Path-Quality Monitoring in the Presence of Adversaries: The Secure Sketch Protocols

Sharon Goldberg, David Xiao, Eran Tromer, Boaz Barak, and Jennifer Rexford

ABSTRACT

Edge networks connected to the Internet need effective monitoring techniques to inform routing decisions and detect violations of Service Level Agreements (SLAs). However, existing measurement tools, like ping, traceroute, and trajectory sampling, are vulnerable to attacks that can make a path look better than it really is. Here we design and analyze a lightweight *path-quality monitoring protocol* that reliably raises an alarm when the packet-loss rate exceed a threshold, even when an adversary tries to bias monitoring results by selectively delaying, dropping, modifying, injecting, or preferentially treating packets. Our protocol is based on sublinear algorithms for sketching the second moment of stream of items, and can monitor billions of packets using only 250–600 bytes of storage and the periodic transmission of a comparably sized IP packet. We also show how this protocol can be used to construct a more sophisticated protocol that allows the sender to localize the link responsible for the dropped packets. We prove that our protocols satisfy a precise definition of security, analyze their performance using numerical experiments, and derive analytic expressions for the trade-off between statistical accuracy and system overhead. This paper contains a deeper treatment of results from earlier conference papers [11], [24], and several new results.

I. INTRODUCTION

Path-quality monitoring is a crucial component of flexible routing techniques (*e.g.,* intelligent route control, source routing, and overlay routing) that give edge networks greater control over path selection. Monitoring is also necessary to verify that service providers deliver the performance specified in Service-Level Agreements (SLAs). In both applications, edge networks need to determine when path quality degrades beyond some threshold, in order to switch from one path to another or report an SLA violation. The problem is complicated by the presence of nodes along the path who try to interfere with the measurement process, out of greed, malice, or just misconfiguration. Here we design and analyze light-weight *path-quality monitoring* (PQM) protocols that detect when packet loss exceeds a threshold, even when adversaries try to bias monitoring results. We also study *failure localization* techniques that allow a sender to localize the specific links along a path where packets were dropped or modified; while protocols for this task are available today (*e.g.,* traceroute), they can easily be gamed in the adversarial setting we consider. Our PQM protocol requires surprisingly little storage and communication, and is intended to run at line rate at the network layer on high-speed routers.

S. Goldberg is with the Department of Computer Science at Boston University, D. Xiao is with CNRS, LIAFA and Université Paris Diderot - Paris 7, E. Tromer is with the School of Computer Science at Tel Aviv University, B. Barak is with Microsoft Research, New England, and J. Rexford is with the Computer Science Department at Princeton University.

Manuscript submitted November 25, 2013. Revised June 8, 2014 and July 15, 2014.

A. The presence of adversaries

Today, path-quality monitoring relies on active measurement techniques, like ping and traceroute, that inject special probe packets into the network. In addition to imparting extra load on the network, active measurements are vulnerable to adversaries that bias the results by correctly transmitting and responding to probe packets while dropping and damaging normal network traffic. Here we design protocols that provide accurate information even when intermediate nodes *adversarially delay, drop, modify, inject or preferentially treat* packets in order to confound measurement. Our motivations for studying this adversarial threat model are threefold:

1. It covers active attacks. Our strong threat model covers a broad class of malicious behavior. Malicious adversaries can attack routing in order to draw packets to (or through) a node of their choosing [17], or compromise a routers along an existing path through the Internet [22], [27]. Biasing path-quality measurements allows the adversaries to evade detection, while continuing to degrade performance or impersonate the legitimate destination at will. In addition, ISPs have both the economic incentive and the technical means to preferentially handle probe packets, to hide discrimination against unwanted traffic like Skype [39] or BitTorrent [48], and evade detection of SLA violations. (In fact, commercial monitoring services, like Keynote, claim to employ "anti-gaming" techniques to prevent providers from biasing measurement results [30].) Finally, adversaries controlling arbitrary end hosts can add spoofed packets to the stream of traffic from one edge network to another, to confound simplistic measurement techniques (*e.g.,* maintaining a counter of received packets).

2. It covers all possible benign failures. In the adversarial setting, we avoid making *ad hoc* assumptions about the nature of failures caused by normal congestion, malfunction or misconfiguration. Even benign modification of packets may take place in a seemingly adversarial manner. For example, an MTU mismatch may cause a router to drop large packets while continuing to forward the small probe packets sent by ping or traceroute [34]. As another example, link-level CRC checks are surprisingly ineffective at detecting the kinds of

errors that corrupt IP packets [45]. Since the adversarial model is the strongest possible model, any protocol that is robust in this setting is automatically robust to all other kind of failures.

3. It is challenging to satisfy in high-speed routers. We choose to work in a difficult space, where we assume the strongest possible adversarial model, and yet design solutions for high-speed routers on multi-Gbit/sec links, where computation and storage resources are extremely limited. We view it as an important research goal to understand what can and cannot be done in this setting, to inform practical decisions about what level of threats future networks should be designed to withstand. Furthermore, designing protocols for this adversarial setting is not simply a matter of adding standard cryptographic tools to existing non-adversarial measurement protocols. Indeed, naive ways of combining such protocols with cryptographic tools may be either insecure or less efficient (*e.g.,* encrypting and authenticating all traffic).

B. Our results

Secure sketch PQM. Despite this strong threat model, we present a secure PQM protocol that is competitive, in terms of storage and communication, with solutions designed for significantly weaker threat models [16], [21]. Our secure sketch PQM protocol, presented in Section III-B, allows a pair of cooperating sender and receiver routers to *detect* when the fraction of *packet delivery failures* on a path between them rises above a certain rate β (say $\beta = 0.01$). Our protocol achieves its low overhead by combining sublinear algorithms for sketching the second moment of a stream of items [1], [3], [15], [47] with simple cryptographic primitives like message authentication codes and pseudorandom functions.

The protocol, which is run by a sender and receiver sharing a symmetric secret key, only requires the transmission of two small control messages for every T data packets sent, while its storage overhead is limited to a single sketch (*i.e.,* an array of counters) of size $O(\log T)$ bits. If $T = 10^7$ packets are sent during an 100ms interval, our protocol requires between 250–600 bytes of storage at the source and destination, and control messages that can easily fit into a single IP packet. Moreover, our protocol does not mark, modify or authenticate data packets in any way, and so it may be implemented off the critical packet-processing path in the router. Not marking packets also makes our protocol backwards compatible with IP, minimizes latency at the router, allows the parties to turn on/off PQM protocols without the need to coordinate with each other, and avoids problems with increasing packet size and possibly exceeding the MTU. As such, we believe that this protocol is strong candidate for deployment in future networks, even in networks where our strong security guarantees are not be essential.

The performance and cost of any particular implementation of protocol would depend on memory speed and the particular choice of cryptographic primitives. As such, we analytically bound the different resources consumed —computation, storage and communication— (Section III-D), and also show somewhat better bounds through numerical experiments (Section III-E). In the course of this analysis we also present a new analysis of [15]'s sketching scheme that may be of independent interest (Theorem III.2). Moreover, the security of our protocol is based on the computation of cryptographic hash over the contents of every packet send during an interval. Fortunately, our protocols use cryptographic hash functions in an *online* setting, where an adversary has very limited time to break the security before the hash parameters are refreshed; in Section III-F we discuss exploiting this for fast packet hashing.

Composing sketches for secure FL. Next, we use the secure sketching protocol to construct a secure fault localization (FL) protocol (Section V-A). Our FL protocol is designed for a trusted sender and receiver that send traffic over a symmetric path of K nodes, where $\sqrt{K}$ nodes may be adversarial; if the rate of packet-delivery failures exceeds β, the sender can localize the responsible link. (Depending on the application, nodes could represent routers (and thus $K < 20$) or Autonomous Systems (and thus $K \approx 4$ on average).) The protocol requires each node on the path to share a symmetric key with the sender, and requires each node to store an $O(K^2 \log T)$-sized sketch. The communication overhead of the protocol is still only two control messages for every T packets sent, and no modifications to data packets are required.

Precise definitions of security. Evaluating the security of a protocol is often challenging. In many problem domains, *e.g.,* intrusion detection, the only viable approach is to enumerate a set of possible attacks, and then show how the protocol defends against these specific attacks. Fortunately, in our problem domain, a more comprehensive security evaluation is possible; we can give a precise definition of the functionality we require from the protocol, and then guarantee that the protocol can carry out these functions *even in the face of all possible attacks by an adversary* with a specific set of powers. Thus, we precisely define the powers that we give to the adversary and our requirements for secure PQM and FL protocols (Sections II-A, IV-A). To evaluate security, we prove that our protocols achieve their required functionalities, *no matter what* the adversary does, short of breaking the security of the basic cryptographic primitives (*e.g.,* message authentication codes and hash functions) from which the protocol is constructed (Section III-C, V-B).

C. Bibliographic note and omitted results

This paper is a deeper treatment of a subset of results that appeared in earlier conference papers [11], [24]. Specifically, the security definitions we present here are the same as those used in these earlier papers. The secure sketch PQM protocol we discuss here was mentioned in [24], but the earlier paper concentrated on showing generic reduction from PQM to second moment estimation. Here we deepen and simplify the presentation by focusing on an instantiation of the secure sketch protocol using [15]'s sketching scheme, and present new results on fast packet hashing (Section III-F) and stronger versions of theorems from [24] (Theorem III.2 and Theorem D.1 in our technical report [25]). We also present completely new results showing the secure sketch PQM can be composed to create an FL protocol (Section V).

In focusing on the construction of efficient protocols, we have chosen to omit a number of negative results that we developed in [11], [24]. These negative results indicate that *any* secure PQM or FL protocol would need to employ the same basic security machinery (secret keys and cryptographic operations) used by our protocols. Specifically:

1. In [24] we showed that any PQM protocol satisfying Definition II.1 requires (1) Alice and Bob to have some form of shared secret (*e.g.,* shared symmetric secret keys), and (2) the shared secret must be used in a "cryptographically-strong" manner. To prove the latter, we used a reduction that shows that any secure PQM protocol is at least as complex as a secure keyed identification scheme (KIS). Because KIS are equivalent to cryptographic tasks like encryption and message authentication [28], this proves that cryptographic computations are necessary for the security of *any* PQM protocol.

