# Protocols and Lower Bounds for Failure Localization in the Internet

Boaz Barak, Sharon Goldberg, and David Xiao

Princeton University, Princeton, NJ 08544

**Abstract.** A secure *failure-localization path-quality-monitoring* (FL-PQM) protocol allows a sender to localize faulty links on a single path through a network to a receiver, even when intermediate nodes on the path behave adversarially. Such protocols were proposed as tools that enable Internet service providers to select high-performance paths through the Internet, or to enforce contractual obligations. We give the first formal definitions of security for FL-PQM protocols and construct:

1. A simple FL-PQM protocol that can localize a faulty link *every time* a packet is not correctly delivered. This protocol's communication overhead is $O(1)$ additional messages of length $O(n)$ per packet (where $n$ is the security parameter).
2. A more efficient FL-PQM protocol that can localize a faulty link when a *noticeable fraction* of the packets sent during some time period are not correctly delivered. The number of additional messages is an arbitrarily small fraction of the total number of packets.

We also prove lower bounds for such protocols:

1. Every secure FL-PQM protocol requires *each* intermediate node on the path to have some shared secret information (*e.g.* keys).
2. If secure FL-PQM protocol exist then so do one-way functions.
3. Every *black-box* construction of a FL-PQM protocol from a random oracle that securely localizes every packet and adds at most $O(\log n)$ messages overhead per packet requires *each* intermediate node to invoke the oracle.

These results show that implementing FL-PQM requires active cooperation (*i.e.* maintaining keys and agreeing on, and performing, cryptographic protocols) from *all* of the intermediate nodes along the path. This may be problematic in the Internet, where links operate at extremely high speeds, and intermediate nodes are owned by competing business entities with little incentive to cooperate.
**Keywords.** Failure localization, secure routing, black-box separation.

## 1 Introduction

The Internet is an indispensable part of our society, and yet its basic foundations remain vulnerable to attack. Secure routing protocols seek to remedy this by not only providing guarantees on the correct setup of paths from sender to receiver through a network (*e.g.* secure BGP [16]), but also by verifying that data packets are actually delivered correctly along these paths. Packet delivery is surprisingly susceptible to simple attacks; in the current Internet, packets are typically sent along a single path from sender to receiver, and so a malicious node along the data path can easily drop or modify packets before they reach their destination. To detect and respond to such attacks, the networking community has recently been studying monitoring and measurement protocols that are used to obtain information about packet loss events on a data path (*e.g.* [2–5,7,18,19,21,23]). The motivation for such protocols is twofold. First, they provide the sender with information that he can use during path setup to select a single, high-performance path to the receiver from the multiple available paths through the network [11]. Second, since Internet service is a contractual business, where senders pay nodes along the data path to carry their packets, information from Internet measurement protocols is highly valuable for enforcing contractual obligations between nodes. In fact, Laskowski and Chuang [17] recently argued that this information is not only valuable, but also *necessary* to counter the Internet industry's growing trend towards degraded path performance. Note that if Internet measurement protocols are used to enforce contractual obligations, nodes may have an economic incentive to bias the information obtained from these protocols.

In this work we provide a rigorous cryptographic examination of *secure* monitoring protocols that are robust even in the presence of malicious nodes on the data path. In particular, we study techniques that allow a sender to *localize* the specific links along the data path where packets were dropped or modified— a task that we call *failure-localization path-quality monitoring*. While some protocols for this task are deployed in the Internet today (*e.g.* traceroute [1]), they are not robust to nodes that behave adversarially in order to bias measurements.

### 1.1 Our results

We make the following contributions to the study of secure failure-localization path-quality monitoring protocols (in the rest of the paper we call these simply *failure localization* or FL protocols). Throughout the paper, we use the word "packet" to denote data that the sender wishes to transmit, and "message" to refer to both data packets and FL-protocol-related messages.

**Definition.** In Section 2, we give the first formal definition of security for failure localization protocols. We note that some of the previous FL protocols suggested in the literature, such as [2, 4, 21], do *not* satisfy our definition. (We sketch attacks in Appendix A.)

We give two variants of the definition— *per-packet* security requires localizing a link each time a packet is not delivered, while *statistical* security only requires this when a noticeable fraction of packets fail to arrive. An important feature of our definition is that it accounts for the fact that messages can be dropped in the Internet for benign reasons like congestion. We note that care must be taken to design protocols that are simultaneously robust to both adversarial behaviour and benign congestion. We discuss the effect of this assumption on some previous work [4] in Appendix A.

**Protocols.** We present two simple protocols satisfying our per-packet (Section 3.1) and statistical (Section 3.2) security definitions. Both of these protocols do not modify the packets sent on the path; instead, they add additional messages. Thus our protocols have the important advantage of allowing backwards compatibility with the current techniques for processing packets in a router, minimizing latency in the router, and not increasing packet size.

Our main measure of efficiency for such protocols is communication overhead—the number and size of messages added by the protocols. The per-packet protocol adds a single $O(n)$-length message to every packet sent ($n$ is the security parameter), and $O(K)$ additional $O(n)$-length messages when a failure occurs (where $K$ is the number of nodes on the path). The statistical protocol only needs $O(K)$ additional $O(n)$-length messages per $T$ packets sent. In our setting $K$ is constant, while $T$ could be $\mathrm{poly}(n)$, which implies the statements in the abstract.

**Lower bounds.** Like many of the protocols in the literature [2–4, 19, 21], both of our protocols require cryptographic keys and computations at each node. These requirements are considered severe in the networking literature; setting up a key infrastructure and agreeing on cryptographic primitives is challenging in the distributed world of the Internet, where each node is owned by a different entity with sometimes incompatible incentives. However, in Section 4 we show that these requirements are to some degree *inherent* by:

1. Proving that every secure (per-packet or statistical) FL protocol requires a key infrastructure, or more precisely, that intermediate nodes and Alice and Bob must all share some secret information between each other.
2. Proving that a one-way function can be constructed from any secure FL protocol.
3. Giving evidence that any practical per-packet secure FL protocol must use these keys in a cryptographic way at *every node* (*e.g.* , it does not suffice to use the secret information with some simple, non-cryptographic, hash functions as in [7]). We show that in every black-box construction of such a protocol from a random oracle, where at most $O(\log n)$ protocol messages are added per packet, then every intermediate node must query the random oracle. We note that known protocols designed for Internet routers currently avoid using public-key operations, non-black-box constructions, or adding more than a constant number of protocol messages per packet. We also show that for statistically-secure FL, or FL protocols adding $\omega(\log n)$ messages per packet, the necessity of cryptography depends on subtle variations in the security definition.

**Implications of our results.** Our lower bounds raise questions about the practicality of deploying FL protocols. In small highly-secure networks or for certain classes of traffic, the high key-management and cryptographic overhead required for FL protocols may be tolerable. However, FL protocols may be impractical for widespread deployment in the Internet; firstly because intermediate nodes are owned by competing business entities that may have little incentive to set up a key infrastructure and agree on cryptographic protocols, and secondly because cryptographic computations are expensive in the core of the Internet, where packets must be processed at extremely high speeds (about 2 ns per packet).

