation for "Weighted Fan-out Selection") takes two parameters: a function $\texttt{E} : \mathcal{P} \to \mathbb{N}$ that allows to take stake into account when deciding how many neighbors each party should select and a parameter k that scales this number.

The idea of the algorithm is that each party p chooses $K \coloneqq k \cdot \texttt{E}(p)$ number of neighbors (excluding themselves). The neighbors are chosen according to weighted sampling without replacement [7] where each party again is being weighted with $\texttt{E}$. More precisely, party p chooses K neighbors from $\mathcal{P} \setminus \{p\}$, and the probability to choose the tuple of neighbors $(q_1, \ldots, q_K)$ is defined as:

$$\Pr\left[(q_1, \ldots, q_K)\right] = \prod_{i=1}^{K} \frac{\texttt{E}(q_i)}{\sum_{q \in \mathcal{P} \setminus \{p, q_1, \ldots, q_{i-1}\}} \texttt{E}(q)}.$$

The probability to choose a certain neighborhood set $\{q_1, \ldots, q_K\}$ is then the sum over the probabilities over all the permuted tuples. We denote by $\mathcal{W}(K, \texttt{E}, p)$ the resulting distribution.

Algorithm $\mathsf{WFS}_p(\mathtt{E}, k)$

1: Let $N \coloneqq \varnothing$.
2: Set $K \coloneqq k \cdot \mathtt{E}(p)$.
3: Sample $N \xleftarrow{\$} \mathcal{W}(K, \mathtt{E}, p)$.
4: **return** N.

Our final protocol is the protocol obtained by instantiating the flooding skeleton π_{Flood} with the neighborhood selection algorithm WFS that again is to be instantiated with the function $\mathtt{E}(p) \coloneqq \lceil \alpha_p \cdot n \rceil$. We name this protocol the *weighted fan-out flooding* protocol and use the abbreviation $\mathsf{WFF}(k) \coloneqq \pi_{\text{Flood}}(\mathsf{WFS}(\mathtt{E}, k))$ for $\mathtt{E}(p) \coloneqq \lceil \alpha_p \cdot n \rceil$. In Sections 3.3 and 3.4 it will become apparent why this exact choice of function is advantageous and ensures a secure protocol, but for now we simply state our final theorem which states that WFF is in fact a flooding protocol with a logarithmic round complexity and a low message complexity.

Theorem 2. *Let* $\Delta \coloneqq \left(7 \cdot \log\left(\frac{6n}{\log(n)+\kappa}\right) + 2\right) \cdot \delta_{Channel}$. *Then* $\mathit{WFF}\left(\frac{\log(n)+\kappa}{\gamma}\right)$ *is a* Δ*-flooding protocol with message complexity less than* $2n \cdot \frac{\log(n)+\kappa}{\gamma}$.

The Honest Sending Process To prove security of WFF we will relate the security of WFF to a series of other protocol which will all take the structure of π_{Flood} but use different neighborhood selection algorithms. Hence, we would like to be able to relate the security of the overall flooding protocol to just the neighborhood selection algorithm used. To do so we first define a random process for creating a graph where each honest party is a node, given a family of neighborhood selection algorithms $\mathcal{N}$, a starting party p, and a distance λ. The intuition is that this process mimics the worst-case behavior of the adversary during a sending process starting from party p. However, separating this into a process without adversarial influence allows us to relate probabilistic experiments without taking into account the choices of an adversary which could have a strategy that depends on parts of the outcome of the experiments.

Definition 4. *Let* $\mathcal{N}$ *be a family of neighborhood selection algorithms, let* $p \in \mathcal{H}$, *and let* $\lambda \in \mathbb{N}$ *be a distance. We let the* honest sending process, $\mathit{HSP}(p, \mathcal{N}, \lambda)$, *be a random process that returns a directed graph* $G = (\mathcal{V}, E)$ *defined by the following random procedure:*

1. *Initially,* $E \coloneqq \varnothing$. *Furthermore, we keep track of set* $\mathtt{Flipped} \coloneqq \varnothing$ *that consists of nodes that have already had their outgoing edges decided, and a first-in-first-out queue* $\mathtt{ToBeFlipped} \coloneqq \{(p, 0)\}$ *of nodes and their distance from* p *that are to have their edges decided.*
2. *The process proceeds with the following until* $\mathtt{ToBeFlipped} == \varnothing$.
   (a) *Take out the first element of* $\mathtt{ToBeFlipped}$ *and let it be denoted by* (p', i).
   (b) *Let* $N \xleftarrow{\$} \mathcal{N}_{p'}$ *and set* $N \coloneqq N \cap \mathcal{H}$.

(c) Update the set of edges $E \coloneqq E \cup \{(p', p'') \mid p'' \in N\}$ *and let* $\texttt{Flipped} \coloneqq \texttt{Flipped} \cup \{p'\}$.
(d) If $i + 1 < \lambda$, *for all* $p'' \in N \setminus \texttt{Flipped}$ *add* $(p'', i + 1)$ *to* `ToBeFlipped`.
3. *Finally, return* $G = (\mathcal{H}, E)$.

Next, we are interested in bounding the probability that a message is delivered within the time guaranteed by the flooding algorithm in terms of the probability that there is a low distance to all parties from the sender. We show that the probability that π_{Flood} ensures timely delivery for a message is lower-bounded by the probability that the honest sending process results in a graph where the sender can reach all other honest nodes within a certain number of steps.

Lemma 1. *Let* $\mathcal{N}$ *be a family of neighborhood selection algorithms, let* $p \in \mathcal{H}$, *and let* $\lambda \in \mathbb{N}$ *be a distance. Further, let* m *be a message that is input to* p *for the first time during the execution of* $\pi_{Flood}(\mathcal{N})$ *and let* $G \xleftarrow{\$} \mathsf{HSP}(p, \mathcal{N}, \lambda)$. *Then,*

$$\Pr[\texttt{MaxDist}(G, p) \leq \lambda] \leq \Pr[\texttt{Timely}_m(\lambda \cdot \delta_{Channel})]. \tag{1}$$

Proof Idea. We observe the random experiment arising from delivering the message m in the protocol and construct a new graph where each honest party corresponds to a node and we include a directed edge from one party to another if a message is sent and delivered before time $\lambda \cdot \delta_{\text{Channel}}$. In this graph, we observe that if the distance is at most λ from the sender to any party then the message was delivered timely. We then define how to use this experiment to define the HSP experiment by copying a subset of the edges from this new graph to the honest sending graph and thereby ensuring that any path in the graph from the honest sending process will also be in this new graph. □

Lemma 1 ensures that it is sufficient to consider neighborhood selection algorithms and prove that graphs constructed via the honest sending process has a low distance from the sender to all other parties.