2. In [11] we show that any FL protocol satisfying Definition IV-A requires a key infrastructure, or more precisely, that intermediate nodes and Alice and Bob must all share some secret information. We also used block-box separation techniques from cryptography [29] to give evidence that a secure FL protocol must use these keys in a cryptographically-strong manner at *every* node on the path.

In addition to these lower bounds, we also omitted a number of protocols from in [11], [24], that are less efficient in terms of storage and communication that the ones presented here, including (1) secure PQM and FL protocols that use PRF-based packet sampling, (2) PQM protocols that use public-key cryptography and timed information release to remove the need for symmetric shared secrets between Alice and Bob, and (3) FL protocols that are designed to detect and localize failures on a per-packet basis.

Online Appendices. Due to space limitations, we have had to defer the appendices of this paper to our technical report [25].

II. Threat Model for Path Quality Monitoring

We first define security for path quality monitoring.

A. Security definition for path-quality monitoring (PQM)

In our model, a source *Alice* sends packets to a trusted destination *Bob* over a path through the Internet. Fix a set of T consecutive packets sent by Alice, which we call an *interval*, we define a *packet delivery failure* to be any instance where a packet that was sent by Alice during the interval fails to arrive unmodified at Bob (before the last packet in the interval arrives at Bob). An adversary *Eve* can sit anywhere on the path between Alice and Bob, and we empower Eve to drop, modify, or delay every packet, or to add her own packets. A *path quality monitoring (PQM) protocol* allows Alice and Bob to detect when the number of failures during the interval exceeds a certain fraction of total packets transmitted.

Definition II.1. Given parameters $0 < \alpha < \beta < 1$ and $0 < \delta < 1$, we say a protocol is a (α, β, δ) *secure PQM protocol* if, letting T be the number of packets sent during the interval:

1) (Few false negatives.) In the *malicious case*, where an adversary Eve can drop, modify, delay, or add packets, the protocol must raise an alarm with probability at least $1 - \delta$ if more than βT packet-delivery failures occur.

2) (Few false positives.) In the *benign case*, where no intermediate node is adversarial (*i.e.,* no packets are added or modified on the path, but packets may be reordered / dropped due to congestion), the protocol must alarm with probability at most δ if at most αT packet-delivery failures occur.

We assume that the T packets sent during an interval are distinct, because of natural variation in packet contents, and the fact that even successive packets sent by the same host have different ID fields in the IP header [21] (note that even retransmissions of the same TCP segment correspond to distinct IP packets, because of the IP ID field).

B. Properties of our security definition

Our definition is motivated by our intended application of enabling routing decisions or detecting SLA violations. It's most important security guarantee is that, regardless of Eve's actions, Eve cannot prevent Alice from raising an alarm when the failure rate for packets that Alice sent to Bob exceeds β. As such, our definition encompasses attacks by nodes on the data path that include (but of course are not limited to): colluding nodes that work together to hide packet loss, an adversarial node that intelligently injects packets based on timing observations or deep packet inspection, a node that preferentially treats packets that it knows are part of the PQM protocol, and a node that masks packet loss by injecting an equal number of nonsense packets onto the data path. We emphasize that we never make any assumptions on the distribution of packet loss on the path; our model allows for any possible failure model, including one where, say, packet loss is correlated across different packets.

On the other hand, we only require PQM protocols to detect failures (so that SLA violations can be confirmed, or packets can be rerouted) but not to *prevent* them. Moreover, PQM protocols must only detect if the number of failures exceeds a certain threshold, rather than determining exactly how many failures occurred. (While solutions that exactly count failures certainly exist, they typically require cryptographically authenticating and/or encrypting all traffic, which we wish to avoid here.) Third, we do not require our protocols to distinguish between packet failures occurring due to adversarial tampering or due to benign congestion or malfunction.

Next, while our security definition requires that our protocols do not raise a (false) alarm when the one-way failure rate is less than α for the *benign setting*, we do allow for the possibility of raising an alarm due to adversarial tampering even when fewer than an α-fraction of failures occur. This is because Eve can always make a path look worse by selectively dropping all PQM protocol messages (*e.g.,* acknowledgments, report messages) that Bob sends to Alice, even if all the original packets that Alice sent to Bob were actually delivered.[1] In such cases, the PQM protocol will raise an alarm, and

[1] We will assume that any acknowledgment or report messages that Bob sends to Alice are sent repeatedly to ensure that, with high probability, they are not dropped due to normal congestion.

the router should look for a different path. We also do not model denial of service attacks, where an adversary exhausts capacity by flooding the path with packets; these attacks can be addressed using standard techniques, *e.g.,* rate limiting.

Choosing α, β. In principle $\alpha, \beta > 0$ can be chosen arbitrarily. However, several practical issues constrain the choice of these parameters. In Section III-D we find that the communication and storage overhead of our protocols is related to the ratio $\frac{\alpha}{\beta}$; a smaller ratio leads to less overhead. The value of α is also constrained by issues related to interval synchronization; we discuss this in Appendix A.

III. Secure sketch PQM

In our secure sketch PQM protocol, Alice and Bob aggregate traffic Alice sends to Bob into a short data structure called a *sketch*. At the end of the interval, Bob sends his sketch to Alice and she compares the sketches to decide whether the failure rate exceeded β. We first discuss notation and the cryptographic building blocks used in our protocol (Section III-A), and then describe the protocol (Section III-B). Next, we discuss its security (Section III-C) and derive analytic bounds that explain the relationship between the protocol's security its storage and communication overhead (Section III-D). We further analyze this relationship using numerical experiments (Section III-E) and conclude by discussing techniques for fast packet hashing (Section III-F).

A. Preliminaries

Notation. We use the notation $[\kappa]$ to represent the set of integers $\{1, ...\kappa\}$. $\mathbf{v}$ is a vector, and its i^{th} entry is v_i. The second moment of a vector $\mathbf{v}$ is $\|\mathbf{v}\|_2^2 = \sum_i (v_i)^2$, and the first moment of a vector is $|\mathbf{v}|_1 = \sum_i |v_i|$. We say that a quantity w (ε, δ)-*estimates* a quantity v if

$$\Pr[(1-\varepsilon)v \leq w \leq (1+\varepsilon)v] \geq 1-\delta$$

We also use the following cryptographic building blocks:

Pseudorandom Function (PRF). A PRF is a keyed function $\mathrm{PRF}_k(\cdot)$ that maps an arbitrary length string to an n-bit string using a key k [26]. If the key k is chosen uniformly at random, then to an adversary with no knowledge of k, the output of the function $\mathrm{PRF}_k(\cdot)$ looks totally unpredictable and cannot be distinguished (except with an insignificant probability) from a truly random function, where each input is mapped to a independent uniformly-random output. Hence, in our analysis we may treat $\mathrm{PRF}_k(\cdot)$ as if it is truly random. PRFs are typically realized via a full-fledged cryptographic hash functions such as SHA-512 in HMAC mode [31], or with a block cipher like AES in a MAC mode of operation.

Keyed packet-hashing function. All our protocols require a hash computation on the invariant contents of every sent packet;[2] and all subsequent processing of the packet relies only on this hash value. Our packet-hashing function will always be keyed with an ephemeral interval key k_u, which is used only for the duration of single interval consisting of T packets (typically $T = 10^7$ and an interval lasts for about 100ms). Once the interval ends, k_u no longer needs to be kept secret (because Bob has already received the packets sent during the interval). In Sections III-D, III-F we consider instantiating the keyed-packet hashing function model with both a PRF and a 4-wise independent hash function [14]. In either case, our keyed packet-hashing function should be (a) fast enough to keep up with multi-Gbit/sec packet streams, (b) remain secure for the duration of an interval, *i.e.,* after about $T = 10^7$ applications and/or for about 100ms.

Message Authentication Code (MAC) is a basic cryptographic primitive that can be realized using a PRF: using a shared key k, for a message m, one party will send $(m, \mathrm{PRF}_k(m))$ and the other party can verify that a pair (m, t) satisfies $t = \mathrm{PRF}_k(m)$. The value t cannot be feasibly forged by an adversary that does not know k. We use the notation $[m]_k$ to denote $(m, \mathrm{PRF}_k(m))$, a message m MAC'd with key k.

B. Description of the secure sketch protocol

Our protocol works in intervals. We assume Alice and Bob share a secret master keys (k_1, k_2), and derive an ephemeral interval key k_u for each interval u. Pairwise master keys are be derived via *e.g.,* authenticated Diffie-Hellman key exchange (as used in TLS/SSL [19]) or some other out-of-band secure channel. Interval keys are computed using a pseudorandom function PRF keyed with master key k_2 (*i.e.,* let $k_u = \mathrm{PRF}_{k_2}(u)$), and Appendix A shows how to synchronize intervals. Alice and Bob also store a sketch, or an array of N counters, each of which can count from $[-\kappa, +\kappa]$. Within interval u, our secure sketch protocol has four phases:

(Sketch.) Alice runs a sketching algorithm, using a keyed packet-hashing function $h_{k_u}(\cdot)$ keyed with secret interval key k_u to incrementally compute a sketch $\mathbf{w}_\mathrm{A}$ of the set of packet she sends during the interval. $h_{k_u}(\cdot)$ may be a 4-wise independent hash function, or a PRF (see Section III-D). For each packet d that Alice sends, she (a) computes $h_{k_u}(d)$ to obtain a (pseudorandom or 4-wise independent) pair of numbers (i, b) where $i \in [N]$ and $b \in \{-1, +1\}$, and (b) adds b to the i^{th} counter in the sketch $\mathbf{w}_\mathrm{A}$. Bob similarly uses $h_{k_u}(\cdot)$ to compute a sketch $\mathbf{w}_\mathrm{B}$ of the set of packets he receives.