Thus, our work can be seen as a motivation for finding security functionalities for the Internet that are more practical than failure localization.

### 1.2 Related work

Some of this work (in particular, the results of Section 3 and a weaker version of Theorem 5) appeared in our earlier technical report [8]. We built on [8] in [9], where, together with Jennifer Rexford and Eran Tromer, we gave formal definitions, constructions, and lower bounds for the simpler task of *path-quality monitoring* (PQM). In a PQM protocol the sender only wishes to *detect* if a failure occurred, rather than localize the specific faulty link along the path. We use the results from [8, 9] in Section 3.2 to show how a PQM protocol can be composed to obtain a statistical FL protocol, and in Section 4.2 to argue that FL protocols need cryptographic computations.

In addition to the FL protocols from the networking literature [2–4,19,21,24], our work is also related to the work on secure message transmission (SMT) begun by Dolev, Dwork, Waart, and Yung in [6]. In SMT, a sender and receiver are connected by a multiple parallel wires, any of which can be corrupted by an adversary. Here, we consider a single path with a series of nodes that can be corrupted by an adversary, instead of multiple parallel paths. Furthermore, while multiple parallel paths allow SMT protocols to *prevent* failures, in our single path setting, an adversarial intermediate node can always block the communication between sender and receiver. As such, here we only consider techniques for *detecting and localizing* failures.

## 2 Our model

In a failure localization (FL) protocol, a sender Alice wants to know whether the packets she sends to receiver Bob arrive unmodified, and if not, to find the link along the path where the failure occurred (see Figure 1). We say a *failure* or *fault* occurs when a data packet that was sent by Alice fails to arrive unmodified at Bob. Following the literature, we assume that Alice knows the identities of all the nodes of the data path. We work in the setting where all traffic travels on symmetric paths (*i.e.* intermediate nodes have bi-directional communication links with their neighbors, and messages that sender Alice sends to receiver Bob traverse the same path as the messages that Bob sends back to Alice). We say that messages travelling towards Alice are going *upstream*, and messages travelling towards Bob are going *downstream*. An adversary Eve can occupy any set of nodes on the path between Alice and Bob, and can add, drop, or modify messages sent on the links adjacent to any of the nodes she controls. She can also use timing information to attack the protocol.

**Localizing links, not nodes.** It is well known that an FL protocol can only pinpoint a *link* where a failure occurred, rather than the *node* responsible for the failure. To see why, refer to Figure 1, and suppose that (a) Eve controlling node $R_2$ becomes unresponsive by ignoring all the messages she receives from $R_1$. Now suppose that (b) Eve controls node $R_1$ and pretends that $R_2$ is unresponsive by dropping all communication to and from $R_2$. Because cases (a) and (b) are completely indistinguishable from Alice's point of view, at best Alice can localize the failure to link $(1, 2)$.

$$\boxed{R_A \text{ (Alice)}} \rightleftarrows \boxed{R_1} \rightleftarrows \boxed{R_2} \rightleftarrows \ldots \rightleftarrows \boxed{R_K} \rightleftarrows \boxed{R_B \text{ (Bob)}}$$

**Fig. 1.** A path from Alice to Bob via $K$ intermediate nodes.

**Congestion.** Congestion-related packet loss is widespread on the current Internet, caused by protocols like TCP [15] that naturally drive the network into a state of congestion. Our definition accounts for congestion by assuming links can drop each message independently with some probability. One could come up with other models for congestion (*e.g.* allowing Eve to specify the distribution of congestion-related packet loss), and for some plausible choices our positive results will still hold. However, we use independent drops for the sake of simplicity. Furthermore, assuming that congestion is not controlled by the adversary only strengthens our negative results and makes our model more realistic.

### 2.1 Security definition

Let $n$ be the security parameter. A failure localization protocol consists of an efficient initialization algorithm $\mathsf{Init}$ taking $n$ uniformly random bits and generating keys for each node, and efficient node algorithms $\mathsf{Alice}, \mathsf{Bob}, R_1, \ldots, R_K$ which take in a key and communicate with each other as in Figure 1. We always fix $K = O(1)$ independent of $n$.[1] The $\mathsf{Alice}$ algorithm takes in a packet that she wants to send to Bob. If communication is successful, then the $\mathsf{Bob}$ algorithm outputs the packet that Alice sent. Our security definitions are game-based:

**Definition 1 (Security game for FL).** The game begins when Eve chooses a subset of nodes $E \subseteq \{1, \ldots, K\}$ that she will occupy for the duration of the game. The $\mathsf{Init}$ algorithm is then used to generate keys for each node, and Eve is given the keys for the nodes $i \in E$ that she controls. We define an oracle $\mathsf{Source}$ that generates data packets $d$ for the $\mathsf{Alice}$ algorithm to send. We allow Eve to choose the packets that the $\mathsf{Source}$ oracle generates, subject to the condition that she may not choose the same packet more than once during the game.[2] We allow Eve to add, drop, or modify any of the messages sent on the links adjacent to the nodes she occupies. We include congestion in our model by requiring that, for each message sent on each link on the path, the link *goes down* or drops the

[1] Typically in the Internet, the path length $K$ is less than 20 when nodes represent individual routers, and when nodes represent Internet Service Providers (ISPs) then there are on average $K \approx 4$, and no more 7 nodes on a typical path [16].

[2] We make this assumption because there is natural entropy in packet contents, due to TCP sequence numbers and IP ID fields [7]. To enforce this assumption in practice, protocol messages can be timestamped with with an expiry time, such that with high probability (over the entropy in the packet contents), no repeated packets are sent for the duration of the time interval for which the protocol messages are valid.

message with some constant probability $\rho > 0$. Notice that this means that a failure can happen at links not adjacent to a node occupied by Eve.

We introduce the notion of time into our model by assuming that the game proceeds in discrete timesteps; in each timestep, a node can take in an input and produce an output, and each link can transmit a single message. (Thus, each timestep represents an event occurring on the network.) Because it is expensive to have securely synchronized clocks in a distributed system like the Internet,[3] we do *not* allow the honest algorithms to take timing information as an input. However, to model timing attacks, we assume that Eve knows which timestep that the game is in.

Then, our per-packet security definition uses the the game defined in Definition 1:

**Definition 2 (Per-packet security for FL).** In the per-packet security game, Eve gets to interact with the Source oracle and the "honest" node algorithms as in Definition 1, until she decides to stop. For each packet sent, Alice must output either $\surd$ (*i.e.* not raise an alarm) or a link $\ell$ (*i.e.* raise an alarm and localize a failure to $\ell$). We assume that the game is *sequential:* Alice must output a decision for each data packet before starting to transmit the next data packet (see remarks below). We say that an FL protocol is *per-packet secure* if the following hold:

1. (*Secure localization*). For every packet $d$ sent by the Source oracle that is not successfully output by Bob, then Alice outputs a link $\ell$ such that either (a) link $\ell$ is adjacent to a node occupied by Eve, or (b) link $\ell$ went down due to congestion for one of the messages (including FL protocol messages) associated with sending packet $d$ from Alice to Bob.
2. (*No false positives*). For every packet $d$ sent by the Source oracle that is successfully output by Bob, for which there was no congestion, and for which Eve does not deviate from the protocol, Alice outputs $\surd$.