3.3 A Theoretical Protocol: Emulating Nodes in Erdős–Rényi Graphs

Our central idea for achieving a flooding network that relies on the honest weight assumption is to let each party emulate a number of nodes proportional to their weight in a hypothetical Erdős–Rényi graph. We will refer to this hypothetical graph as the *emulated graph*. Now, our idea is that if there is an edge between an emulated node v and another emulated node z corresponds to that the party emulating node v should forward the message to the party emulating z. Our goal is now to ensure each honest party emulates at least one node and that the emulated graph has a low diameter, as this will result in that all parties will receive the message quickly.

Concretely, we introduce a function $\texttt{E} : \mathcal{P} \to \mathbb{N} \setminus \{0\}$ which for each party determines how many nodes this party should act as in the emulated graph. We refer to this function as the *emulation function*.[9] For such emulation function

[9] For a function to be an emulation function, we require that all parties should emulate at least 1 node, which is why the codomain of the function is defined to be $\mathbb{N} \setminus \{0\}$.

we define notation for the number of emulated nodes $n_{\mathtt{E}} \triangleq \sum_{p \in \mathcal{P}} \mathtt{E}(p)$ and the number of honest nodes that are emulated $h_{\mathtt{E}} \triangleq \sum_{p \in \mathcal{H}} \mathtt{E}(p)$.

Before looking at how to choose an emulation function, let us present how the idea leads to a very simple algorithm for selecting neighbors by letting the emulated graph take the form of an Erdős–Rényi graph. We let ρ denote the probability that there should be an edge between any two nodes in the emulated graph. The probability that party p_i should forward a message to party p_j is:

$$\begin{aligned}
&\Pr[p_i \text{ should forward a message to } p_j] \\
&:= \Pr[\text{exists edge from any of } p_i\text{'s emulated nodes to any of } p_j\text{'s}] \\
&= 1 - \Pr[\text{there are no edges between any of } p_i \text{ and } p_j\text{'s emulated nodes}] \\
&= 1 - (1 - \Pr[\text{there is an edge between any two emulated nodes})^{\mathtt{E}(p_i)\cdot\mathtt{E}(p_j)} \\
&= 1 - (1-\rho)^{\mathtt{E}(p_i)\cdot\mathtt{E}(p_j)}.
\end{aligned} \tag{2}$$

This gives rise to the following family of neighbor selection algorithms indexed by a party $p \in \mathcal{P}$ and parameterized by an emulation function $\mathtt{E}$ and an edge probability ρ.

Algorithm $\mathsf{ER\text{-}Emulation}_p(\mathtt{E}, \rho)$

Let $N := \varnothing$.
Let $P := \mathcal{P} \setminus p$.
while $P \neq \varnothing$ **do**
    Pick $r \in P$.
    Sample $c \xleftarrow{\$} \mathcal{U}([0,1])$.
    if $c \leq 1 - (1-\rho)^{\mathtt{E}(p)\cdot\mathtt{E}(r)}$ **then**
        Update $N := N \cup \{r\}$.
    Update $P := P \setminus \{r\}$.
return N.

Relating Erdős–Rényi graphs and the honest sending process. We now formalize the intuition that given that an emulated graph is "well connected" then the graph from the honest sending process is also "well connected". In particular, we relate the probability that the distance in a directed Erdős–Rényi graph is large to the probability that the distance from the sender is large in the honest sending process.

Lemma 2. *Let $\rho \in [0,1]$, let $\lambda \in \mathbb{N}$, let $p \in \mathcal{H}$, and let $\mathtt{E} : \mathcal{P} \to \mathbb{N} \setminus \{0\}$ be an emulation function. Further, let $G_1 \xleftarrow{\$} \mathsf{HSP}(p, \mathsf{ER\text{-}Emulation}(\mathtt{E}, \rho), \lambda)$ and let $G_2 \xleftarrow{\$} \mathcal{G}^{\rightarrow}_{\mathrm{ER}}(h_{\mathtt{E}}, \rho)$. Then for any node $v \in \mathcal{V}$ we have,*

$$\Pr[\mathtt{MaxDist}(G_2, v) \leq \lambda] \leq \Pr[\mathtt{MaxDist}(G_1, p) \leq \lambda]. \tag{3}$$

Proof Idea. We use Eq. (2) and a mapping between the nodes of G_2 and the honest parties to define both graph distributions in terms of the same random experiment. We then observe that the edges that are relevant for the distance from v in G_2 are also included in G_1. □

Next, we show that the probability that a particular node can reach all other nodes within a certain distance in a directed Erdős–Rényi graph is lower-bounded by the probability that an undirected Erdős–Rényi graph has a high diameter.

Lemma 3. *Let $\rho \in [0,1]$, let $\lambda \in \mathbb{N}$ and let $\eta \in \mathbb{N}$. Further, let $G_1 = (\mathcal{V}_1, E_1) \xleftarrow{\$} \mathcal{G}^{\rightarrow}_{\mathrm{ER}}(\eta, \rho)$ and let $G_2 = (\mathcal{V}_2, E_2) \xleftarrow{\$} \mathcal{G}_{\mathrm{ER}}(\eta, \rho)$. Then for any node $v \in \mathcal{V}_1$ we have,*

$$\Pr[\mathtt{Diam}(G_2) \leq \lambda] \leq \Pr[\mathtt{MaxDist}(G_1, v) \leq \lambda]. \tag{4}$$

Proof Idea. We define a coupling between the two graphs such that the edges that are relevant for the distance from v in G_1 are ensured to have undirected counterparts included in G_2. Hence, any path starting from v in G_2 translates to a similar path in G_1. □

Choosing a good emulation function. Let us now turn our attention to how to select a good emulation function. Before looking at a concrete function, let us consider what properties constitute a good emulation function. The only property of the emulation function that we have used so far is that all parties should emulate at least 1 node.[10] However, there are additional things that we want from a useful emulation function:

1. It should ensure a low distance from any sender in the graph resulting from the honest sending process.
2. The message complexity of the protocol should be as small as possible.

Lemmas 2 and 3 bounds the probability that the honest sending process results in a graph with some nodes not reachable within the sender in terms of the probability that an Erdős–Rényi graph (of size identical to the number of honest emulated nodes) has a large diameter. Furthermore, looking ahead we will instantiate $\rho \approx \frac{\log(h_{\mathtt{E}})+\kappa}{h_{\mathtt{E}}}$ to obtain an Erdős–Rényi graph that has a diameter logarithmic in $h_{\mathtt{E}}$ unless with a probability that is negligible in κ. Unfortunately, $h_{\mathtt{E}}$ will not be known at the time of instantiation, so we will have to instantiate ρ with a lower bound on $h_{\mathtt{E}}$ in the denominator and similarly an upper bound in the denominator. For this discussion, let us use $n_{\mathtt{E}}$ as an upper bound.