(Interval End.) After sending the T^{th} packet in the interval, Alice sends an 'Interval End' message to Bob containing her sketch $\mathbf{w}_\mathrm{A}$ and the next interval number $u+1$, which she signs with a message authentication code (MAC) keyed with k_1. She then refreshes her sketch (*i.e.,* sets $\mathbf{w}_\mathrm{B} = 0$) and computes the next interval key using a pseudorandom function PRF keyed with master key k_2 (*i.e.,* lets $k_{u+1} = \mathrm{PRF}_{k_2}(u+1)$).

(Report.) Upon receiving the 'Interval End' message and verifying the correctness of its MAC, Bob computes the *difference sketch* $\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}$. Bob then sends a 'Report' message to Alice, containing the difference sketch and the current interval number u, which he signs with a MAC keyed with k_2. Bob then refreshes his sketch and computes the interval key for the next interval $u+1$.

[2] Whenever the packet-hashing function is applied to a packet, the non-invariant fields of the packet header are discarded from the input. In the case of IPv4, this means excluding the ToS, TTL and IP checksum (see [21, SectionII.A]).

(Security Check.) Upon verifying the MAC on the 'Report' message, Alice uses the difference sketch $\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}$ to raise an alarm if

$$\|\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}\|_2^2 > \Gamma = 2\frac{\alpha\beta T}{\beta + \alpha}$$

or if the 'Report' message is missing or has an invalid MAC. We call $\|\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}\|_2^2$ the *estimator* and Γ the *threshold.*

Our protocol has a few attractive properties. First, we need only transmit two control messages ('Interval End', and 'Report') per interval; we require no other packet modifications. Second, the 'Security Check' phase can be computed offline. Thirdly, Alice and Bob need only store single sketch at any given time; at the end of each interval, Alice and Bob immediately transmit their sketches as control messages, refresh their sketches, and begin monitoring a new interval. Finally, our protocol has low storage and communication requirement; in Section III-D we show that the sketch is not much larger than an ordinary counter, having size $O(\log T)$ where T is the number of packets sent during the interval. In fact, the number of counters N in the sketch depends only on security parameters α, β, δ but not on T, while the size of each counter $\log \kappa$ is $O(\log T)$.

A performance optimization. To reduce resource consumption, a router can 'turn off' the secure sketching protocol during certain intervals. However, to prevent an adversary from exploiting this to bias monitoring results (*e.g.,* by selectively dropping packets when the protocol is 'off', and behaving itself while the protocol is 'on'), intervals when the protocol is 'off' must be indistinguishable from intervals when the protocol is 'on'. Fortunately, the only indication that the protocol is 'on' are the two control messages ('Interval End' and 'Report'). Thus, while Alice and Bob need not compute hashes over packet contents or to maintain sketches in an 'off' interval, we still require (a) Alice to count the number of packets she sends to Bob and send a dummy 'Interval End' message each time the counter reaches T, and (b) Bob to respond with a dummy 'Report' packet. To make the dummy control messages indistinguishable from real control messages, we will also require (c) that *all* information fields in the control messages sent by the protocol are encrypted and padded to a fixed length (and subsequently authenticated). Selection of 'on' intervals should also be random, to prevent an adversary from selectively attacking the 'off' intervals by using side-channel information (*e.g.,* observing if the sender switches to a new path) to distinguish between 'on' or 'off' intervals.

C. Security analysis

We now prove the security of our protocol by explaining the connection between PQM, and sketches that can be used to estimate the second moment of a set of items. To do this, we explain why the process of sketching can be viewed as a linear transformation that preserves second moment, and then show why this suffices to satisfy our security definition.

Packet streams as vectors. We can think of the stream of packets sent by Alice during a given interval as vector. Let U be the "universe" of all possible packets sent by Alice (*e.g.,* if packets are 1500 bytes long then $|U| \approx 2^{1500\cdot 8}$). Let the characteristic vector of a stream of packets be a $|U|$-dimensional vector that has integer c in the position corresponding to packet x if packet x appears in the stream c times (*e.g.,* for the stream 1,2,4,2,2 of packets drawn from universe $U = [4]$, the characteristic vector is [1 3 0 1].) Naturally, a characteristic vector is too long (*e.g.,* $2^{1500\cdot 8}$) to be represented explicitly; we will use it only as a tool for analyzing the security of our protocol. Let $\mathbf{v}_\mathrm{A}$ be the characteristic vector for the stream of packets sent by Alice during a given interval, and analogously $\mathbf{v}_\mathrm{B}$ for the stream of packets received by Bob.

Sketching as matrix multiplication. The sketch we use in our protocol was first proposed by [15]. While Section III-B presented sketching as an incremental streaming process applied individually to each packet, we now view sketching as a single linear transformation applied to a characteristic vector. Specifically, for a sketch $\mathbf{w}$ computed from the packet stream represented by characteristic vector $\mathbf{v}$ per the description in Section III-B, we can write

$$\mathbf{w} = R\mathbf{v} \tag{1}$$

where R is a $N \times |U|$ matrix that is completely determined by the keyed packet-hashing function $h_{k_u}(\cdot)$; specifically, for every packet $d \in U$, the d^{th} column of matrix R has its i^{th} row equal to b, where $(i, b) = h_{k_u}(d)$ for $i \in [N]$ and $b \in \{-1, +1\}$, and all its other rows are zero.

A sufficiently large sketch $\mathbf{w}$ can (ε, δ)-estimate the second moment of $\mathbf{v}$. That is, for a sketch with N counters that counts from $[-\kappa, \kappa]$, where the packet hashing function is either a PRF or a 4-wise independent hash function that is chosen independently of the characteristic vector $\mathbf{v}$, then

$$(1-\varepsilon)\|\mathbf{v}\|_2^2 < \|\mathbf{w}\|_2^2 = \|R\mathbf{v}\|_2^2 < (1+\varepsilon)\|\mathbf{v}\|_2^2 \tag{2}$$

with probability $1 - \delta$ as long as N and κ are sufficiently large [15], [47]. In Section III-D, we show how to size N and κ so that (2) holds.

PQM as second moment estimation. We are now ready to see how PQM can be derived from second moment estimation. Observe that characteristic vector $\mathbf{v}_\mathrm{A} - \mathbf{v}_\mathrm{B}$ can be decomposed into two vectors $\mathbf{d} - \mathbf{a}$. Vector $\mathbf{d}$ is the characteristic vector of packets *dropped* on the path from Alice to Bob, and contains the non-negative components of $\mathbf{v}_B - \mathbf{v}_A$. Vector $\mathbf{a}$ is the characteristic vector of packets *added* on the path from Alice to Bob, and corresponds to the non-positive components of $\mathbf{v}_B - \mathbf{v}_A$. (Note that a packet modification amounts to a dropped packet *plus* an added packet.) Also notice that the non-zero coordinates of $\mathbf{d}$ and $\mathbf{a}$ are disjoint. Now let D be the number of packets dropped on the path from Alice to Bob during the interval, and let A be the number of packets added during the interval. Thus, we have the following simple and very useful identity:

$$||\mathbf{v}_\mathrm{A} - \mathbf{v}_\mathrm{B}||_p^p = ||\mathbf{d}||_p^p + ||\mathbf{a}||_p^p = D + ||\mathbf{a}||_p^p \geq D + A \tag{3}$$

The identity tells us that, for any integer $p \geq 1$, the p^{th} moment of the characteristic vector $\mathbf{v}_\mathrm{A} - \mathbf{v}_\mathrm{B}$ overestimates the number of packets that are added and dropped during a given interval. The first equality in (3) follows because the non-zero coordinates of $\mathbf{d}$ and $\mathbf{a}$ are disjoint. The second equality

follows because every packet that Alice send is unique, so that that $\mathbf{d}$ is a $\{0,1\}$-vector. Finally, the last inequality follows because $\mathbf{a}$ is an integer vector, so that for any $p \geq 1$, it follows that $\|\mathbf{a}\|_2^2 \geq |\mathbf{a}|_1 = A$ with equality when $p = 1$.

We are now ready to prove security. Set $\varepsilon = \frac{\beta-\alpha}{\beta+\alpha}$.

1) Few false positives. To satisfy this condition we need to consider the *benign case* in which at most $\|\mathbf{d}\|_2^2 = D \leq \alpha T$ packets are dropped during the interval, and there is no adversary Eve on the path. Because Eve is absent, we can assume that no packets are added, so that $\|\mathbf{a}\|_2^2 = 0$. Equation 3 gives

$$\|\mathbf{v}_\mathrm{A} - \mathbf{v}_\mathrm{B}\|_2^2 = \|\mathbf{a}\|_2^2 + \|\mathbf{d}\|_2^2 \leq \alpha T \tag{4}$$

Next, we observe that in the benign case, the keyed-packet hashing function $h_{k_u}(\cdot)$ is chosen independently of $\mathbf{v}_\mathrm{A}$ and $\mathbf{v}_\mathrm{B}$. Thus, for an appropriately-sized sketch (where the number of counters N and the size of each counter κ is sufficiently large), we know that equation (2) holds with probability $1-\delta$. Thus we have:

$$\begin{aligned}
\|\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}\|_2^2 &= \|R\mathbf{v}_\mathrm{A} - R\mathbf{v}_\mathrm{B}\|_2^2 && \text{(From (1))}\\
&= \|R(\mathbf{v}_\mathrm{A} - \mathbf{v}_\mathrm{B})\|_2^2 \\
&\leq (1+\varepsilon)\|\mathbf{v}_\mathrm{A} - \mathbf{v}_\mathrm{B}\|_2^2 && \text{(From (2))}\\
&\leq (1+\varepsilon)\alpha T && \text{(From (4))}\\
&= 2\tfrac{\alpha\beta T}{\beta+\alpha} T = \Gamma && \text{(Since } \varepsilon = \tfrac{\beta-\alpha}{\beta+\alpha}\text{)}
\end{aligned}$$

Thus, Alice will not raise an alarm in the benign case, with probability $1-\delta$, if N, κ are sufficiently large.