We need to introduce a few new concepts for our statistical security definition. First, we define an *interval* as a sequence of $T$ packets (and associated FL protocol messages) that Alice sends to Bob.[4] Next, we use the following parameters: a false alarm threshold $\alpha$, a detection threshold for the path $\beta$ (where $0 < \alpha < \beta < 1$) and an error parameter $\delta \in \{0, 1\}$. Usually, we will set $\alpha$ such that congestion alone almost never causes the failure rate on a path to exceed the false alarm threshold.

**Definition 3 ($(\alpha, \beta, \delta)$-Statistical security for FL).** In the statistical security game, Eve is allowed to choose the number of *intervals* for which she wants to interact with the Source oracle and the honest nodes as in Definition 1. The number of packets per interval $T$ may grow with $n$, but is always at least some

[3] Indeed, the NTP protocol used for clock synchronization on the Internet is not secure [12], and thus should not be used as an input to a secure FL protocol.

[4] We can think of an interval as all the packets sent in some time period (*e.g.* approximately $10^7$ packets are sent 100 msec over a 5 Gbps Internet path).

minimum number depending $\alpha, \beta, \delta, K$. At the end of each *interval*, Alice needs to output either $\surd$ (*i.e.* not raise an alarm) or a link $\ell$ (*i.e.* raise an alarm and localize a link). The game is sequential; Alice must output a decision for each interval before starting the next interval. Then, an FL protocol is *statistically secure* if the following hold:

1. (*Secure localization*). For any interval in the security game where Eve causes the failure rate on the path to exceed the detection threshold $\beta$, then with probability $1 - \delta$ Alice raises alarm for a link $\ell$ that is adjacent to Eve, or a link $\ell$ whose failure rate exceeds $\frac{\alpha}{K+1}$.
2. (*Few false positives*). For any interval in the security game where Eve does not deviate from the correct algorithm $R_i$ of any of the nodes $i \in E$ that she controls and the failure rate on each link is below the (per-link) false alarm threshold $\frac{\alpha}{K+1}$, then the probability that Alice outputs $\surd$ is at least $1 - \delta$.

We now discuss some properties of our security definition.

**Benign and malicious failures.** Our security definitions require Alice to accurately localize failures, but these failures may be caused by Eve, or may be the result of *benign causes*, such as congestion. We do not require Alice to distinguish between benign or malicious (*i.e.* due to Eve) failures, because Eve can always drop packets in a way that "looks like" congestion.

**Sequential games.** For simplicity, in our per-packet security game we required Alice to make FL decisions before she sends a new data packet. This is to capture the fact that such protocols should provide "real-time" information about the quality of the paths she uses, and so we did not allow Alice in the per-packet case to make decisions only after sending many packets (as is done in the statistical security case). We note that while our negative results (*i.e.* attacks) are sequential, our positive results (*i.e.* , protocols) do not use the assumption of sequential execution in any way, and are secure in a more general setting where Eve can choose at each point in time which of the previously sent packets "time-out", and then Alice needs to output FL decisions for these packets. We emphasize that the sequential assumption does *not* prevent Alice from keeping state and using information from *past* packets in order to make FL decisions. (Though none of our positive results require that Alice does this.)

**Movements of the adversary.** Our model does not allow Eve to move from node to node in a single security game. This assumption makes sense when Eve models a Internet service provider that tries, for business reasons, to bias the results of FL protocol. Furthermore, when Eve is an external attacker or virus that compromises a router, "leaving" a router means that the legitimate owner of the router removed the attacker from the router, *e.g.* by refreshing its keys. We model this key refresh process as a re-start of the security game. Furthermore, in practice "movements" to a new router happen infrequently, since an external attacker typically needs a different strategy each time it compromises a router owned by a different business entity.

**Generalizations.** All our results generalize to the setting where congestion rates, false alarm thresholds, and detection thresholds are different per link; we set them all equal here for simplicity. Our negative results also hold for the weaker adversary model where Eve can occupy only one node and the Source oracle generates independent (efficiently-samplable) packets from a distribution that is *not* controlled by Eve.

## 3 Protocols

We now present protocols for secure per-packet and statistical FL. Our protocols are related, though not identical to those of [2–4]. (In Appendix A we show that the protocols in [2, 4] do not satisfy our security definitions.)

We use the notation $[m]_k$ to denote a message $m$ authenticated by a key $k$ using a *message authentication code* (MAC); such schemes can be constructed from any one-way function [10, 22]. We'll often use the well-known notion of an *onion report*: if every node $R_i$ wants to transmit a report $\tau_i$ to Alice in an authenticated way, then we define inductively $\theta_{K+1} = [(K+1, \tau_{\mathsf{Bob}})]_{k_{\mathsf{Bob}}}$ and for $1 \leq i \leq K$, $\theta_i = [(i, \tau_i, \theta_{i+1})]_{k_i}$. That is, each $R_i$'s report is appended with its downstream neighbors' reports before being authenticated and passed upstream. Onion reports prevent Eve from selectively dropping reports — if Eve occupies $R_j$ and wants to drop the report $\tau_j$ of $R_i$ for some $i > j$ then, under the assumption that Eve cannot forge MACs, Alice will discover that $R_j$ tampered with the onion report. We also note that every time we send or store a packet $d$ in acknowledgments and reports, we could save space by replacing $d$ with an $O(n)$-length hash of $d$ via some collision-resistant hash function, where $n$ is the security parameter.

### 3.1 Optimistic Per-Packet FL Protocol

We assume that each node $R_i$ shares a symmetric key $k_i$ with Alice. For each packet that Alice sends, the protocol proceeds in two phases:

**The detect phase.** Alice stores each packet $d$ that she sends to Bob. When Bob receives the packet $d$, he responds with an ack of the form $a = [d]_{k_B}$. Alice removes the the packet $d$ from storage when she receives a validly MAC'ed corresponding ack, and raises an alarm if a valid ack is not received.[5] We also require each intermediate node to store each data packet and corresponding ack.

**The localize phase.** This phase is run only if Alice raises an alarm for a packet $d$. Alice sends an *onion report request* $q = (\mathsf{report}, d)$ downstream towards Bob. To respond to the request, each node $R_i$ checks if he stored data packet $d$; if he did, $R_i$ sets $\tau_i = (q, i, d, a)$ where $a$ is the ack he saw corresponding to packet $d$, and substituting the symbol $\perp$ for $d$ and/or $a$ if he failed to to receive that packet or an ack. $R_i$ then creates an onion report $\theta_i$ using $\tau_i$ as described above. In the onion report, $R_i$ can substitute the symbol $\theta_{i+1} = \perp$ if he fails to receive a $\theta_{i+1}$ from $R_{i+1}$.

[5] In practice, each packet $d$ should be stored along with a local timeout at Alice. If the ack does not arrive before the timeout expires, then Alice should raise an alarm.

To localize the failure, Alice classifies the onion reports that she received in response to her onion report request $q$. An onion report $\theta_i = [q', i', d', a', \theta_{i+1}]_{k_i}$ is "consistent" if it is present, *i.e.* $\theta_i \neq \perp$, and all of the following four conditions hold. Otherwise, an onion report is "inconsistent".