The expected number of neighbors for a party is linear in ρ. To see this let $N \xleftarrow{\$} \mathsf{ER\text{-}Emulation}_p(\mathtt{E}, \rho)$ and let us estimate the expected size of N using Bernoulli's inequality:

$$\mathrm{E}[|N|] = \sum_{r \in \mathcal{P} \setminus \{p\}} 1 - (1-\rho)^{\mathtt{E}(p)\cdot\mathtt{E}(r)} \leq \sum_{r \in \mathcal{P} \setminus \{p\}} \rho \cdot \mathtt{E}(p) \cdot \mathtt{E}(r) \leq \rho \cdot \mathtt{E}(p) \cdot n_{\mathtt{E}}. \tag{5}$$

[10] This property was used in the proof of Lemma 2.

Hence, for ρ chosen according to the above, a bound on the expected message complexity will be

$$O\left((\log(n_{\mathtt{E}}) + \kappa) \cdot \frac{n_{\mathtt{E}}^2}{h_{\mathtt{E}}}\right). \tag{6}$$

Our approach for finding a good emulation function has thus been to search for an emulation function which makes this value as small as possible. As result of this approach we choose the emulation function to be $\mathtt{E}(p) \coloneqq \lceil \alpha_p \cdot n \rceil$. For this emulation function above we can derive the following bounds using only the honest weight assumption:

$$h_{\mathtt{E}} = \sum_{p \in \mathcal{H}} \mathtt{E}(p) \geq \sum_{p \in \mathcal{H}} \alpha_p \cdot n \geq \gamma \cdot n, \tag{7}$$

and

$$n_{\mathtt{E}} = \sum_{p \in \mathcal{P}} \mathtt{E}(p) \leq \sum_{p \in \mathcal{P}} (\alpha_p \cdot n + 1) = 2 \cdot n. \tag{8}$$

By plugging the bounds from Eqs. (7) and (8) into Eq. (6) we acquire an expected message complexity that is upper bounded by $O\left((\log(n)+\kappa)\cdot\frac{n}{\gamma}\right)$ when parameters are instantiated to obtain a logarithmic diameter of the graph. If we instead of assuming a constant fraction of honest weight assumed a constant fraction of honest parties, we could let $\mathtt{E}(p) \coloneqq 1$, which would result in $n_{\mathtt{E}} \coloneqq 1$ and thereby a protocol identical to the one proposed in [30]. By using the same analysis as above we would then be able to bound the expected message complexity by $O\left((\log(n) + \kappa) \cdot \frac{n}{\gamma}\right)$. Interestingly, the bound on the message complexity for the weighted section would only be a factor of ≈ 4 larger than the corresponding bound for the non-weighted setting.

Proving security of our theoretical flooding protocol. We now state and prove that the probability that $\pi_{\mathrm{Flood}}(\mathsf{ER\text{-}Emulation}(\mathtt{E}, \rho))$ protocol does not ensure timely delivery is negligible for certain choices of $\mathtt{E}$ and ρ. To prove this, we make use of probabilistic bounds on the diameter of undirected Erdős–Rényi graphs from the full version of [30]. As a first step, we bound the probability that the distance of the honest sending process using the neighborhood selection algorithm $\mathsf{ER\text{-}Emulation}(\mathtt{E}, \rho)$ has a large distance from the sender.

Lemma 4. *Let* $\mathtt{E}(p) \coloneqq \lceil \alpha_p \cdot n \rceil$*, let* $d \in [7, \infty]$*, and let* $\rho \coloneqq \frac{d}{\gamma \cdot n}$*. Further, let* $p \in \mathcal{H}$ *and* $G \xleftarrow{\$} \mathit{HSP}(p, \mathsf{ER\text{-}Emulation}(\mathtt{E}, \rho), (\lceil 7 \cdot \log\left(\frac{n}{d}\right) + 2 \rceil))$*. Then*

$$\begin{aligned} &\Pr\left[\mathtt{MaxDist}(G, p) \leq \left(7 \cdot \log\left(\frac{n}{d}\right) + 2\right)\right] \\ &\geq 1 - \left(2 \cdot n \cdot \left(e^{-\frac{d}{18}} + \left(6 \cdot \log\left(\frac{n}{d}\right) + 1\right) \cdot e^{-\frac{7 \cdot d}{108}}\right) + e^{-\gamma \cdot n \cdot \left(\frac{d}{9} - 2\right)}\right). \end{aligned} \tag{9}$$

Proof Idea. We use Eqs. (7) and (8) to bound the size of the emulated graph in the honest sending process and apply Lemmas 2 and 3 to reduce the probability to the probability that an Erdős–Rényi graph has a low diameter. The bound then follows by instantiating Lemma 3 in the full version of [30]. □

A direct corollary of Lemmas 1 and 4 is that the probability that the protocol $\pi_{\text{Flood}}(\mathsf{ER\text{-}Emulation}(\mathtt{E}, \rho))$ ensures timely delivery is lower bounded by Eq. (9) when choosing $\mathtt{E}$ and ρ as discussed above.

3.4 Security of WFF

In the previous section we proved that ER-Emulation induces a secure protocol. Unfortunately, it is not a practical neighborhood selection algorithm, as it requires each party to do n coin-flips per message that is sent and forwarded. In this section, we introduce two intermediate algorithms in order to prove WFF secure (Fast-ER-Emulation and Practical-ER-Emulation) by doing gradual changes to ER-Emulation, until we finally arrive at the algorithm WFS which is both practical, simple, and similar to algorithms deployed in practice (except that this algorithm maintains its complexity even for weighted corruptions).

Intermediary Neighborhood Selection Algorithms We first introduce the algorithm Fast-ER-Emulation, which is distributed identically to ER-Emulation, but is more practical. The algorithm exploits that another way of creating an Erdős–Rényi graph is to first decide how many edges each node should have using the binomial distribution and then select these neighbors at random.

Below we will abuse notation slightly and write $\mathtt{E}(P)$ to denote the set of emulated nodes for a set of parties $P \subseteq \mathcal{P}$ and an emulation function $\mathtt{E}$,

$$\mathtt{E}(P) \triangleq \{p_i \mid p \in P \wedge i \in \{1, 2, \ldots, \mathtt{E}(p)\}\}.$$[11]

Algorithm Fast-ER-Emulation$_p(\mathtt{E}, \rho)$

1: Let $N \coloneqq \varnothing$.
2: **for** $i \coloneqq 0;\ i < \mathtt{E}(p);\ i\,{+}{+}$ **do**
3: Sample $k \xleftarrow{\$} \mathcal{B}(|\mathtt{E}(\mathcal{P} \setminus \{p\})|, \rho)$.
4: Let A be k nodes sampled from $\mathtt{E}(\mathcal{P} \setminus \{p\})$ without replacement.
5: Set $N \coloneqq N \cup \{p' \mid p'_j \in A \wedge j \in \mathbb{N}\}$.
6: **return** N.