2) Few false negatives. To satisfy this condition, we need to consider the malicious case. First observe that in this case, Eve cannot forge the 'Interval End' or 'Report' control messages, since the control messages are authenticated using a secure MAC (and dropping the report will only cause Alice to raise an alarm). Thus, we can suppose that Alice correctly receives the difference sketch $\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}$ from the 'Report' message. In the malicious case, Eve drops $D \geq \beta T$ packets, and adds an arbitrary number of (potentially non-unique) packets $A \geq 0$. In this case, (3) tell us that

$$\|\mathbf{v}_\mathrm{A} - \mathbf{v}_\mathrm{B}\|_2^2 = \|\mathbf{a}\|_2^2 + \|\mathbf{d}\|_2^2 \geq \beta T \tag{5}$$

Now, observe that (b) k_u is chosen independently of the packets sent by Alice $\mathbf{v}_\mathrm{A}$, and (b) k_u is kept secret from Eve until packets reach Bob at the end of the interval. k_u is therefore independent or the packets received by Bob $\mathbf{v}_\mathrm{B}$ (some of which may have been sent by Eve). We can therefore assume equation (2) holds with probability $1-\delta$ for a sketch with an appropriately-sized N and κ. Thus

$$\begin{aligned}
\|\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}\|_2^2 &= \|R(\mathbf{v}_\mathrm{A} - \mathbf{v}_\mathrm{B})\|_2^2 && \text{(From (1))}\\
&\geq (1-\varepsilon)\|\mathbf{v}_\mathrm{A} - \mathbf{v}_\mathrm{B}\|_2^2 && \text{(From (2))}\\
&\geq (1-\varepsilon)\beta T && \text{(From (5))}\\
&= \tfrac{2\beta\alpha}{\beta+\alpha} T \;=\; \Gamma && \text{(Since } \varepsilon = \tfrac{\beta-\alpha}{\beta+\alpha}\text{)}
\end{aligned}$$

with probability $1-\delta$ Alice will thus alarm in the malicious case, with probability $1-\delta$, if N, κ are sufficiently large.

This concludes our argument, since Alice can use the decision threshold Γ to decide between the benign and malicious cases with probability $1-\delta$, as long as the sketch is sized appropriately.

D. Sizing the sketch

We determine the parameters N (number of counters in the sketch) and κ (the size of each counter) when the keyed packet-hashing function $h_k(\cdot)$ is (a) 4-wise independent hash function, and (b) it is a PRF. In both cases we show that N depends only on α, β, δ, while $\kappa = O(\log T)$.

Sizing each counter. A counter holds integers in $[-\kappa, +\kappa]$.

$$\log_2(2\kappa) = 1 + \tfrac{1}{2}\log_2\left(4\tfrac{T}{N}\ln(\tfrac{200N}{\delta})\right) \quad \text{bits / counter} \tag{6}$$

suffices to ensure that the probability that each counter overflows is at most $\frac{\delta}{N}\frac{1}{100}$; it follows from the union bound that, with probability $1-\delta/100$, no counter will overflow. To see how we obtained (6), observe that if X_i is a random variable that equals 1 with probability $\frac{1}{2N}$, -1 with probability $\frac{1}{2N}$, and 0 otherwise, then the count in each bin is the random variable $X = \sum_{i=1}^{T} X_i$. Then, adapting the Chernoff bound that appears in [33], we have that

$$\Pr[\,|X| \;\geq\; \kappa\,] \;\leq\; 2\exp(-\tfrac{\kappa^2}{4T\mathrm{VAR}[X_i]}) \;\leq\; \tfrac{\delta}{100N}$$

Finally, we get (6) since $\mathrm{VAR}[X_i] = 1/N$.

Sizing the number of counters N. While N depends only on α, β, δ, its exact value depends on the instantiation of the keyed-packet hashing function. We first consider its instantiation with a 4-wise independent hash, and then with a PRF.

Packet-hashing with 4-wise independent hashes. A 4-wise independent hash is a function $h : \{0,1\}^{|k_u|} \times \{0,1\}^* \to \{0,1\}^n$ that guarantees that for any four distinct inputs x_1, x_2, x_3, x_4 and (possibly non-distinct) outputs y_1, y_2, y_3, y_4, then

$$\Pr[h_{k_u}(x_i) = y, \forall i = 1...4] = \left(\tfrac{1}{2^n}\right)^4 \tag{7}$$

where the probability is over the choice of k_u used to key h. A 4-wise independent hash can be realized using polynomials of degree 3 [14]. For example, to compute $h_{k_u}(x)$ set key $k_u = (a_0, a_1, a_2, a_3)$ and output $a_3x^3 + a_2x^2 + a_1x + a_0$; this can be done in three multiplications with Horner's rule.

Thorup and Zhang [47] showed that 4-wise independent hashing suffices to realize second moment estimation using [15]'s sketching algorithm:

Theorem III.1 (From [47])**.** *If we construct sketch* $\mathbf{w}$ *using two 4-wise independent packet hashing functions* $h : U \to [N]$ *and* $b : U \to \{-1, 1\}$ *then*

$$\mathbb{E}[\|\mathbf{w}\|_2^2] = \|\mathbf{v}\|_2^2 \tag{8}$$

$$\mathbb{VAR}[\|\mathbf{w}\|_2^2] = \tfrac{2}{N}(\|\mathbf{v}\|_2^4 - \|\mathbf{v}\|_4^4) \tag{9}$$

And $\|\mathbf{w}\|_2^2$ (ε, δ)*-estimates* $\|\mathbf{v}\|_2^2$ *as long as* $N > \frac{2}{\varepsilon^2\delta}$.

Since $\varepsilon = \frac{\beta-\alpha}{\beta+\alpha}$, it suffices to take

$$N \geq \tfrac{2}{\varepsilon^2 0.99\delta} = \tfrac{2.02}{\delta}\tfrac{(\beta+\alpha)^2}{(\beta-\alpha)^2} \tag{10}$$

counters in our sketch. What is remarkable about this result is the fact that N, the number of counters in our sketch, is completely independent of the number of packets T sent during the interval! We shall see in Section III-E that this means that our sketches can monitor a large number of packets using a very limited amount of space.

Packet-hashing with a PRF. Since every PRF is also indistinguishable from a 4-wise independent hash function, choosing N as in (7) also suffices when the keyed packet-hashing function is instantiated with a PRF. However, we can do better when the packet hashing function is a PRF. Specifically, can take $N = O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)$ by exploiting a few special properties of our PQM setting, including (a) the assumption that Alice sends unique packets, and (b) the fact that we only care about deciding whether $\|\mathbf{v}\|_2^2$ lies above or below a threshold, rather than getting an accurate estimate of $\|\mathbf{v}\|_2^2$.

The following theorem, which may be of independent interest, supposes that packet hashing uses two independent random functions: one to chose $i \in [N]$ and another to choose $b \in \{-1, 1\}$. We can therefore think of the sketching matrix R as chosen *uniformly at random* from set of matrices $\mathcal{S}$, where $\mathcal{S}$ is the set of $N \times |U|$ matrices where each column contains a single ± 1 entry in one row, and zeros in all other rows. We prove the following in Appendix D of [25]:

Theorem III.2. *For any vector $\mathbf{v} \in \mathbb{Z}^U$, choose the $N \times |U|$ sketching matrix R uniformly at random from $\mathcal{S}$. If $\mathbf{w} = R\mathbf{v}$, then for all $\varepsilon \in [0, 1)$ and η such that*

$$\left(\frac{1-\eta}{1+\eta}\right)^2 = \max\left(\frac{1+\frac{\varepsilon}{2}}{1+\varepsilon}\ ,\ \frac{1-\frac{3\varepsilon}{4}}{1-\frac{\varepsilon}{2}}\right) \tag{11}$$

the following two items occur with probability at least $1 - \delta$:

1) *If $\mathbf{v} \in \{-1, 0, 1\}^U$, and $\|\mathbf{v}\|_2^2 \le q$, then $\|\mathbf{w}\|_2^2 < (1 + \varepsilon)q$.*
2) *The number of* non-zero *entries in $\mathbf{v}$ is r, then $\|\mathbf{w}\|_2^2 > (1 - \varepsilon)r$.*

as long as

$$N \ge \tfrac{24}{\varepsilon^2}\ln\tfrac{2}{\delta} \tag{12}$$

$$q, r \ge \tfrac{3N}{\eta^2}\ln\tfrac{4N}{\delta} \tag{13}$$

To apply the theorem into our setting, we plug $\varepsilon = \frac{\beta-\alpha}{\beta+\alpha}$ and δ into (12) to obtain a bound on N, the number of counters in our sketch. Then, we plug ε in (11) to determine η. Finally, we set $q = \alpha T$ and $r = \beta T$ in (13) to obtain a lower bound on T, the minimum number of packets sent in an interval. (This lower bound on T is awkward, and we believe that it is an artifact of our proof technique; indeed, numerical experiments in Section III-E suggest that this bound on T is not tight.)

We show why these parameters guarantee security. First, assume that the PRF used for packet hashing is indistinguishable from a random function. Then, using the language of Section III-C, the false positive condition is satisfied because in the benign case we have $\mathbf{v}_\mathrm{A} - \mathbf{v}_\mathrm{B} \in \{0, 1\}^U$ (since packets may only be dropped, and all packets are distinct) and $\|\mathbf{v}_\mathrm{A} - \mathbf{v}_\mathrm{B}\|_2^2 = D \le \alpha T = q$, so with probability $1 - \delta$, the first item in Theorem III.2 gives

$$\|\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}\|_2^2 \le (1+\varepsilon)\|\mathbf{v}_\mathrm{A} - \mathbf{v}_\mathrm{B}\|_2^2 \le (1+\varepsilon)\alpha T = \tfrac{2\alpha\beta}{\alpha+\beta}T = \Gamma$$

The false negative condition is satisfied because in the malicious case, we have $\|\mathbf{v}_B - \mathbf{v}_A\|_2^2 = \|\mathbf{d}\|_2^2 + \|\mathbf{a}\|_2^2$ where $\|\mathbf{d}\|_2^2$ is a $\{0, 1\}$-vector (since all dropped packets are distinct) that has at least $r = \beta T$ non-zero entries. So, with probability $1 - \delta$, the second item in Theorem III.2 gives

$$\|\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}\|_2^2 \ge (1-\varepsilon)\|\mathbf{v}_\mathrm{A} - \mathbf{v}_\mathrm{B}\|_2^2 \ge (1-\varepsilon)r \ge (1-\varepsilon)\beta T = \tfrac{2\alpha\beta}{\alpha+\beta}T = \Gamma$$

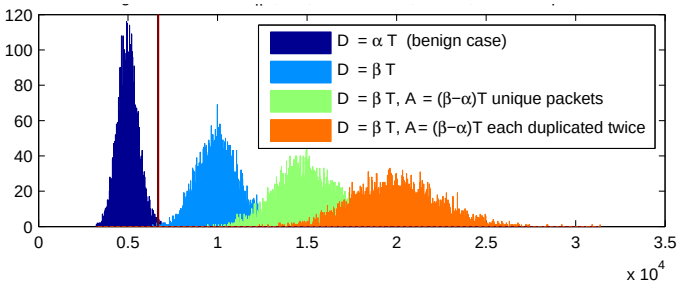

Fig. 1. Distribution of estimator $\|\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}\|_2$ using packet-hashing with a PRF and with $N = 300$, $T = 10^6$, $\beta = 2\alpha = 1\%$ and threshold $\Gamma = 6667$, computed via numerical experiments.