1. $q' = q$ sent out by Alice.
2. The MAC on $\theta_i$ is valid.
3. $d' = d$, where $d$ is the packet queried in $q$.
4. $a'$ is *not* a valid ack for packet $d$.

Alice localizes then localizes the upstream-most link $(i, i+1)$ where the onion reports transition from consistent to inconsistent.

**Theorem 1.** *The optimistic FL protocol is per-packet secure.*

The proof follows via a simple reduction to the security of the MAC, and is defered to the full version. We remark that the detect phase of this protocol requires a large amount of storage and communication overhead at each node. This high overhead makes this protocol impractical for regular Internet traffic; however, it might be useful for specialized highly-secure networks, or for certain classes of traffic *e.g.* network management traffic.

### 3.2 A Composition Technique for Statistical FL

We now consider statistical security protocols, that apply results from our previous work on statistical PQM [8,9] to obtain statistical FL protocols with much lower overhead. In a statistical PQM protocol, Alice *detects* whenever the average failure rate exceeds a threshold $\beta$ (but she need not localize a link).

Here we show how to compose the lightweight PQM protocols we presented in [9] to obtain a statistical FL protocol. While it is possible to give a very general composition theorem, for clarity and concreteness in this version we describe only how to compose the simpler symmetric secure sampling (SSS) protocol of [9]. We defer our more general composition result to the full version of this paper. In particular, we can compose the *secure sketch protocol* of [9] that has a communication overhead of only a single $O(\log T + n)$ length packet for every interval, thus yielding the result stated in Section 1.1.

**Symmetric Secure Sampling (SSS), a statistical PQM protocol from [8, 9].** SSS requires Alice and Bob to securely designate a random $p$ fraction of the data packets that Alice sends to Bob as "probes", and require that Bob send MAC'd acknowledgments for all the probes. We call $p$ the *probe frequency.* To do this, Alice and Bob share a secret $k = (k_1, k_2)$. For each packet $d$ that Alice sends to Bob, they use $k_1$ to compute a function $\mathsf{Probe}$ that determines whether or not a packet $d$ is a probe and should therefore be stored, and acknowledged. To acknowledge a probe, Bob sends Alice an ack $[d]_{k_2}$ that is MAC'ed using $k_2$. The $\mathsf{Probe}$ function is implemented using a pseudorandom function (PRF) $f$ keyed with $k_1$, that we think of as mapping strings to integers in $[0, 2^{n-1}]$; we define $\mathsf{Probe}_{k_1}(d)$ output "Yes" if $f_{k_1}(d) < p2^n$ and output "No" otherwise.

For each interval, Alice stores each probe packet (*i.e.* each packet $d$ such that $\mathsf{Probe}_{k_1}(d)$ =Yes). At the end of the interval, after $T$ packets are sent, Alice computes $V$, a count of the number of stored (probe) packets for which she failed to receive a valid ack. She computes the average failure rate as $\frac{V}{pT}$.

**A composition that does not work.** Perhaps the most natural approach to construct a statistical FL protocol is to have Alice run $K$ simultaneous PQM protocols with each of the intermediate nodes, and use the statistics from each protocol to infer behaviour at each link (similar to [4, 21, 24]). However, we now show that this composition is vulnerable to the following *timing attack*: Suppose a packet $d$ that Alice sends to Bob is ack'd by innocent node $R_j$ with message $a$. Then, if Eve occupies node $R_i$ for $i < j-1$, she can determine that $R_j$ originated the ack $a$ by counting the timesteps that elapsed between the timestep in which she saw $d$ and timestep in which she saw $a$. Then, Eve can implicate $R_j$ by selectively dropping every ack that originates at $R_j$. Notice that this attack results from the structure of this composition, and cannot be prevented even when acks are encrypted.[6] In practice, this attack can be launched when isolated burst of packets triggers a separate burst of acks at each intermediate node.

**Composing PQM to statistical FL.** We require that every node $R_i$ shares pairwise keys $k_i^A, k_i^B$ with Alice and Bob respectively. Using $k_i^B$, each intermediate node runs a statistical PQM protocol with Bob with the following modification: whenever Bob decides to send an ack for a packet $d$ to an intermediate node $R_i$, Bob will (1) always address the ack to Alice and (2) MAC the ack in onion fashion, starting with $k^B_{\mathsf{Alice}}$ (on the inside of the onion) and ending with $k^B_K$ (on the outside of the onion). Each node forwards all acks upstream, and processes only the ack he expects. At the end of the interval $u$, Alice will send an onion report request $q = (\mathsf{report}, u)$ to all the intermediate nodes. Each intermediate node produces a MAC'd onion report $\theta_i = [q, i, V_i, \theta_{i+1}]_{k_i^A}$ where $V_i$ is his estimate of the average failure rate on the path between himself and Bob. Letting $\alpha, \beta$ be the false alarm and detection thresholds, when Alice receives the final onion report $\theta_1$, she computes $F_\ell = V_i - V_{i+1}$ for each link $\ell = (i, i+1)$, and outputs $\ell$ if $F_\ell > \frac{\alpha+\beta}{2(K+1)}$, or if $\ell = (i, i+1)$ is the upstream-most link when the onion report $\theta_{i+1}$ refers to the wrong interval, is missing, or is invalidly MAC'ed.

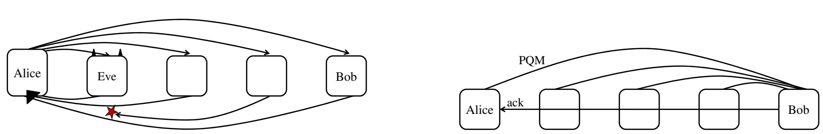


**Fig. 2.** On the left an insecure composition, on the right our secure composition.

We prove that this scheme is secure provided that the interval length $T$ is long enough and the congestion rate $\rho$ is small enough.

**Theorem 2.** *The composition of SSS described above with probe frequency $p$ satisfies $(\alpha, \beta, \delta)$-strong statistical security when each interval contains at least $T = O(\frac{K^2}{p(\beta-\alpha)^2} \ln \frac{K}{\delta})$ packets and the congestion rate satisfies $\beta - \alpha \gg K\rho$.*

[6] [24] deals with this by randomizing the sending time of acks.