We now show Fast-ER-Emulation and ER-Emulation are identically distributed.

Lemma 5. *Let $\rho \in [0, 1]$, let $\lambda \in \mathbb{N}$, let $p \in \mathcal{H}$, and let $\mathtt{E} : \mathcal{P} \to \mathbb{N} \setminus \{0\}$ be an emulation function. If $G_1 \xleftarrow{\$} \mathsf{HSP}(p, \mathsf{ER\text{-}Emulation}(\mathtt{E}, \rho), \lambda)$ and $G_2 \xleftarrow{\$} \mathsf{HSP}(p, \mathsf{Fast\text{-}ER\text{-}Emulation}(\mathtt{E}, \rho), \lambda)$ then $G_1 \sim G_2$.*

[11] This set may be different from the actual set of nodes that will be emulated in an execution of the protocol as dishonest parties might choose to deviate from the protocol. However, it is still useful to define the set in order to define honest behavior.

Proof Idea. We show that the graphs are distributed identically by showing that their respective neighborhood selection algorithms are distributed identically. This is shown by showing that for both distributions each edge between emulated nodes appears with independent probability ρ. □

A problem of Fast-ER-Emulation is that each party p needs to make $\mathtt{E}(p)$ number of draws from the binomial distribution. One way to avoid this is to make a single random draw for the number of nodes all emulated nodes should send to and then afterward choose this number of nodes uniformly without replacement. Below we present the algorithm Practical-ER-Emulation, which does exactly that.

Algorithm Practical-ER-Emulation$_p(\mathtt{E}, \rho)$

1: Let $N \coloneqq \varnothing$.
2: Sample $k \xleftarrow{\$} \mathcal{B}\left(\mathtt{E}(p) \cdot |\mathtt{E}(\mathcal{P} \setminus \{p\})|, \rho\right)$.
3: Let A be k nodes sampled from $\mathtt{E}(\mathcal{P} \setminus \{p\})$ without replacement.
4: Set $N \coloneqq \{p \mid p_i \in A \wedge i \in \mathbb{N}\}$.
5: **return** N.

Practical-ER-Emulation is not distributed identically to Fast-ER-Emulation, as there is a smaller expected overlap between the selected emulated nodes. However, it still holds that the graph resulting from the honest sending process based upon Practical-ER-Emulation has a higher chance of having a low distance from the sender than the graph resulting from the honest sending process based upon Fast-ER-Emulation. We make this intuition formal in the lemma below.

Lemma 6. *Let $\rho \in [0,1]$, let $\lambda \in \mathbb{N}$, let $p \in \mathcal{H}$, and let $\mathtt{E} : \mathcal{P} \to \mathbb{N} \setminus \{0\}$ be an emulation function. If $G_1 \xleftarrow{\$}$ HSP$(p,$ Fast-ER-Emulation$(\mathtt{E}, \rho), \lambda)$ and $G_2 \xleftarrow{\$}$ HSP$(p,$ Practical-ER-Emulation$(\mathtt{E}, \rho), \lambda)$ then*

$$\Pr[\mathtt{MaxDist}(G_1, p) \leq \lambda] \leq \Pr[\mathtt{MaxDist}(G_2, p) \leq \lambda]. \tag{10}$$

Proof Idea. We define a coupling between the two graphs by defining a coupling between their respective neighborhood selection algorithms and ensuring that the set of neighbors sampled by Practical-ER-Emulation is a superset of the neighbors of those sampled by Fast-ER-Emulation. We define the coupling using rejection sampling and ensure that any neighbor that is rejected when sampling neighbors for Fast-ER-Emulation will also be rejected when sampling neighbors for Practical-ER-Emulation. □

Note that Lemmas 1 and 4 to 6 together imply that the probability that $\pi_{\mathrm{Flood}}(\text{Fast-ER-Emulation}(\mathtt{E}, \rho))$ and $\pi_{\mathrm{Flood}}(\text{Practical-ER-Emulation}(\mathtt{E}, \rho))$ do not ensure timely delivery is negligible for a certain choice of $\mathtt{E}$ and ρ. Note that Practical-ER-Emulation is very similar to WFS. The main difference is that in Practical-ER-Emulation the number of neighbors is sampled according to the binomial distribution whereas WFS chooses a fixed number of neighbors. We use this observation to relate the probability that the graph constructed by the honest

sending process of Practical-ER-Emulation has a low distance from the sender to the probability that the honest sending process of WFS has a low distance from the sender.

Lemma 7. *Let* $\rho \in [0,1]$*, let* $\epsilon \in [0,1]$ *let* $\lambda \in \mathbb{N}$*, let* $p \in \mathcal{H}$*, let* $k \geq \lceil (1+\epsilon) \cdot n_{\mathtt{E}} \cdot \rho \rceil$*, and let* $\mathtt{E} : \mathcal{P} \to \mathbb{N} \setminus \{0\}$ *be an emulation function. If* $G_1 \xleftarrow{\$} \mathit{HSP}(p, \mathit{Practical\text{-}ER\text{-}Emulation}(\mathtt{E}, \rho), \lambda)$ *and* $G_2 \xleftarrow{\$} \mathit{HSP}(p, \mathit{WFS}(\mathtt{E}, k), \lambda)$ *then*

$$\Pr[\mathtt{MaxDist}(G_1, p) \leq \lambda] - |\mathcal{H}| \cdot e^{-\frac{\epsilon^2 \cdot (n-1) \cdot \rho}{3}} \leq \Pr[\mathtt{MaxDist}(G_2, p) \leq \lambda]. \tag{11}$$

Proof Idea. Similarly to the proof of Lemma 6, we define a coupling between the two graphs by defining a coupling between their neighborhood selection algorithms using rejection sampling. However, in this coupling the invariant that the edges sampled by WFS are a superset of those of Practical-ER-Emulation is only maintained when no party samples more than k neighbors in Practical-ER-Emulation. We bound the probability that this happens using a Chernoff bound. □

We now provide a corollary that bounds the concrete probability that a message that is input via WFF is delivered timely.