E. Some sample parameters and experiments

We now compute the size of our sketch, using both our analytic results and numerical experiments, for the following sample parameters: We suppose the detection threshold is $\beta = 0.01$, the false alarm threshold is $\alpha = \beta/2$ and about $T = 10^7$ packets are sent during an interval. We will require a confidence of $1 - \delta = 99\%$.

Analytic results. We plug these parameters into our analytic results. When 4-wise independent hashing is used with these parameters, equation (10) indicates that we require $N = 1800$ counters, and equation (6) requires 10 bits/counter; the total size of the sketch is therefore 2.25KB. Storage become even smaller when we use a PRF; applying the refined version of Theorem III.2 in Appendix D of our technical report [25] to obtain bounds on N, we find that we can use $N = 300$ counters if there are at least $T_{min} = 1.2 \times 10^{10}$ packets in the interval, for a total sketch size of only 375B!

Numerical experiments. We preformed a number of numerical experiments of the case where keyed-packet hashing function is instantiated with PRF. In each numerical experiment, we operate on synthetic traffic, where we model every distinct packet sent during the interval with fresh pair of uniformly-independent random numbers (i, b) where $i \in [N]$ and $b \in \{-1, 1\}$, and use these numbers to create the difference sketch $\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}$ in the natural way (incrementing the i^{th} counter by b every time that packet appeared in the stream); the final output of the numerical experiment is the estimator $\|\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}\|_2^2$. For a given stream of packets, we can repeat this process multiple times to obtain the distribution of the estimator $\|\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}\|_2^2$.

Figure 1. Figure 1 shows the resulting distribution of the $\|\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}\|_2^2$ for a number of cases. From left to right, we have: The benign case where $D = \alpha T$ (here we want the estimator to be below the threshold Γ so that Alice does not raise an alarm), and three cases where $D = \beta T$ (here we want Alice to raise an alarm): a case where Eve does not add any packets, a case where Eve adds $A = (\beta - \alpha)T$ distinct packets, and a case where Eve adds a total of $A = (\beta - \alpha)T$ packets where each packet is duplicated twice. The figure reveals that the threshold $\Gamma = 2\frac{\alpha\beta T}{\alpha+\beta}$ clearly distinguishes between cases where $D = \beta T$ and the benign cases where $D = \alpha T$. Moreover, when Eve adds packets to the link, she only increases the probability that Alice raises an alarm, as predicted by equation (3). Figure 1 also suggests that taking $N = 300$ suffices even if we have shorter interval lengths

β/α	N	Sketch Size (in bytes)				
		$T = 10^4$	$T = 10^5$	$T = 10^6$	$T = 10^7$	$T = 10^8$
2	128	128B	144B	176B	208B	208B
4	64	64B	88B	88B	104B	104B
8	32	36B	48B	48B	52B	52B
16	32	36B	48B	48B	52B	52B
32	32	36B	48B	48B	52B	52B
64	32	36B	48B	48B	52B	52B

Fig. 2. Minimum number of counters N in the sketch when N is taken as a power of 2, computed via numerical experiments where keyed packet hashing is performed using a PRF. The number of bits per counter is obtained from (6). We fix $\beta = \delta = 1\%$.

of $T = 10^6$, suggesting that the awkward bound of T in Theorem III.2 is an artifact of our proof technique.

Figure 2. We preformed more numerical experiments to determine N, the number of bins in the sketch, for a given choice of α, β, δ. Recall from Section III-C that threshold $\Gamma = 2\frac{\alpha\beta T}{\alpha+\beta}$ must be able to distinguish between $\|\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}\|_2^2$ in the benign case vs. the malicious case, with probability $1 - \delta$. We therefore need to compare the extremes of the benign and the malicious cases. In the malicious case, the expected value of $\|\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}\|_2^2$ is minimized when the number of dropped packets $D = \beta T$ and the number of added packets is $A = 0$. Meanwhile, in the benign case, the expected value of $\|\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}\|_2^2$ is maximized when $D = \alpha T$. (This follows from equations (8) and (3) and is confirmed by Figure 1.) We therefore preform numerical experiments (as in Figure 1) to obtain the distribution of $\|\mathbf{w}_\mathrm{A} - \mathbf{w}_\mathrm{B}\|_2^2$ in these two extreme cases. We do this for various values of N that are powers of 2,[3]and then find the smallest value of N for which threshold Γ is able to distinguish between the two extreme cases with probability at least $1 - \delta$.

Our results are presented in Figure 2. We see that N varies with the ratio β/α. Meanwhile, as long as there are $T > 1/\alpha$ packets/interval, the choice of T does not impact the value of N; for a given β/α ratio, the minimum choice of N as power of 2 was the same for any value of T ranging from 10^4 to 10^8, which suggests that the lower bound on T in Theorem III.2 is not tight. Moreover, Figure 2 also confirms that since total sketch size grows logarithmically with T (per (6)), the total storage required by our sketching scheme remain modest.

F. Implementation issues and fast packet hashing.

As we see it, two main barriers stand in the way of the deployment of our secure sketch PQM protocol.

The first is establishing shared (symmetric) cryptographic keys between Alice and Bob; unfortunately, this overhead is unavoidable, since our lower bound in [24] establishes that keys are *necessary* for any secure PQM scheme satisfying our definition. However, shared keys can always be established in an enterprise setting, where *e.g.,* a central office (Alice) wants to monitor its connection to a branch office or datacenter (Bob) over the public Internet. Moreover, if a public-key infrastructure is in place (either localized within the enterprise, or just using the SSL/TLS PKI) a key-exchange protocol [19] could always be used to establish the shared keys.

[3]We take N to be a power of two because this makes packet hashing more convenient. That is, if $N = 2^q$ and we use PRF that produces q pseudorandom bits, then the binary representation of these q bits is a uniformly chosen element of $[N]$; if N is not a power of 2, a more complicated mapping is required to uniformly choose an element of $[N]$, so we avoid this.

The second barrier is computational overhead. The most expensive part of our sketching protocol is the computation of the per-packet hash, which must be computed over the contents of the entire packet.[4] However, we now discuss several hashing techniques that can be used to speed this computation up, and estimate its computational overhead by citing recent results on the implementation of fash hash functions.

For speedy packet hashing, we suggest first hashing packets using ε_g-almost universal hash function to obtain a short n_1 bit string, and then applying a fast PRF.

ε_g-almost universal hash function is a keyed hash function $g : \{0,1\}^{|k_u|} \times \{0,1\}^* \to \{0,1\}^n$ that maps variable-length inputs to n-bit outputs, and guarantees that for any pair of distinct inputs x, x' that

$$\Pr\left[g_{k_u}(x) = g_{k_u}(x')\right] \leq \varepsilon_g \tag{14}$$

where the probability is over the choice of k_u keying g. The value of ε_g depends on the parameters of the hash, and these hash functions have many practical realizations that are significantly faster than PRFs and 4-wise independent functions [12].

We can speed up packet-hashing by first hashing each packet using a fast ε_g-almost universal hash function (converting a packet of length ℓ bits to a string of length n_1 bits), and then hashing the resulting n_1-bit string using a PRF or 4-wise independent hash (to obtain the pair of values (i, b) for $i \in [N]$ and $b \in \{-1, 1\}$ used for sketching). Appendix B proves it suffices to take:

$$\varepsilon_g < \frac{\delta}{10^3 T}\frac{\beta - \alpha}{\alpha + \beta} \tag{15}$$

Using GHASH [35] as our ε_g-almost 2-wise independent hash function, and suppose GHASH produces outputs of length n_1, and each packet is at most 1500B long, and block lengths are m, where m is taken as a power of two. Since $\varepsilon_g = \frac{1500\cdot 8}{m} 2^{-n_1}$, for $T = 10^7$ packets/interval, and $\delta = \beta = 2\alpha = 1\%$, applying (15) we find that it suffices to choose GHASH with $n_1 = m = 64$ bits. Smaller block sizes lead to faster implementations, and this choice of m and n_1 is quite short! (Typically GHASH has block lengths of $m = 128$ bits [35].) Because n_1 is shorter than standard hashing block lengths (*i.e.,* 128, 256, or 512 bits), we just need one invocation of a fixed-input-length PRF (or 4-wise independent hash) to hash the final n_1-bit GHASH output.

Our next task is to determine how the n_1-bit string be hashed. Is it faster to use a PRF or a 4-wise independent hash? Our answer is very counterintuitive — we find that a PRF is actually faster in practice. This is quite surprising, given that a PRF provides a strictly stronger theoretical guarantees that a 4-wise independent hash function (since every PRF is indistinguishable from a 4-wise independent hash function). To explain this, we observe that 4-wise independent hash functions are typically based on constructions with rigorous proofs of correctness (*e.g.,* polynomials of degree 3 [14]). Meanwhile, practical PRFs come with only *heuristic* guarantees (*e.g.,*

[4]Hashing just the packet header won't work, because then the protocol could not detect an adversary that tampers with the payload of the packet but keeps the header intact.