*Proof.* First, observe that the probability that any efficient adversary Eve successfully forges an ack for a dropped packet by forging a MAC used in SSS is negligible. As in the Optimistic Protocol, the probability that any efficient adversary Eve successfully forges the onion report of an honest node (by forging the MAC on the onion report) is negligible as well. Hence, for the rest of this proof assume that Eve does not forge an ack to a dropped packet or validly forge the onion report of an honest node. Moreover, we can assume that Eve does not tamper with the onion report, or else she will implicate a link adjacent to one of the nodes she controls. We now work within a single interval:

- Let $V_i$ be $R_i$'s *estimate* of the failure rate between $R_i$ and Bob.
- Let $D_i$ be a count of the number of packets that were dropped or modified on the path between $R_i$ and Bob.
- Let $C_i$ be the number of acks intended for *any node* that were dropped or modified on the path between Bob and $R_i$.
- Let $p' = \frac{p}{1-(1-p)^{K+1}}$ be the probability that a node $R_i$ expects an ack to a packet $d$ (*i.e.* $\mathsf{Probe}_{k_i^B}(d) = \text{Yes}$) conditioned on there being at least one node expecting an ack to packet $d$ (*i.e.* $\exists j \in \{0, \ldots, K\},\ \mathsf{Probe}_{k_j^B}(d) = \text{Yes}$).[7]

Note that when $R_i$ estimates the average failure rate on the path from $R_i$ to Bob, she is unable to distinguish between dropped packets and dropped acks. Also, it is possible that $D_i > D_{i+1}$ or $C_i > C_{i+1}$ for two adjacent uncorrupted nodes because of congestion. In the absence of adversarial behavior at $R_i$, the expectation of the estimator $V_i$ that Alice receives in the onion report is $\frac{1}{T}(D_i + \frac{p'}{p}C_i)$. Finally, notice that the average failure rate on link $(i, i{+}1)$ is $\frac{1}{T}(D_i - D_{i+1})$.

Set $\gamma = \frac{\beta-\alpha}{2(K+1)}$. If $T = O(\frac{K^2}{p(\beta-\alpha)^2} \ln \frac{K}{\delta})$ then we have the following lemmata:

**Lemma 1 (Deviation of the estimator $V_i$).** *For each $i \notin E$ where $E$ is the set of nodes corrupted by Eve it holds (up to negligible error) that*

$$\Pr\left[\left|V_i - \tfrac{1}{T}(D_i + \tfrac{p'}{p}C_i)\right| > \tfrac{1}{4}\gamma\right] < \tfrac{\delta}{4(K+1)}$$

**Lemma 2 (Acks dropped due to congestion).** *For each $i, i+1 \notin E$, it holds (up to negligible error) that*

$$\Pr\left[\tfrac{p'}{p}\tfrac{C_i - C_{i+1}}{T} > \tfrac{\gamma}{2}\right] < \tfrac{\delta}{2(K+1)}$$

The proofs of these lemmata are technical, but not difficult. We defer them to the full version. Both proofs are applications of the Chernoff bound under the assumption that the $\mathsf{Probe}$ function is implemented with a truly random function; the negligible error refers the difference between a PRF and a truly random function. The proof of Lemma 1 relies on the fact that Eve cannot bias

[7] This quantity is the probability that a node $R_i$ samples an ack that was dropped between $R_i$ and $R_B$, since at least one node must have sampled the corresponding packet in order for the ack to be transmitted at all.

node $R_i$'s estimate of $C_i$ by selectively dropping acks because (1) acks destined for different nodes look identical, and they all originate at Bob (so that an adversary cannot use timing to distinguish between them), and (2) acks are onion MAC'd, so the adversary cannot selectively tamper with an ack intended for an upstream node. The proof of Lemma 2 also relies on the fact that $\beta - \alpha \gg K\rho$.

**Few false positives:** To prove this, we consider an interval where all the nodes on the path behave honestly, and show that, with probability at least $1-\delta$, Alice will not raise an alarm during this "honest interval".

Consider link $\ell = (i, i+1)$ where the average failure rate is less than the false alarm threshold so $\frac{1}{T}(D_i - D_{i+1}) < \frac{\alpha}{K+1}$. We now show that Alice will not raise an alarm for this link $\ell$ by proving that Alice's estimate of the failure rate for $\ell$, *i.e.* $V_i - V_{i+1}$, does not exceed her alarm decision threshold, *i.e.* $\frac{\alpha+\beta}{2(K+1)}$. We do this by proving that

$$\Pr\left[\left|(V_i - V_{i+1}) - \tfrac{1}{T}(D_i - D_{i+1})\right| > \tfrac{\alpha+\beta}{2(K+1)} - \tfrac{\alpha}{K+1} = \gamma\right] < \tfrac{\delta}{K+1} \tag{3.1}$$

Notice that "Few false positives" condition follows from (3.1) by a union bound over all $K+1$ links.

To prove (3.1), we start with the expression below, and apply the triangle inequality, and then Lemma 1:

$$\begin{aligned}
&\Pr[|(V_i - V_{i+1}) - (\tfrac{D_i - D_{i+1}}{T} + \tfrac{p'}{p}\tfrac{C_i - C_{i+1}}{T})| > \gamma/2] \\
&\quad \le \Pr[|V_i - \tfrac{1}{T}(D_i + \tfrac{p'}{p}C_i)| > \gamma/4] + \Pr[|V_{i+1} - \tfrac{1}{T}(D_{i+1} + \tfrac{p'}{p}C_{i+1})| > \gamma/4] \\
&\quad \le \tfrac{\delta}{2(K+1)}
\end{aligned} \tag{3.2}$$

Next, from Lemma 2 we know that $\Pr[\frac{p'}{p}\frac{C_i - C_{i+1}}{T} > \gamma/2] \le \frac{\delta}{2(K+1)}$, and so a union bound over this expression and (3.2) proves (3.1).

**Secure localization:** We now show that if Eve drops more than a $\beta$ fraction of packets in any interval, then Alice will catch her with probability at least $1-\delta$. Since the actual failure rate on the path is $\frac{1}{T}D_A > \beta$, we start by applying Lemma 1 to find that Alice's estimate of the failure rate is $V_A > \beta - \frac{\gamma}{4}$ with probability at least $1 - \frac{\delta}{4(K+1)}$. We now use an averaging argument to claim that there exists some link $\ell = (i, i+1)$ such that $V_i - V_{i+1} > \frac{\alpha+\beta}{2(K+1)}$. To see why, suppose for the sake of contradiction that for all $i$ we had $V_i - V_{i+1} \le \frac{\alpha+\beta}{2(K+1)}$. Then, it follows that

$$V_A = \sum_{i=0}^{K}(V_i - V_{i+1}) \le \sum_{\ell} \tfrac{\alpha+\beta}{2(K+1)} = \tfrac{\alpha+\beta}{2} < \beta - \tfrac{\gamma}{4}$$

where $V_{K+1} = 0$ (Bob's estimate of drops to himself is 0). But this contradicts our condition that $V_A > \beta - \frac{\gamma}{4}$, so there is at least one link $\ell = (i, i+1)$ with $V_i - V_{i+1} > \frac{\alpha+\beta}{2(K+1)}$ so that Alice raises an alarm.

Next, recall that we assume that for any link where the true failure rate due to congestion less than $\frac{\alpha}{K+1}$, we have from our proof of the "Few false positives" condition that with probability $\frac{\delta}{K+1}$, Alice does not raise an alarm for link $\ell$ between two honest nodes. Then, Alice must have raised the alarm for a link adjacent to Eve with probability at least $1-\delta$ (by a union bound) or a link with actual fauilure rate larger than $\frac{\alpha}{K+1}$, and secure localization follows. ■

## 4 Lower bounds

We now argue that in any secure per-packet FL scheme Alice requires shared keys with Bob and the intermediate nodes, and Alice, Bob and each intermediate node must perform cryptographic operations. We only argue for intermediate nodes $R_2, \ldots, R_K$; $R_1$ is a border case which requires neither keys nor crypto because we assume Alice is always honest.