Corollary 1. *Let* $k \in \mathbb{N}$ *such that* $k \geq \frac{42}{\gamma}$*. If* m *is a message that is input to some honest party in* $\mathit{WFF}(k)$ *then*

$$\begin{aligned} &\Pr\left[\mathtt{Timely}_m\left(\left(7 \cdot \log\left(\frac{n \cdot 6}{k \cdot \gamma}\right) + 2\right) \cdot \delta_{Channel}\right)\right] \geq 1 - n \cdot e^{-\frac{(n-1)\cdot k}{n \cdot 24}} \\ &- e^{-\gamma \cdot n \cdot \left(\frac{k \cdot \gamma}{54} - 2\right)} - \left(2 \cdot n \cdot \left(e^{-\frac{k \cdot \gamma}{108}} + \left(6 \cdot \log\left(\frac{n \cdot 6}{k \cdot \gamma}\right) + 1\right) \cdot e^{-\frac{7 \cdot k \cdot \gamma}{648}}\right)\right). \end{aligned} \tag{12}$$

Proof Idea. We bound the size of the emulated graph and apply Lemmas 1 and 4 to 7. □

As observed earlier, it is sufficient to bound the probability that a message is timely delivered in order to bound the probability that any of the two properties of a flooding protocol is achieved. Further, note that a party p sends at most $k \cdot \mathtt{E}(p)$ messages when a message is forwarded. Hence, the message complexity is bounded by $\sum_{p \in \mathcal{H}} k \cdot \mathtt{E}(p) \leq 2 \cdot k \cdot n$. Therefore, the security of WFF (and thereby Theorem 2) follows directly from this corollary.

4 Asymptotic Optimality and Practical Considerations

Our results from Section 3.2 show that the protocol $\mathsf{WFF}(k)$ provides provably secure flooding. With respect to efficiency, the results show that there are two possible drawbacks: First, the emulation function $\mathtt{E}(p) = \lceil \alpha_p \cdot n \rceil$ forces parties with very high weight to send to many parties, which lead to bandwidth issues. Secondly, Theorem 2 shows that in our protocol, the number of parties each node has to send to increases logarithmically in the total number of nodes. In this section, we show that both properties are inherent for "flooding protocols".

4.1 Workload of Heavy Parties

It is easy to see that in at least in extreme cases, very heavy parties have to send to a lot of other parties: If there is a single party that has the majority of the total weight, it could be that only this party and an additional one are honest. Since the heavy party is the only one that can be relied upon for message delivery, it needs to send to all other parties. The following lemma generalizes this idea to less extreme settings.

Lemma 8. *For any protocol Π that guarantees delivery to* all honest *parties, and for any subset $S \subseteq \mathcal{P}$ such that $\sum_{p\in S} \alpha_p \geq \gamma$, we have with overwhelming probability that*

$$\sum_{p\in S} \mathtt{degree}_{\Pi}(p) \geq |\mathcal{P} \setminus S|. \tag{13}$$

Proof. Let S be any such set. By the honesty assumption it could be that there is exactly one honest party in $\mathcal{P} \setminus S$. To guarantee delivery to this party, some party in S must send to it. Since it cannot be distinguished which party in $\mathcal{P} \setminus S$ is honest, the parties in S must send to all parties in $\mathcal{P} \setminus S$. □

Another consequence of Lemma 8 is that having a huge number of nodes with very little weight also increases the workload for all other nodes, as shown below.

Corollary 2. *Assume there is a large set $T \subseteq \mathcal{P}$ of parties with combined relative weight $\leq 1-\gamma$ and $|T| \geq n - \epsilon$ for some constant $\epsilon > 0$, and define $S := \mathcal{P} \setminus T$. Then, the average degree of the parties in S must be at least $\frac{n-\epsilon}{\epsilon} \in \Omega(n)$ with overwhelming probability.*

Proof. Since $\sum_{p\in S} \alpha_p = 1 - \sum_{p\in T} \alpha_p \geq \gamma$, Lemma 8 implies that the average degree of the parties in S is at least $\frac{|\mathcal{P}\setminus S|}{|S|}$ with overwhelming probability. By assumption, we have $\frac{|\mathcal{P}\setminus S|}{|S|} = \frac{|T|}{n-|T|} \geq \frac{n-\epsilon}{\epsilon} \in \Omega(n)$. □

Limiting the workload. As we have seen above, having very heavy or many very light parties necessarily yields a large number of outgoing connections for some of the nodes. This is not only undesirable but may also become prohibitive in practice due to limited network bandwidth. If the flooding is deployed, say for a proof-of-stake blockchain, this can be mitigated by putting a lower and an upper limit on the amount of stake for actively participating nodes. This implies that people holding a lot of stake need to split their stake over several nodes (which is anyway beneficial for decentralization if they are run in different locations), and people with too little stake need to, e.g., delegate their stake to another node if supported by the blockchain. The latter can still passively participate by fetching data from other nodes. We discuss how zero-weight parties can fetch in the full version of this work [26].

4.2 Logarithmic Growth of Message Complexity

It is well known that Erdős–Rényi graphs are connected with high probability if and only if edges are included with probability larger than $\frac{\log n}{n}$ [8, Theorem 7.3]. This means the expected degree of a node must be larger than $\log n$ to obtain a connected graph, even without considering corruptions. Since our proofs in Section 3 depart from Erdős–Rényi graphs, one cannot hope to prove a better message complexity with our proof techniques.

On the other hand, our final protocol $\mathsf{WFF}(k)$ does not choose neighbors in the way Erdős–Rényi graphs are constructed, but more closely correspond to so-called directed k-out graphs, which have also been considered in the literature. Those are directed graphs where for each node v independently, k uniformly random other nodes are sampled and directed edges from v to the k sampled nodes are added. It is known that such graphs are connected with probability approaching 1 for $n \to \infty$ already for constant $k = 2$ [19]. Hence, at least without corruptions, $O(n)$ overall message complexity should be enough for our protocol. When considering corruptions, however, a result by Yagan and Makowski [42] implies that $\log n$ connections for each node are necessary, as we show below. This shows that $\mathsf{WFF}(k)$ and Theorem 2 are asymptotically optimal, at least for the special case in which all parties have the same weight.