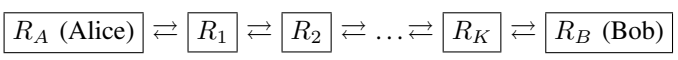

Fig. 3. A path from Alice to Bob via K intermediate nodes.

GHASH-AES [35] is a PRF under the heuristic assumption that AES is a fixed-input-length PRF). Moreover, significantly more research effort has been expended on developing fast implementations of PRFs in software and hardware.

Sample implementation numbers. Since the task of evaluating the performance of a hash function on the different platforms that be used in high-speed Internet routers (software, FPGA, ASICS) is a entire research project in itself, here we only mention some performance numbers from other published work. We follow the literature [35] by assuming that AES is a fixed-input length PRF. In 2007, Krovetz and Dai found that VMAC instantiated with AES allows for hashing in software at a speeds of between 1-10 cycles per message byte [32]. In 2004, the designers of GHASH found that it could process 127 bits per clock cycle in hardware and 2666 bits per clock cycle in software, assuming 1500 byte packets and 128-bit block lengths [35]. There have been various efforts to improve the performance of GHASH. In 2014, for example, a new FGPA implementation of GHASH claims a throughput of 43.32Gbps [10]. (Note that [10] used 128-bit blocks lengths; recall that we can use even short block lengths of 64-bits.)

Before we conclude, we note that the security of our PRF need not be especially high. Our PQM adversary is presented with an online problem: Eve must break the security of the PRF only during the short time interval before the ephemeral interval key k_u is refreshed. Thus, while a full-fledged block-cipher (*e.g.,* AES) could be used to realize the PRF, performance could further be improved by the replacing AES with a weaker block cipher like DES, or by reducing the number of rounds of AES [18]; an adversary would still require enormous resources to break the PRF's security within the short time limit (≈100ms) imposed by our online setting.

IV. Threat model for Failure localization (FL)

We now show howw secure sketch PQM can be composed to create a secure failure localization (FL) protocol. We first discuss a security definition for FL (Sections IV-A-IV-B), and then present our FL protocol (Section V).

A. Security definition for failure localization (FL)

In a failure localization (FL) protocol, Alice learns if the packets she sent to a trusted destination Bob arrived correctly; if they did not, Alice learns at least one link along the path where the failures occurred.

We work in a model where Alice knows the identities of all the nodes on the path to Bob, and all traffic travels on symmetric paths as in Figure 3 (*i.e.,* intermediate nodes $R_1, ..., R_K$, 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 let K be the number of nodes on the path between Alice and Bob. Messages traveling towards Alice are going *upstream*, and messages traveling towards Bob are going *downstream*. As in Section II-A, we fix a set of T consecutive packets sent by Alice to be an *interval*. A *failure* is any instance where a packet that was sent by Alice during the interval fails to arrive unmodified at Bob (before the last packet of interval arrives at Bob).

Adversary Eve can occupy any subset of nodes $R_1, ..., R_K$ on the network path between Alice and Bob; Eve will remain at those nodes for the duration of the interval. Eve 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. Because packet loss occurs naturally in the network layer, FL protocols should also be able to deal with packet failures that do *not* results from Eve's actions. Therefore, we model congestion by supposing that each link can independently drop each packet transmitted over that link with uniform probability $\rho > 0$.

An FL protocol allows Alice to detect (1) whether the number of failures during the interval exceeds a certain fraction of total packets transmitted, and (2) if it does, Alice can localize the failures to a link on the path. At the end of each interval of an FL protocol, Alice either (a) decides not to alarm and outputs $\surd$, or (b) or raises an alarms and outputs a link ℓ to which she localized the failures.

Definition IV.1 ((α, β, δ)-security for FL)**.** *Given parameters $0 < \alpha < \beta < 1$ and $0 < \delta < 1$, an FL protocol is (α, β, δ)-secure if, letting T be the number of packets sent during the interval:*

1) *(*Secure localization*). Consider the* malicious case*, where the adversary Eve can drop, modify, delay, or add packets. We require that if more that βT failures occur, then Alice raises alarm for a link ℓ that is adjacent to a node occupied by Eve, or a link ℓ whose failure rate exceeds $\frac{\alpha}{K+1}$, with probability $1 - \delta$.*
2) *(*Few false positives*). In the* benign case*, where no intermediate node behaves adversarially (*i.e., *no packets are added or modified on the path, but packets may be reordered or dropped due to congestion) 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 $1 - \delta$.*

As in Section II-A, we assume that the T packets sent during an interval are distinct.

B. Properties of our FL security definition

We discuss a few properties of our security definition.

Localizing links, not nodes. It is well known (see *e.g.,* [24]) that an FL protocol can only pinpoint a *link* where a failure occurred, rather than the *node* responsible for the failure.

Benign and malicious failures. Our definition requires 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.

Modeling congestion. Our definition accounts for messages that are dropped for benign reasons like congestion. If we had

not included this in our model, then our model would have required *Eve* to cause *every* packet delivery failure. This is problematic because such a definition would admit protocols that attempt to only localize a *single* packet failure (rather than the overall packet loss rate) since this link would necessarily be adjacent to Eve. Indeed, [9] has fallen into this trap, by implementing FL via a binary search through the path, where a single step of the binary search is initiated each time a packet is dropped; this can be confounded by congestion-related packet loss that causes the binary search algorithm to search for Eve in the wrong part of the path.

Movements of the adversary. Our model does not allow Eve to move from node to node in a single interval for the following reasons: First, when Eve models a Internet service provider (ISP) that tries to bias the results of FL protocol for business reasons, she may only occupy nodes owned by her ISP. Second, when Eve is an external attacker 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 key refresh as a re-start of the security game. Third, "movements" to a new router could be infrequent, since an external attacker might need a different strategy each time it compromises a router owned by a different entity.

V. From Secure Sketch PQM to FL

We show how to compose the secure sketch PQM protocol of Section III-B to obtain a efficient FL protocol with about $O(K^2 \log T + n)$ storage overhead at each node and only two control messages. Our composition of secure sketch PQM to statistical FL will have Alice run K simultaneous secure sketch PQM protocols with each of the intermediate nodes in Figure 3, and use the statistics from each protocol to infer behavior at each link. We first describe the protocol (Section V-A) and then prove its security (Section V-B).

A. Description of the secure sketch FL protocol

As usual, the protocol works in intervals. Every node R_i shares pairwise symmetric master keys k_i, k_i' with Alice. In interval u, the protocol proceeds as follows:

(Sketch.) Using k_i', each intermediate node runs a secure sketch PQM protocol with Alice, so that Alice will keep a sketch $\mathbf{w}_i^A$ for every $i \in [K]$ and every other node R_i will keep a single sketch $\mathbf{w}_i$. Sketching is accomplished as described in the (Sketch) phase of the PQM protocol of Section III-B.

(Interval End.) At the end of interval u, Alice sends a single control 'Interval End' control message formatted as an *onion report*. Using notation $[m]_k$ to denote message m authenticate by a MAC with k, the 'Interval End' message contains a series of nested MAC'd messages as follows:

$$q = [(u, 1, \mathbf{w}_1^A)[(u, 2, \mathbf{w}_2^A)...[(u, B, \mathbf{w}_B^A)]_{k_B}...]_{k_2}]_{k_1}$$

The 'Interval End' control message is sent upstream along the path from Alice to Bob. Upon receiving a validly-MAC'd 'Interval End' message, intermediate node R_i (a) extracts the sketch $\mathbf{w}_i^A$, (b) strips off his portion of the message $(u, i, \mathbf{w}_i^A)$ and its associated MAC, (c) passes what remains to R_{i+1}, and (d) finally initializes a local timer. R_i must drop the 'Interval End' message if the MAC is invalid.

(Report.) Bob initiates reporting, by forming an 'Onion Report' message $\theta_B = [u, B, V_B]_{k_B}$ and sending it downstream. Upon receipt of the 'Onion Report', each node R_i appends his own information as $\theta_i = [u, i, V_i, \theta_{i+1}]_{k_i}$ and passes θ_i downstream to R_{i-1} until it eventually reaches Alice. Here, V_i is node R_i's *estimator* computed as

$$V_i = \|\mathbf{w}_i^A - \mathbf{w}_i\|_2^2$$

If a node R_i's local timer expires before it receives θ_i from its upstream neighbor R_{i+1}, then R_i constructs his own Report θ_i as above, but setting $\theta_{i+1} = \perp$ to indicate his upstream neighbor failed to send his report.

(Security Check.) Letting α, β be the false alarm and detection thresholds, when Alice receives the final onion report θ_1, she computes

$$F_\ell = V_i - V_{i+1}$$

for each link $\ell = (i, i+1)$, and outputs ℓ if $\ell = (i, i+1)$ is the upstream-most link where

$$F_\ell > \tfrac{T}{K+1} \tfrac{\beta(2\alpha+\beta)}{\alpha+2\beta} = \Gamma$$

or the onion report θ_{i+1} refers to the wrong interval, is missing, or is invalidly MAC'd. If there is no such link, she outputs $\surd$.

B. Security analysis for secure sketch FL

To prove that this scheme is secure, we need to assume that interval length T is long enough, the sketches are big enough, and the congestion rate ρ is small enough. Our proof also relies on limiting the number of links occupied by Eve to $O(\sqrt{K})$.

Theorem V.1. *The composition of secure sketch PQM described above satisfies (α, β, δ)-statistical security if the congestion rate satisfies $\rho K^2 \le \beta$, Eve occupies*

$$M \le \sqrt{(K+1)(1 - \tfrac{\rho}{\beta} K^2)} \tag{16}$$

links, each interval contains at least $T > \frac{K+1}{\alpha}$ packets, and for each $i \in [K]$, sketches $\mathbf{w}_i, \mathbf{w}_i^A$ have

$$N_i = \tfrac{32}{\delta} \left(i(K+1) \tfrac{2\beta+\alpha}{\beta-\alpha} \right)^2 \tag{17}$$

counters, each of size $1 + \frac{1}{2} \log_2 \left(4 \frac{T}{N} \ln(\frac{200N}{\delta}) \right)$ bits.