### 4.1 Failure Localization Needs Keys at Each Node

Since FL provides strictly stronger security guarantees than path-quality monitoring, it follows from the results in [9] that in any secure FL protocol, Alice and Bob must have shared keys. We also have the following theorem that proves that in any secure FL protocol, *each* intermediate node must share keys with some Alice:

**Theorem 3.** *Suppose* $\mathsf{Init}$ *generates some auxiliary information* $\mathsf{aux}_i$ *for each node* $R_i$ *for* $i = 1, ..., K, \mathsf{Alice}, \mathsf{Bob}$. *A FL protocol cannot be (per-packet or statistical) secure if there is any node* $i \in \{2, \ldots, K\}$ *such that* $(\mathsf{aux}_{\mathsf{Alice}}, \mathsf{aux}_1, \ldots, \mathsf{aux}_{i-1})$ *and* $\mathsf{aux}_i$ *are independent.*

*Proof.* Suppose $R_i$ has $\mathsf{aux}_i$ that is independent of $(\mathsf{aux}_{\mathsf{Alice}}, \ldots, \mathsf{aux}_{i-1})$. Then, the following two cases are indistinguishable from Alice's view: (a) Node $R_{i+1}$is malicious and blocks communication on link $(i, i+1)$, and (b) Eve occupies node $R_{i-1}$, and drops packets while simulating case (a) by picking an independent $\mathsf{aux}'_i$ and running $R_i(\mathsf{aux}'_i)$ while pretending as if $(i, i+1)$ is down. These two cases are indistinguishable because $\mathsf{aux}_i$ is independent of $(\mathsf{aux}_{\mathsf{Alice}}, \ldots, \mathsf{aux}_{i-1})$, and so Alice will localize the failure to the same link in both case (a) and (b). But this breaks security, since $R_{i+1}, R_{i-1}$ do not share a common link. ■

### 4.2 Failure Localization Needs Crypto at Each Node

In [9], we give a reduction from one-way functions to secure PQM, proving:

**Theorem 4 (From [9]).** *The existence of a per-packet secure PQM protocol implies the existence of an infinitely-often one-way function (i.o.-OWF).*

Since one-way functions are equivalent to many cryptographic primitives (in the sense that these primitives exist if and only if one-way functions exist [13]), this result can be interpreted to mean that nodes participating in any secure PQM protocol must perform cryptographic computations. Since FL gives a strictly stronger security guarantee than PQM, we also have that in any FL protocol, some node on the data path must perform cryptography. However, Theorem 4 only implies that the *entire system* performs cryptography. We want to prove that any secure FL protocol requires *each intermediate node* $R_1, \ldots, R_K$ to perform cryptography. Because it is not clear even how to formalize this in full generality, we instead apply the methodology of Impagliazzo and Rudich [14] to do this for *black-box* constructions of FL protocols from a random oracle $\mathsf{RO}$. We model "performing cryptography" as querying the random oracle, and show that in such a secure FL protocol *each node* must query the $\mathsf{RO}$.

In [14], Impagliazzo and Rudich showed that there can be no secure black-box construction of key agreement (KA) from a random oracle. They argued that if any such KA construction is secure, then it must also be secure in a relativized world where every party has access to a random oracle $\mathsf{RO}$, and a $\mathsf{PSPACE}$ oracle. (A $\mathsf{PSPACE}$ oracle solves any PSPACE-complete problem, *e.g.* True Quantified Boolean Formulae (TQBF)). Intuitively, in this $(\mathsf{PSPACE}, \mathsf{RO})$ world, every computation is easy to invert *except* for those computed by the $\mathsf{RO}$. They obtain their result by showing, for every possible black-box construction of KA from a random oracle, that there exists an efficient algorithm (relative to $(\mathsf{PSPACE}, \mathsf{RO})$) that breaks the security of KA. Using the the same reasoning, any secure black-box FL protocol constructed from a $\mathsf{RO}$ must remain secure even relative to a $(\mathsf{RO}, \mathsf{PSPACE})$ oracle. Then, to obtain our result, it suffices to exhibit an efficient algorithm (relative to $(\mathsf{PSPACE}, \mathsf{RO})$) that breaks security of any black-box FL protocol where one node does not call $\mathsf{RO}$. We do this below.

We will use the notion of an *exchange* to denote a data packet and all the FL-protocol-related messages associated with that packet. Because our game is sequential (see Section 2), Alice's must decide to localize a link $\ell$ or output $\surd$ before the next exchange begins. We now prove that a per-packet FL protocol with $r = O(\log n)$messages per exchange must invoke the random oracle at every node. We note that protocols where number of messages per packet grows with $n$ are impractical and so "practical" protocols should use $r = O(1)$ messages per exchange. (See Remark 1 below on the possibility of extending this result to statistical security and/or protocols with $\omega(\log n)$ messages per exchange.)

**Theorem 5.** *Fix a fully black-box per-packet FL protocol that uses access to a random oracle* $\mathsf{RO}$*, where at least one node* $R_i$ *for* $i \in \{2, \ldots, I\}$ *never calls the* $\mathsf{RO}$ *and where the maximum number of messages per exchange is* $O(\log n)$*. Then there exists an efficient algorithm relative to* $(\mathsf{PSPACE}, \mathsf{RO})$ *that breaks the security of the scheme with non-negligible probability over the randomness of* $\mathsf{RO}$ *and the internal randomness of the algorithm.*

The proof of Theorem 5 is quite technical and is deferred to the full version. We sketch the proof, which resembles that of Theorem 3. Eve controls node $R_{i-1}$ and impersonates $R_i$, but now $\mathsf{aux}_i$ is secret, so Eve must first *learn* $\mathsf{aux}_i$:

1. *Learning to impersonate.* Sitting at $R_{i-1}$, Eve observes $t$ exchanges ($t$ is polynomial in $n$), where Eve asks $\mathsf{Source}$ to transmit a uniformly random data packet. She then uses the learning algorithm of Naor and Rothblum [20] to obtain a pair of impersonator algorithms $A', B'$, whose interaction generates

a distribution over transcripts for the $t+1$'th exchange. $A'$ impersonates nodes $\mathsf{Alice}, R_1, \ldots, R_{i-1}$ and $B'$ impersonates nodes $R_i, \ldots, R_K, \mathsf{Bob}$.
2. *Dropping and impersonating.* On the $t+1$'th exchange, for each message $m$ going from $R_{i-1}$ to $R_i$, Eve computes a response $m'$ herself using algorithm $B'$ and returns $m'$ to $R_{i-1}$; she does not send any messages to $R_i$.

Now, Eve at $R_{i-1}$ will break security if she manages to use $B'$ to impersonate an *honest* exchange during which link $(i, i+1)$ is down. (This breaks security since link $(i, i+1)$ is not adjacent to $R_{i-1}$.) The crucial observation is that here, Eve need only impersonate node $R_i$, and that $R_i$ does not "protect" its secret keys by calling the RO. Intuitively, Eve should be able to impersonate $R_i$ since any computations that $R_i$ does are easy to invert in the $(\mathsf{PSPACE}, \mathsf{RO})$ world. We now argue that Eve can break security with non-negligible probability.