Lemma 9. *For any flooding protocol in which all honest parties send to k uniformly chosen nodes and delivery to all honest nodes is guaranteed with probability $\geq 1/2$ where up to a $(1-\gamma)$ fraction of nodes can be corrupted, we have for sufficiently large n that*

$$k \geq \frac{\log n}{\gamma + 1/n - \log(1-\gamma-1/n)}.$$

Proof. Yagan and Makowski [42] have considered the setting in which for each of the n nodes p_i, k distinct random other nodes are sampled and undirected edges between p_i and all k sampled nodes are added to a graph. They then consider the subgraph H consisting of the first $\lfloor \gamma' n \rfloor$ nodes for some constant $\gamma' \in (0,1)$ and show in [42, Theorem 3.2] that

$$k < \frac{\log n}{\gamma' - \log(1-\gamma')} \implies \lim_{n\to\infty} \Pr[H \text{ contains isolated node}] = 1.$$

To translate this to our setting, first note that corrupting at most $\lfloor (1-\gamma)n \rfloor$ nodes from the end to leave the first $\lfloor \gamma n + 1 \rfloor$ parties honest is a valid adversarial strategy. To be compatible with the result above, we can set $\gamma' := \gamma + 1/n$. Further note that a node p being isolated in H has the same probability as an honest node not sending to any other honest node and no honest node sending to that one in a flooding protocol. In that case, if p is the sender in the flooding protocol, no honest node will receive the message, and if some other node is the sender, p will not receive the message. Hence, the flooding protocol will fail to deliver the message to all honest nodes in both cases. This implies that, for sufficiently large n, flooding protocols with $k < \frac{\log n}{\gamma+1/n-\log(1-\gamma-1/n)}$ fail to deliver messages with high probability. □

5 Performance Evaluation via Simulations

In order to show that our protocol WFF performs well in practice, we perform various benchmarks with varying weight distributions and adversarial strategies. The source code, and a description of how to run the benchmarks, can be found at https://github.com/guilhermemtr/Weighted-Flooding-Simulator.[12]

5.1 Scope of Simulations

Weight distributions. We consider weight distributions covering scenarios where parties have similar weights as well as scenarios with different weights. More concretely, we consider:

- The constant distribution (Const), characterized by the number of parties n. In this distribution all parties have equal weights and is therefore equal to the non-weighted setting. This serves as a baseline for our simulations.
- The exponential distribution (Exp), characterized by the number of parties n and the weight ratio r between the heaviest party and the lightest party. It corresponds to the (perhaps more realistic) exponential weight distribution—wherein the weights of parties form an exponential curve. More concretely, for $i \in \{1, \ldots, n-1\}$, the weight of p_{i+1} is $r^{-(n-1)}$ times the weight of p_i.
- The few heavy distribution (FH), characterized by the number of parties n, the weight ratio r between the heaviest and lightest party, and the number of heavy parties c. It corresponds to the distribution where $n - c$ parties have constant weight, and the other c parties have r times more weight. This weight distribution is meant to capture extreme scenarios.

Sender. In order to ensure that our protocol performs well *independently* of the weight of the sender, for the exponential distribution, we consider three choices for the sender: heaviest, lightest and median-weight party, and for the few heavy weight distribution, we consider both a heavy and a light party as the sender.

Corruption strategies. Given that parties in our protocol simply forward a message to their neighbors, we consider the worst behavior that prevents message propagation, i.e. corrupted parties simply do not send. We consider adversaries that can corrupt up to 50% of the total weight. To ensure that our protocol performs well independently of how adversaries spend their corruption budget, we consider adversaries that greedily corrupt as many parties as possible, following one of the strategies below:

- Random corruption (Rand) where the adversary corrupts parties uniformly at random.
- Light-First corruption (Light) where the adversary corrupts parties by their weight in increasing order, starting by the lighter ones.
- Heavy-First corruption (Heavy) where the adversary corrupts parties by their weight in decreasing order, starting from the heavier ones.

[12] All simulations were performed on the ETH Zurich Euler cluster, but there are no hindrances to running them on less powerful computers.

As one might note, for the constant weight distribution the corruption strategy is irrelevant. For this reason, for the constant weight distribution we only consider the random corruption strategy.

5.2 Methodology

To obtain statistical confidence, we make 10 000 runs for each parameter configuration (e.g. weight distribution, adversary strategy, choice of sender, number of parties, etc.). All runs are executed independently.

In the evaluations, a run is considered successful if the sender's message is delivered to *all* (honest and dishonest) parties. As one might note, this contrasts with the timely predicate (see Definition 3), which only requires a message to be delivered to all honest parties. Thus, the success rate metric we consider for the evaluations is actually a lower bound on the real success rate of our protocol. The rationale behind this definition is as follows: consider an adversary that corrupts a set C of parties; if the adversary would alternatively pick some party $p \in C$, and corrupt $C \setminus \{p\}$, then the protocol would have to guarantee that every honest party, including p still gets the message. Since p is now honest, it seems a harder requirement to make p now also receive the message. This justifies our choice of making adversaries corrupt as many parties as possible.

When counting successful runs, we do not take latency into account. The reason for this is that all our simulations have in common that once they succeed, they have a very low latency.[13] Further details on this and plots of the actual latency can be found in the full version of this paper [26].

To ensure our protocol performs well *independently* of the sender's weight, we take the worst result among the sender choices (for each weight distribution).

5.3 Simulations and Results

Comparison against weight-oblivious protocols. To compare the performance of WFF and a weight oblivious protocol, we measured the success rate for $\mathsf{WFF}(k)$ and a weight oblivious protocol $\mathsf{WOF} := \pi_{\mathrm{Flood}}(\mathsf{WFS}(\mathtt{E}, k))$ with $\mathtt{E}(p) := 1$ for different exponential weight distribution (with changing ratios between the heaviest and lightest party).[14] The results can be found in Fig. 1. The plot shows that our protocol (WFF) achieves 100% success rate at a much lower number of transmitted messages than the weight-oblivious one (WOF). Only exception is when the weight ratio between the heaviest and the lightest parties is 1, the exponential weight distribution is the same as the constant weight distribution, and hence the protocols become identical. Note, that while the WFF protocol achieves practical security with low message complexity regardless of the ratio

[13] The maximum latency observed in any of our simulations is $9 \cdot \delta_{\mathrm{Channel}}$ for any succeeding run

[14] The protocol $\mathsf{WOF} := \pi_{\mathrm{Flood}}(\mathsf{WFS}(\mathtt{E}, k))$ for $\mathtt{E}(p) := 1$ corresponds to the protocol where each party simply selects k parties uniformly at random as their neighbors without taking weight into account.