The proof is in Appendix C. We remark that, for a given interval of length T, this protocol requires $O(K^2 \log T)$ storage overhead at Bob and each intermediate node, while the storage overhead at Alice is $O(K^3 \log T)$. The communication overhead of the protocol is two control messages of length $O(K^3 \log T)$ each for every T packets sent.

A note on Eve's strategy. Theorem V.1 guarantees that our protocol accommodates all possible strategies by Eve (satisfying the conditions in the theorem), but the reader might wonder what is strategy is best from Eve's perspective. Should she drop all βT packets on just one link, or spread her βT

packet drops over multiple links? It turns out the latter is best for Eve; because the estimator $V_{i-1} - V_i$ is proportional to the number of packets dropped/modified on link $(i-1, i)$, then if fewer packets are dropped on each link, the estimator is smaller and therefore closer to Γ, and it is more likely that Alice will not alarm (and thus fail to catch Eve).

C. Sample parameters for our FL protocol.

We now consider sizing the sketches for the following sample parameters: There are $K = 4$ nodes between Alice and Bob (for a total of 5 links). We suppose the detection threshold is $\beta = 0.01$, the false alarm threshold is $\alpha = \beta/2$ and about $T = 10^6$ packets are sent during an interval. We require a confidence of $1 - \delta = 99\%$. Applying Theorem V.1, we find that the i^{th} node needs a sketch with $N_i = 2 \times 10^6 i^2$ counters and $5 - \log_2 i$ bits per counter. This sketch is quite large, but recall that Theorem V.1 assumes packet hashing uses a 4-wise independent hash function. Per Theorem III.2, however, we already know that sketches can be smaller when a PRF is used for packet hashing. Indeed, our numerical experiments confirm this. We now find that it suffices for node i to use a sketch with $800i^2$ counters, so that the i^{th} node requires a sketch of size $i^2(10 - \log_2 i) \times 100$B.

Numerical experiments. We preformed numerical experiments for the case where keyed-packet hashing function is instantiated with PRF. We assumed that and each node uses a sketch of size $N_i = 800i^2$. Our numerical experiments operates on synthetic traffic, similar to the experiments in Section III-E; this time, however, we need to simulate packet dropping on each individual link, not just on the entire path. To do this, we model every distinct packet sent during the interval *on a given link* $(i-1, i)$ with fresh pair of uniformly-independent random numbers (j, b) where $j \in [N_i]$ and $b \in \{-1, 1\}$, and use the same technique used in Section III-E to compute the estimator $V_{i-1} - V_i$ on link $(i-1, i)$. The distributions of the resulting estimators are shown in Figure 4, along with the alarm threshold Γ. We present two cases.

Benign case (Figure 4 (Top)). We suppose each link drops exactly $\alpha/(K+1)$ packets. As required, the distribution of the estimator as the $V_{i-1} - V_i$ for each link $(i-1, i)$ is below the threshold Γ, and Alice will not raise an alarm.

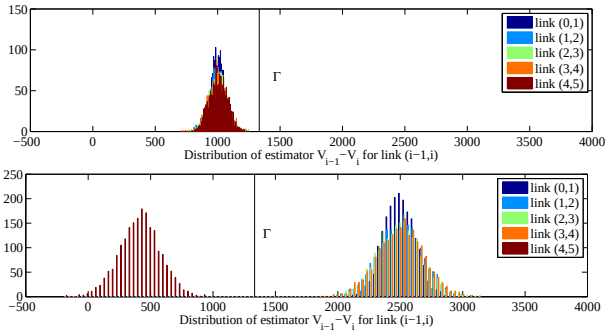

Fig. 4. Distribution of estimator $V_{i-1} - V_i$ for each link $(i-1, i)$ using packet-hashing with a PRF and with $N_i = 800i^2$, $T = 10^6$, $\delta = \beta = 2\alpha = 1\%$ and threshold $\Gamma = 1333$, computed via numerical experiments. $K = 4$ nodes. (Top) The benign case, where each link drops exactly $\alpha/(K+1)$ packets. (Bottom) The malicious case. There is a congestion of rate $\rho = \beta/K^2$ randomly dropping packets on each link, and Eve drops $\beta/K - \rho$ packets on every link except the (4,5) link.

Malicious case (Figure 4 (Bottom)). We suppose that there is a congestion of $\rho = \beta/K^2$ randomly dropping packets on each link (per Theorem V.1), and that Eve drops $\beta/K - \rho$ packets on every link except the (4,5) link. As required, the distribution of the estimators for all links except (4,5) are above the threshold Γ, while the estimator for the (4,5) link is well below the threshold Γ. Thus, Alice will correctly raise an alarm and localize all links except the (4,5) links.

D. Implementation issues.

We discuss a few limitations of our secure sketch FL protocol, beyond those related to the computational overhead of packet hashing per Section III-F. First, the protocol requires all traffic being monitored to flow along the same symmetric path. Load balancers, that could split traffic across different paths, complicate its operation; to deal with this, nodes would need to ensure that each sets of flows that take the same path are monitored by a single instance secure sketch FL protocol. Another limitation is that secure sketch FL requires symmetric keys between all nodes on the path and Alice. We do not deny that this is a heavy overhead, but we do note that such keys are *necessary* for any FL protocol that satisfies our security definition (per our lower bounds in [11]), and note that these keys could be established on the fly using traditional key-exchange schemes [19]. Other works in the FL space have also had to grapple with these two important limitations; see [38] for more ideas on how to address them.

VI. Related work

The literature on path-quality monitoring typically deals only with the benign setting; most approaches either have the destination return a count of the number packets he receives from the source, or are based on active probing (ping, traceroute, [42]–[44] and others). However, both approaches fail to satisfy our security definition. The counter approach is vulnerable to attack by an adversary who hides packet loss by adding new, nonsense packets to the data path. Active probing fails when an adversary preferentially treats probe packets while degrading performance for regular traffic, or when an adversary sends forged reports or acknowledgments to mask packet loss. Even known passive measurement techniques, where normal data packets are marked as probes, either explicitly as in IPPM [44] or implicitly as in Trajectory Sampling [21] and PSAMP [16], are vulnerable to the same attacks as active probing techniques if the adversary can distinguish the probe packets from the non-probe packets (*e.g.,* see [23]).

To deal with the adversarial setting, our protocols that are more closely related to those used for traffic characterization [50]; our protocols less computationally efficient because they *necessarily* require keys and cryptographic hash functions to prevent adversaries from biasing measurement results (we proved this in [24]). On the other hand, the special structure of PQM allows us to prove new analytical bounds, resulting in lower communication and storage than those typically needed in traffic characterization applications (Theorem III.2).

In concurrent work, [36] considered a setting which Alice and Bob are required to sketch adversarially-chosen sets, and

then compute metrics on their sets after exchanging sketches over a secure channel; their model maps directly to our PQM model, where Alice and Bob's sets (*i.e.,* packet streams) may be chosen adversarially, and then sketches are exchanged via an authenticated channel. Our work deals with the fact that streams are chosen adversarially by requiring Alice and Bob to compute their sketches using shared secret keys. However, [36] require that sketching is performed *without* any shared randomness.[5] The main advantage of their approach is the reduced key-management overhead. However, this comes at a significant cost; [36] show that any moment estimation protocol for sets of size T requires at least $\Omega(\sqrt{T})$ storage at Alice and Bob. Thus, these protocols are less efficient than our $O(\log T)$-storage *keyed* sketches.

There is also related work in the security literature. Early work, *e.g.,* [20], [41], focused on providing guarantees for *every packet*, resulting in schemes with very high overhead. Later secure PQM protocols detected when the *packet-loss rate* became too high [8], [37], [46], but our protocol was the first to provide a formal security model and to prove the security of our protocols in this model. Indeed, one of [37]'s PQM approaches is based on a simple counter (and is thus vulnerable to the attack described above), while Listen [46] does not use cryptographic operations (and is thus vulnerable to intermediate nodes that inject false acknowledgments onto the path). Meanwhile, [8] is secure in our model but incurs the extra overhead of encrypting all traffic.

Before we published [11], there were a number of proposals for secure FL [6], [7], [9], [37], [40], [49]. Our protocol is the only one to exploit the storage and communication savings created by sublinear sketching algorithms; moreover, [6], [9], [40] had a number security flaws that we discussed in [11].

After we published [11], [51] considered FL protocols that are more closely related to the sampling-based FL protocols that we presented in [11] (but omitted from this paper), focusing on optimizing the tradeoffs between communication/storage overhead and the protocol's detection rate (*i.e.,* the number of packets in an interval). In another interesting work, [52] use a small trusted computing base and remote code attestation to circumvent our lower bounds from [11], that argued that in any secure FL protocol, the intermediate nodes and Alice and Bob must all share some secret information. Also subsequently to our work, [4], [13], [53] considered FL in a more stringent setting of multiple paths, similar to the SMT framework; in these protocols, a sender must not only localize a faulty node that is dropping packets, but must also find a path through the network that is guaranteed to deliver its packets. Finally, ICING [38] is a cryptographic network primitive that ensures that packets traverse a known path, selected by the source, in an adversarial setting similar to ours.

[5]In [24], we argued that PQM protocols require shared randomness; the existence of [36]'s protocol does not contradict this. If we used [36]'s sketching results in a PQM protocol, we would require shared randomness to cryptographically authenticate the report messages (containing the sketches) sent from Bob to Alice.

VII. Conclusion

We have designed and analyzed an efficient protocol that give accurate estimates of path quality in a challenging environment where adversaries may drop, delay, modify, or inject packets. Our protocols have reasonable overhead, even when compared to previous solutions designed for the *non-adversarial* settings. We combine techniques from sublinear streaming algorithms with simple cryptographic primitives to obtain a protocol that can monitor millions of packets using less than a single kilobyte of storage, and only two small control messages. We have also showed how to compose multiple instances of our PQM protocols to localize the adversary to particular links on a data path. We believe that our secure sketch protocols, and our associated models of their properties, are valuable building blocks for the design of future networks with predictable security and performance guarantees.