Recall (Section 2) that Alice is allowed to use information from past exchanges to help her decide how to send messages in new exchanges. Fortunately, the algorithm of Naor and Rothblum [20] is specifically designed to deal with this, and guarantees that observing $t = \text{poly}(n/\varepsilon)$ many exchanges (in Step 1) Eve can obtain, with probability $1-\varepsilon$, algorithms $A', B'$ that generate an impersonated transcript that is $\varepsilon$-statistically close to the "honest" transcript of messages on the link $(i-1, i)$ (generated by interactions of honest $\mathsf{Alice}, R_1, ..., R_K, \mathsf{Bob}$.)

Suppose Eve obtained an $A', B'$ that satisfy the guarantee above. Our first challenge is that the Naor-Rothblum algorithm does *not* guarantee that $A', B'$ generates an impersonated transcript that is statistically close to the "honest" transcript of messages on $(i-1, i)$ when *the observer has access to the* RO. Fortunately, with probability $\rho^r$ all the messages sent from $R_i$ to $R_{i-1}$ are computed without access the RO. This happens when *congestion* causes link $(i, i+1)$ to go down for the duration of an exchange (so that $R_i$, who never calls the RO, has to compute all his upstream messages on his own).

Our next challenge is that Eve has no control, or even knowledge, of when congestion causes this event to occur. Indeed, the distribution generated by $A', B'$ is only guaranteed to be close to the honest transcript overall; there is no guarantee that it is close to the honest transcript *conditioned on congestion on* $(i, i+1)$. Fortunately, we can show that with probability $\rho^r$, $A', B'$ will generate a "useful" impersonated transcript that is $\varepsilon/\rho^r$-statistically close to the honest transcripts conditioned on the event that link $(i, i+1)$ is down. Eve does not necessarily know *when* she impersonates a useful transcript; she simply has to hope that she is lucky enough for this to happen.

The last challenge is that even when Eve is lucky enough to obtain a useful transcript, we still need a guarantee that (a) conditioned on $B'$ generating a useful transcript, using $B'$ to *interact* with the honest algorithm $R_{i-1}$ results in a transcript that is statistically close to (b) the transcript between honest algorithms $R_{i-1}$ and $R_i$ conditioned on link $(i, i+1)$ being down. Unfortunately, the Naor-Rothblum algorithm does not give any guarantees when an honest algorithm *interacts* with an impersonated algorithm for more than 1 round. Thus, we prove that, with probability at least $(\rho/2)^r$, the impersonator algorithm $B'$ interacting with honest $\mathsf{Alice}, ...R_{i-1}$ still generates a useful transcript such that the statistical distance between (a) and (b) is at most $1/100$. (This assumes we take $\varepsilon$ small enough; $\varepsilon = (\rho/10)^{4r} = 1/\text{poly}(n)$ suffices.)

To summarize, with probability $\geq 99/100$ Eve obtains algorithms $A', B'$ from the Naor-Rothblum algorithm that can successfully impersonate *all* the honest algorithms. Then, with probability roughly $(\rho/2)^r$, she can use $B'$ to *interact* with $R_{i-1}$ as in Step 2 to drop a packet at $R_{i-1}$ and generate a *useful* impersonated transcript that is $1/100$-statistically close to the honest transcript produced when $R_{i-1}$ and $R_i$ interact conditioned on link $(i, i+1)$ being down. This breaks security with non-negligible probability, since link $(i, i+1)$ is not adjacent to Eve at $R_{i-1}$. ■

**Statistical security.** Our negative results in the statistical setting are more subtle. First of all, from [8, 9] the analog of Theorem 4 also holds, showing that the *entire system* needs to "perform cryptography". However, we run into trouble when we try to show that cryptography is required at *each intermediate node*. It turns out that Definition 3 does *not* inherently require complexity-based cryptography at intermediate nodes. We sketch a statistically secure FL protocol where the intermediate nodes $R_1, \ldots, R_K$ use only information-theoretically secure primitives (although Alice and Bob still use regular MAC's). While this protocol is completely impractical in terms of communication and storage overhead, we present it here to demonstrate the subtleties of Definition 3.[8]

*Remark 1 (Impractical "crypto-free" statistical FL protocol.).* The protocol uses one-time MACs (OTMAC), information-theoretic objects that have the same properties as regular MACs except that they can only be used a single time. (OTMACs and can be constructed from Carter-Wegman hashing.) Each node $R_i$ shares pairwise keys with Alice. All the intermediate nodes and Bob store each packet that Alice sends to Bob. For each packet, Bob replies with an ack signed using a regular MAC. At the end of the interval, Alice counts the number of acks that she either fails to receive, or are invalid. The first time this count exceeds a $\beta$-fraction, Alice sends a "report request" message that is signed using a OTMAC to $R_1, \ldots, R_K, R_{K+1}$. Each node $R_1, \ldots, R_K$ responds with a report of every single packet they have witnessed, that is "onion signed" using the OTMAC (as in Section 3.1). Alice uses these reports in the usual way to localize link $\ell$ adjacent to Eve. From this point onwards Alice simply counts valid acknowledgments from Bob, and blames link $\ell$ each time the count exceeds a $\beta$ fraction.

The protocol satisfies Definition 3 because the probability that the failure rate at any link exceeds $\beta$ by congestion alone is negligible. Since we do not allow Eve to move during the security game, if Alice successfully localizes Eve to link $\ell$ once, it means it must have been Eve's fault, and so from then on Alice can always blame all failures on link $\ell$. As noted above, similar "impractical" protocols exist

[8] In concurrent work, Wong et al. [24] propose a statistical FL scheme where no cryptography is performed *during an interval*. Instead, they precompute shared secrets that are appended to packets over the course of an interval and are used guarantee security. The secrets must refreshed periodically, which requires cryptographic participation by the intermediate nodes. This contrasts with the impractical scheme we describe here, which truly *never* requires any intermediate node to perform crypto.

for per-packet protocols with $\omega(\log n)$ additional messages per packet (since all $\omega(\log n)$ messages are lost to congestion with only negligible probability), except that we replace the idea of "exceeding $\beta$ fraction of failures" with "losing an entire exchange due to congestion". We may interpret this as follows:

1. It is unreasonable to assume that the failure rate at a link exceeds $\beta$ only due to adversarial behaviour (*i.e.* Eve). For example, occasionally congestion might spike, or a router might malfunction or go down due maintenance, causing more than a $\beta$-fraction of packets to be dropped. If we assume such events happen with non-negligible probability, we can adapt the proof of Theorem 4 to show that cryptography is necessary at intermediate nodes for statistical security. As a corollary, if Eve can control congestion at links she does not occupy, then we need cryptography at every intermediate node. Our FL protocols remain secure even under the strongest such definition, where the failure rate on a link not occupied by Eve can exceed $\beta$.
2. We can take this issue outside of our model. If we say that it is reasonable that Eve cannot move during the security game, and that the failure rate cannot exceed $\beta$ on a link that Eve does not control, then, as we showed above, there exist protocols where the intermediate nodes do not use complexity-based cryptography. However, we must be cognizant that in the real world there can be multiple adversaries that we would like to localize correctly, or the adversary may be able to move from one link to another. If protocols that do not use cryptography at intermediate nodes are to remain secure after Eve moves (and learns the key of previous nodes she occupied), then the keys at each node should be refreshed periodically. This key refresh process would require each intermediate node to use cryptography.