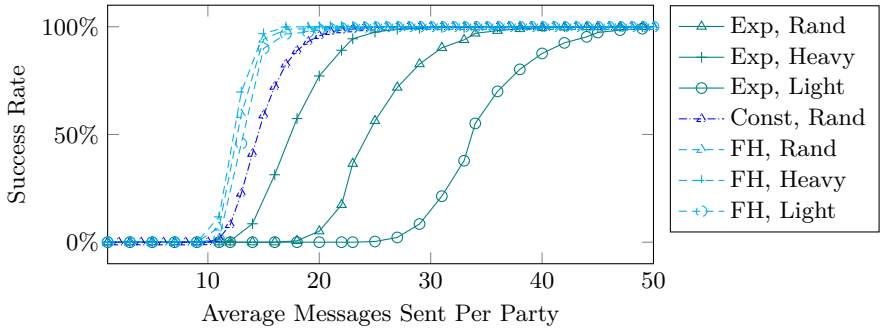

Fig. 2. Success rate of WFF protocol for different weight distributions and corruption strategies, depending on the average number of messages sent per party, for $n = 1024$ parties, a 50% corruption threshold, a ratio of 10^6 between heaviest and lightest parties and $c = 10$ heavy parties for FH.

between the heaviest and lightest party, the message complexity of WOF in order to achieve a 100% success rate, increases drastically as the ratio increases.

Performance for changing weight distributions. In Section 3.2, we bounded the message complexity of WFF by $\frac{2 \cdot n \cdot (\log(n)+\kappa)}{\gamma}$ (see Theorem 2), and in Section 4.2 we showed that this number of messages is inherent for the constant weight distribution (see Lemma 9), implying that WFF is optimal up to a constant factor for this distribution. Although the obtained upper-bound is independent of the weight, since it is tight only for the constant weight distribution, it could be that WFF performs poorly for other distributions. To show this is not the case, we measured the success rate for sending a single message in $\mathsf{WFF}(k)$ as a function of the message complexity (induced by adjusting k) for different weight distributions and corruption strategies. See Fig. 2.

Unsurprisingly, the adversarial strategy inducing the highest cost is corrupting as many light nodes as possible. This fits the intuition from Section 3.3: By corrupting as many light nodes as possible, an adversary can get a slight advantage in terms of the number of emulated nodes that they control because the ceiling embedded in the emulation function has a proportionally larger effect on such nodes. Furthermore, note that for the constant weight distribution $\mathsf{WFF}(k)$ simply selects k neighbors uniformly at random and at least $\lceil \gamma \cdot n \rceil$ of the parties remains honest. Hence, this corresponds to the performance that can be expected by additionally assuming that a certain fraction of the parties remains honest and use flooding protocols tailored to this setting. We emphasize that our protocol only induces marginally larger (within a small constant factor) message complexity for all the considered weight distributions and corruption strategies. This aligns with Section 3.3, where our bound on the message complexity for the weighted setting was only worse by a factor of 4 compared to the bound that relied on a constant fraction of honest parties. Therefore, security for our protocol in the weighted setting comes at a much lower cost comparatively.

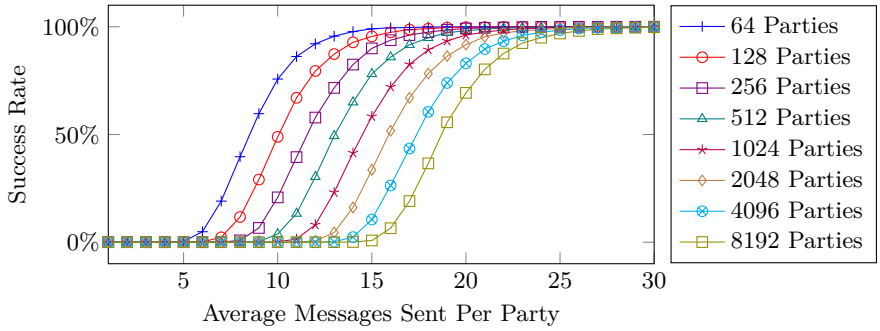

Fig. 3. Scalability of WFF protocol. We consider the constant weight distribution, and the random corruption strategy, with a 50% corruption threshold.

Scalability. A key feature of flooding protocols is their scalability. To benchmark the scalability of our proposed protocol WFF, we measured, for different numbers of parties, the success rate of the protocol depending on the average number of messages each party sends (again induced by varying k). For simplicity, we chose to only include the constant weight distribution (the performance for varying weight distributions is plotted in Fig. 2). The results can be found in Fig. 3. From the plot, it is clear that the message complexity (normalized by the number of parties) of our protocol only increases logarithmically in the number of parties, and hence this once again confirms our theory from Section 3.4.

By the time of writing, there are around 12k running Bitcoin nodes [1] and roughly 8k nodes in the Ethereum network [2]. Extrapolating from Figs. 2 and 3, it seems that independently of the stake distribution, WFF can realize a secure flooding network with an average number of connections per message of just $\sim$55 for such number of nodes. We conclude that this is within the realm of the number of connections existing widely used implementations maintain by default. Note, however, that the workload is not distributed evenly among nodes in WFF, as heavier nodes need to maintain more connections. In Section 4, we showed that this is inherent for this type of protocol in the weighted setting.

Acknowledgements

The work was in part done while Chen-Da Liu-Zhang was at Carnegie Mellon University and Søren Eller Thomsen was at Purdue University. Chen-Da Liu-Zhang was supported in part by the NSF award 1916939, DARPA SIEVE program, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award.