## 5 Open problems

We gave lower bounds on the key-management and cryptographic overhead of secure FL protocols. The problem of bounding the *storage* requirements in an FL protocol is also still open. Furthermore, our results here only apply to FL on *single symmetric paths* between a single sender-receiver pair. An interesting question would be to consider FL for *asymmetric paths*, where the packets Bob sends back to Alice may take a different path than the packets that Alice sends to Bob. Another interesting direction is to consider FL in networks where packets can travel simultaneously on *multiple paths*, as in the SMT framework [6].

**Acknowledgements.** We thank Jennifer Rexford and Eran Tromer for very useful discussions and collaborations on failure localization and routing security in general. Shai Halevi and the anonymous EuroCrypt reviewers made a number of very helpful comments to improve the presentation of this paper. Boaz Barak is supported by NSF grants CNS-0627526 and CCF-0426582, US-Israel BSF grant 2004288 and Packard and Sloan fellowships. Sharon Goldberg is supported by NSF grant CNS-0627526. David Xiao is supported by a NSF Graduate Research Fellowship and a NDSEG Graduate Fellowship, and part of this work was done while visiting the Institute for Pure and Applied Mathematics at UCLA.

## References


1. Traceroute. *Available: http://costard.lbl.gov/cgi-bin/man/man2html?traceroute+8*, Sept 2001.
2. K. Argyraki, P. Maniatis, D. Cheriton, and S. Shenker. Providing packet obituaries. *ACM HotNets-III*, 2004.
3. I. Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy. Highly secure and efficient routing. *INFOCOM 2004*, 1:–208, 7-11 March 2004.
4. B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens. An secure routing protocol resilient to byzantine failures. In *WiSE '02*, pages 21–30. ACM, 2002.
5. M. Crovella and B. Krishnamurthy. *Internet Measurement*. Wiley, 2006.
6. D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *J. ACM*, 40(1):17–47, 1993.
7. N. G. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. *IEEE/ACM Trans. Netw.*, 9(3):280–292, 2001.
8. S. Goldberg, D. Xiao, B. Barak, and J. Rexford. A cryptographic study of secure internet measurement, TR-783-07. Technical report, Princeton University, Department of Computer Science, May 2007.
9. S. Goldberg, D. Xiao, B. Barak, J. Rexford, and E. Tromer. Path-quality monitoring in the presence of adversaries. In *ACM SIGMETRICS*, 2008.
10. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
11. J. He and J. Rexford. Towards Internet-wide multipath routing. *IEEE Network Magazine Special Issue on Scalablity*, March 2008.
12. IETF. Network time protocol (ntp) charter. *Available: http://www3.ietf.org/proceedings/05mar/ntp.html*.
13. M. Impagliazzo, R.; Luby. One-way functions are essential for complexity based cryptography. *FOCS*, pages 230–235, Oct 1989.
14. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *STOC '89*, pages 44–61. ACM, 1989.
15. V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM '88*, pages 314–329, Stanford, CA, Aug. 1988.
16. S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol (S-BGP). *IEEE Jour. Sel. Areas in Comm.*, 18(4):582–592, April 2000.
17. P. Laskowski and J. Chuang. Network monitors and contracting systems: competition and innovation. In *SIGCOMM '06*, pages 183–194. ACM, 2006.
18. R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. User-level internet path diagnosis. *SIGOPS Oper. Syst. Rev.*, 37(5):106–119, 2003.
19. A. Mizrak, Y.-C. Cheng, K. Marzullo, and S. Savage. Fatih: detecting and isolating malicious routers. *DSN 2005*, pages 538–547, 28 June-1 July 2005.
20. M. Naor and G. N. Rothblum. Learning to impersonate. In *ICML 2006*, pages 649–656. ACM, 2006.
21. V. N. Padmanabhan and D. R. Simon. Secure traceroute to detect faulty or malicious routing. *SIGCOMM Comput. Commun. Rev.*, 33(1):77–82, 2003.
22. J. H. stad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.
23. L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. H. Katz. Listen and whisper: security for BGP. In *USENIX NSDI 2004*, pages 10–10, 2004.
24. E. L. Wong, P. Balasubramanian, L. Alvisi, M. G. Gouda, and V. Shmatikov. Truth in advertising: Lightweight verification of route integrity. In *PODC*, 2007.

## A Vulnerabilities of Other FL Protocols

We sketch why the protocols of [2, 4, 21] do not satisfy our security definition.

**An On-demand Secure Routing Protocol Resilient to Byzantine Failures [4]:** Awerbuch, Holmer, Nita-Rotaru and Rubens present a statistical FL protocol in which Alice and Bob run a secure *failure detection* protocol, where Bob sends out authenticated acks for each packet he receives. Once the number of detected faulty exchanges exceeds some threshold, say $\beta$, then Alice appends a encrypted list of "probed nodes" to each *new* packet that she sends out. If a node is included in the list of probed nodes, it is expected to send Alice an ack when it receives the packet containing the list. The acks are formed as our "onion reports". To localize failures, Alice chooses probed nodes according to a binary search algorithm, until she localizes a single link.

Now, consider an adversary Eve that sits at $R_i$ and, for every sent packet where $R_i$ is not included in the list of probed nodes, Eve happily causes failures. Eve stops causing failures whenever $R_i$ is included in the list of probed nodes. Alice will never be able to localize such an Eve to a single link; as long as Eve behaves herself when she is part of the list of probed nodes, Alice has no way to find her. Our protocols avoid this problem by running their "detection phases" and "localization phases" on the same set of packets.

Furthermore, care must be taken in implementing this protocol in the presence of both adversarial behaviour and benign congestion. To see why, suppose that Eve causes the protocol to enter the localization phase. In [4], the binary search algorithm proceeds by one step each time failures are detected. It is important to ensure that normal congestion (on a link that is not adjacent to Eve) cannot cause the binary search algorithm to search for Eve in the wrong part of the path. To do this, the binary search algorithm should proceed by one step only when the *failure rate* exceeds some carefully chosen false alarm threshold (related to loss rate caused by normal congestion and the length of the portion of path that is currently being searched).

**Packet Obituaries [2]:** Argyraki, Maniatis, Cheriton, and Shenker propose an FL protocol that is similar to our Optimistic Protocol of Section 3.1. Each node locally stores digests of the packets they see, and at the end of some time interval, nodes send out reports to Alice that contain these packet digests. Alice then uses the information from these reports to localize failures on the path. The designers of this protocol focused on the benign setting, but mentioned that reports should also be *individually authenticated*. However, because these reports are not formed in a onion manner (as in our Optimistic Protocol) an adversarial node can implicate a innocent downstream node by selectively dropping the innocent node's reports.

**Secure Traceroute [21]:** We sketch attacks in the full version; this protocol has many of the same problems as [2, 4].