References

1. Bitnodes.io. https://bitnodes.io/ (2022), [Online; accessed 16-September-2022]

2. ethernodes.org. https://ethernodes.org/ (2022), [Online; accessed 16-September-2022]
3. Abraham, I., Malkhi, D., Nayak, K., Ren, L., Yin, M.: Sync hotstuff: Simple and practical synchronous state machine replication. In: IEEE Symposium on Security and Privacy. pp. 106–118. IEEE (2020)
4. Alangot, B., Reijsbergen, D., Venugopalan, S., Szalachowski, P., Yeo, K.S.: Decentralized and lightweight approach to detect eclipse attacks on proof of work blockchains. IEEE Trans. Netw. Serv. Manag. **18**(2), 1659–1672 (2021)
5. Apostolaki, M., Zohar, A., Vanbever, L.: Hijacking bitcoin: Routing attacks on cryptocurrencies. In: IEEE Symposium on Security and Privacy. pp. 375–392. IEEE (2017)
6. Badertscher, C., Gaži, P., Kiayias, A., Russell, A., Zikas, V.: Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. p. 913–930. CCS '18, ACM (2018). https://doi.org/10.1145/3243734.3243848
7. Ben-Hamou, A., Peres, Y., Salez, J.: Weighted sampling without replacement. Brazilian Journal of Probability and Statistics **32**(3), 657–669 (2018), https://www.jstor.org/stable/26496522
8. Bollobás, B.: Random Graphs. Cambridge Studies in Advanced Mathematics, Cambridge University Press, 2 edn. (2001). https://doi.org/10.1017/CBO9780511814068
9. Chandran, N., Chongchitmate, W., Garay, J.A., Goldwasser, S., Ostrovsky, R., Zikas, V.: The hidden graph model: Communication locality and optimal resiliency with adaptive faults. In: ITCS. pp. 153–162. ACM (2015)
10. Chandran, N., Garay, J.A., Ostrovsky, R.: Improved fault tolerance and secure computation on sparse networks. In: ICALP (2). Lecture Notes in Computer Science, vol. 6199, pp. 249–260. Springer (2010)
11. Chandran, N., Garay, J.A., Ostrovsky, R.: Almost-everywhere secure computation with edge corruptions. J. Cryptol. **28**(4), 745–768 (2015)
12. Chen, J., Micali, S.: Algorand: A secure and efficient distributed ledger. Theor. Comput. Sci. **777**, 155–183 (2019)
13. Coretti, S., Kiayias, A., Moore, C., Russell, A.: The generals' scuttlebutt: Byzantine-resilient gossip protocols. Cryptology ePrint Archive, Report 2022/541 (2022), https://ia.cr/2022/541
14. Daian, P., Pass, R., Shi, E.: Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In: Financial Cryptography. Lecture Notes in Computer Science, vol. 11598, pp. 23–41. Springer (2019)
15. David, B., Gazi, P., Kiayias, A., Russell, A.: Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 10821, pp. 66–98. Springer (2018)
16. Dinsdale-Young, T., Magri, B., Matt, C., Nielsen, J.B., Tschudi, D.: Afgjort: A partially synchronous finality layer for blockchains. In: SCN. Lecture Notes in Computer Science, vol. 12238, pp. 24–44. Springer (2020)
17. Dolev, D., Strong, H.R.: Authenticated algorithms for byzantine agreement. SIAM J. Comput. **12**(4), 656–666 (1983)
18. Dwork, C., Peleg, D., Pippenger, N., Upfal, E.: Fault tolerance in networks of bounded degree. SIAM J. Comput. **17**(5), 975–988 (1988)
19. Fenner, T.I., Frieze, A.M.: On the connectivity of random m-orientable graphs and digraphs. Combinatorica **2**(4), 347–359 (1982). https://doi.org/10.1007/BF02579431

20. Garay, J.A., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: Analysis and applications. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 9057, pp. 281–310. Springer (2015)
21. Garay, J.A., Ostrovsky, R.: Almost-everywhere secure computation. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 4965, pp. 307–323. Springer (2008)
22. Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse attacks on bitcoin's peer-to-peer network. In: USENIX Security Symposium. pp. 129–144. USENIX Association (2015)
23. Jayanti, S., Raghuraman, S., Vyas, N.: Efficient constructions for almost-everywhere secure computation. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 12106, pp. 159–183. Springer (2020)
24. Kermarrec, A., Massoulié, L., Ganesh, A.J.: Probabilistic reliable dissemination in large-scale systems. IEEE Trans. Parallel Distributed Syst. **14**(3), 248–258 (2003)
25. King, V., Saia, J., Sanwalani, V., Vee, E.: Towards secure and scalable computation in peer-to-peer networks. In: FOCS. pp. 87–98. IEEE (2006)
26. Liu-Zhang, C.D., Matt, C., Maurer, U., Rito, G., Thomsen, S.E.: Practical provably secure flooding for blockchains. Cryptology ePrint Archive, Paper 2022/608 (2022), https://eprint.iacr.org/2022/608, https://eprint.iacr.org/2022/608
27. Malkhi, D., Mansour, Y., Reiter, M.K.: On diffusing updates in a byzantine environment. In: SRDS. pp. 134–143. IEEE (1999)
28. Malkhi, D., Pavlov, E., Sella, Y.: Optimal unconditional information diffusion. In: DISC. Lecture Notes in Computer Science, vol. 2180, pp. 63–77. Springer (2001)
29. Marcus, Y., Heilman, E., Goldberg, S.: Low-resource eclipse attacks on ethereum's peer-to-peer network (2018), https://eprint.iacr.org/2018/236, https://eprint.iacr.org/2018/236
30. Matt, C., Nielsen, J.B., Thomsen, S.E.: Formalizing delayed adaptive corruptions and the security of flooding networks. In: Advances in Cryptology – CRYPTO 2022. Springer (2022), to appear
31. Minsky, Y., Schneider, F.B.: Tolerating malicious gossip. Distributed Comput. **16**(1), 49–68 (2003)
32. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review p. 21260 (2008)
33. Nayak, K., Kumar, S., Miller, A., Shi, E.: Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In: EuroS&P. pp. 305–320. IEEE (2016)
34. Pass, R., Shi, E.: Fruitchains: A fair blockchain. In: PODC. pp. 315–324. ACM (2017)
35. Pass, R., Shi, E.: Hybrid consensus: Efficient consensus in the permissionless model. In: DISC. LIPIcs, vol. 91, pp. 39:1–39:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017)
36. Pass, R., Shi, E.: Thunderella: Blockchains with optimistic instant confirmation. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 10821, pp. 3–33. Springer (2018)
37. Rohrer, E., Tschorsch, F.: Kadcast: A structured approach to broadcast in blockchain networks. In: Proceedings of the 1st ACM Conference on Advances in Financial Technologies, AFT 2019. pp. 199–213. ACM (2019). https://doi.org/10.1145/3318041.3355469
38. Tran, M., Choi, I., Moon, G.J., Vu, A.V., Kang, M.S.: A stealthier partitioning attack against bitcoin peer-to-peer network. In: IEEE Symposium on Security and Privacy. pp. 894–909. IEEE (2020)

39. Tsimos, G., Loss, J., Papamanthou, C.: Gossiping for communication-efficient broadcast. Cryptology ePrint Archive, Report 2020/894 (2020), https://ia.cr/2020/894
40. Upfal, E.: Tolerating a linear number of faults in networks of bounded degree. Inf. Comput. **115**(2), 312–320 (1994)
41. Xu, G., Guo, B., Su, C., Zheng, X., Liang, K., Wong, D.S., Wang, H.: Am I eclipsed? A smart detector of eclipse attacks for ethereum. Comput. Secur. **88** (2020)
42. Yagan, O., Makowski, A.M.: On the scalability of the random pairwise key predistribution scheme: Gradual deployment and key ring sizes. Perform. Evaluation **70**(7-8), 493–512 (2013). https://doi.org/10.1016/j.peva.2013.03.001
43. Zhang, S., Lee, J.: Eclipse-based stake-bleeding attacks in pos blockchain systems. In: BSCI. pp. 67–72. ACM (2019)
44. Zheng, H., Tran, T., Arden, O.: Total eclipse of the enclave: Detecting eclipse attacks from inside tees. In: IEEE ICBC. pp. 1–5. IEEE (2021)