Acknowledgments

We thank Eugene Brevdo, Moses Charikar, Nick Feamster, Piotr Indyk, Changhoon Kim, Amir Shpilka, and Yi Wang for useful discussions, and Shai Halevi, Elliott Karpilovsky, Haakon Ringberg, Augustin Soule and the anonymous SIGMETRICS'08 and EUROCRYPT'08 reviewers for comments on drafts of this work. We thank Fabio Manio, Flavio Bonomi, David McGrew, and Syam Appala at Cisco for practical advice, and IPAM at UCLA for hosting a semester on securing cyberspace in 2006, where this work began. Most of this work was performed while S.G., D.X, and B.B were at Princeton University, and E.T. was at MIT. S.G. and J.R were supported by HSARPA grant 1756303. S.G. was supported by NSF grant CNS-0627526. D.X. was supported by an NDSEG Graduate Fellowship and an NSF Graduate Research Fellowship. E.T. was supported by Rothschild Fellowship. B.B. was supported by NSF grants CNS-0627526 and CCF-0426582, US-Israel BSF grant 2004288 and Packard and Sloan fellowships.

References

[1] D. Achlioptas. Database-friendly random projections. In *PODS*, pages 274–281, 2001.

[2] R. Ahlswede and A. Winter. Strong converse for identification via quantum channels. *IEEE Trans. IT*, 48(3):569–579, 2002.

[3] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *STOC*, pages 20–29, 1996.

[4] Y. Amir, P. Bunn, and R. Ostrovsky. Authenticated adversarial routing. In *Theory of Cryptography Conference, TCC*, 2009.

[5] D. Angluin and L. G. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. *Journal of Computer and system Sciences*, 18(2):155–193, 1979.

[6] K. Argyraki, P. Maniatis, O. Irzak, A. Subramanian, and S. Shenker. Loss and delay accountability for the Internet. *ICNP*, 2007.

[7] I. Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy. Highly secure and efficient routing. In *IEEE INFOCOM*, 2004.

[8] I. Avramopoulos and J. Rexford. Stealth probing: Data-plane security for IP routing. *USENIX*, 2006.

[9] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens. An on-demand secure routing protocol resilient to byzantine failures. In *ACM WiSE*, 2002.

[10] Y. Bai, G. Shou, Y. Hu, and Z. Guo. High performance pipelined architecture of ghash. In *Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on*, pages 716–720. IEEE, 2010.

[11] B. Barak, S. Goldberg, and D. Xiao. Protocols and lower bounds for failure localization in the Internet. In *IACR EUROCRYPT*, 2008.

[12] D. J. Bernstein. Polynomial evaluation and message authentication. Technical report, http://cr.yp.to/ Document ID: b1ef3f2d385a926123e1517392e20f8c., October 2007.

[13] P. Bunn and R. Ostrovsky. Asynchronous throughput-optimal routing in malicious networks. In *ICALP (2)*, pages 236–248, 2010.

[14] J. L. Carter and M. N. Wegman. Universal classes of hash functions. *JCSS*, 18(2):143–154, 1979.

[15] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004.

[16] B. Claise. Packet sampling (psamp) protocol specifications. 2009.

[17] J. Cowie. Rensys blog: China's 18-minute mystery, 2010. http://www.renesys.com/blog/2010/11/chinas-18-minute-mystery.shtml.

[18] J. Daemen and V. Rijmen. A new MAC construction ALRED and a specific instance ALPHA-MAC. In *FSE: Fast Software Encryption*, volume 3557, pages 1–17. Springer, 2005.

[19] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard), 2006.

[20] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *J. of the ACM*, 40(1), 1993.

[21] N. G. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. *IEEE/ACM Trans. Networking*, 9(3), 2001.

[22] S. Gaudin. Interview with a convicted hacker: Robert Moore tells how he broke into routers and stole VoIP services. *InformationWeek*, September 26 2007.

[23] S. Goldberg and J. Rexford. Security vulnerabilities and solutions for packet sampling. *IEEE Sarnoff Symposium*, 2007.

[24] S. Goldberg, D. Xiao, E. Tromer, B. Barak, and J. Rexford. Path quality monitoring in the presence of adversaries. In *SIGMETRICS*, June 2008.

[25] S. Goldberg, D. Xiao, E. Tromer, B. Barak, and J. Rexford. Path-quality monitoring in the presence of adversaries: The secure sketch protocols. Technical report, Computer Science, Boston University, 2014.

[26] O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2007.

[27] K. J. Houle and G. M. Weaver. Trends in denial of service attack technology. Technical report, CERT Coordination Center, October 2001.

[28] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. pages 230–235, 1989.

[29] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 44–61. ACM, 1989.

[30] Keynote. Keynote launches new SLA services, June 2001. http://investor.keynote.com/phoenix.zhtml?c=78522&p=irol-newsArticle_Print&ID=183745.

[31] H. Krawczyk, M. Bellare, and R. Canetti. HMAC : Keyed-Hashing for Message Authentication. RFC 2104, 1997.

[32] T. Krovetz and W. Dai. VMAC, April 2007. CFRG Working Group, Internet Draft.

[33] K. Levchenko. Chernoff bound. www-cse.ucsd.edu/~klevchen/techniques/chernoff.pdf, 2008.

[34] M. Luckie, K. Cho, and B. Owens. Inferring and debugging path MTU discovery failures. In *Internet Measurement Conference*, 2005.

[35] D. A. McGrew and J. Viega. The security and performance of the galois/counter mode (GCM) of operation. In *INDOCRYPT*, pages 343–355. Springer-Verlag, 2004.

[36] I. Mironov, M. Naor, and G. Segev. Sketching in adversarial environments. In *STOC*, 2008.

[37] A. T. Mizrak, Y.-C. Cheng, K. Marzullo, and S. Savage. Detecting and isolating malicious routers. *Dependable and Secure Computing, IEEE Transactions on*, 3(3):230–244, 2006.

[38] J. Naous, M. Walfish, A. Nicolosi, D. Mazieres, M. Miller, and A. Seehra. Verifying and enforcing network paths with icing. In *CoNEXT*, page 30. ACM, 2011.

[39] A. Nucci. Skype detection: Traffic classification in the dark, 2006. http://www.narus.com/_pdf/news/Converge-Skype-Detection.pdf.

[40] V. N. Padmanabhan and D. R. Simon. Secure traceroute to detect faulty or malicious routing. *ACM SIGCOMM Computer Communication Review*, 33(1):77–82, 2003.

[41] R. Perlman. *Network Layer Protocols with Byzantine Robustness*. PhD thesis, MIT, 1988.

[42] J. Sommers, P. Barford, N. Duffield, and A. Ron. Improving accuracy in end-to-end packet loss measurement. In *ACM SIGCOMM*, 2005.

[43] J. Sommers, P. Barford, N. Duffield, and A. Ron. Accurate and efficient SLA compliance monitoring. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 109–120. ACM, 2007.

[44] E. Stephan. Ip performance metrics (ippm) metrics registry. 2005.

[45] J. Stone and C. Partridge. When the CRC and TCP checksum disagree. In *ACM SIGCOMM*, 2000.

[46] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. H. Katz. Listen and Whisper: Security mechanisms for BGP. In *NSDI*, 2004.

[47] M. Thorup and Y. Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *SODA*, pages 615–624, 2004.

[48] E. Vos. List of bad isps shaping p2p traffic - muniwireless, 2009. www.muniwireless.com/2009/01/16/is-your-isp-a-bad-isp/.

[49] E. L. Wong, P. Balasubramanian, L. Alvisi, M. G. Gouda, and V. Shmatikov. Truth in advertising: Lightweight verification of route integrity. In *PODC*, 2007.

[50] J. Xu. Tutorial on network data streaming. In *ACM SIGMETRICS*, 2008.

[51] X. Zhang, A. Jain, and A. Perrig. Packet-dropping adversary identification for data plane security. In *CoNEXT: Conference on emerging Networking EXperiments and Technologies*, December 2008.

[52] X. Zhang, Z. Zhou, G. Hasker, A. Perrig, and V. Gligor. Network fault localization with small TCB. In *IEEE International Conference on Network Protocols (ICNP)*, pages 143–154, 2011.

[53] X. Zhang, Z. Zhou, H.-C. Hsiao, T. H.-J. Kim, A. Perrig, and P. Tague. Shortmac: Efficient data-plane fault localization. *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2012.

Sharon Goldberg is an assistant professor in the Computer Science Department at Boston University, focusing on network security. She received her Ph.D. from Princeton University in 2009, has worked at Cisco, IBM Research, Microsoft Research, Bell Canada, and Hydro One Networks, and has served on working groups of the FCC and IETF. In 2014 she received two IETF/IRTF Applied Networking Research Prizes and a Sloan Fellowship.

David Xiao is a CNRS researcher in the Algorithms and Complexity Group at LIAFA, focusing on complexity theory, cryptography, and privacy.

Eran Tromer received the Ph.D degree in computer science from the Weizmann Institute of Science, Israel, in 2007. His research focus is cryptography and information security. He is a Senior Lecturer at the School of Computer Science, Tel Aviv University, and codirector of the Check Point Institute for Information Security. Formerly he was at MIT and at Microsoft Research.

Boaz Barak is a Senior Researcher at Microsoft Research, New England. He received his Ph.D in 2004 from the Weizmann Institute of Science. Following a postdoc at the Institute for Advanced Study, he joined the faculty of Princeton University where he was most recently an associate professor of Computer Science. He is interested in theoretical computer science, and in particular cryptography and computational complexity. He has won the ACM doctoral dissertation award in 2004 and a Packard fellowship in 2007.

Jennifer Rexford is the Gordon Y.S. Wu Professor of Engineering in the Computer Science department at Princeton University. Before joining Princeton in 2005, she worked for eight years at AT&T Labs–Research. Jennifer received her BSE degree in electrical engineering from Princeton University in 1991, and her PhD degree in electrical engineering and computer science from the University of Michigan in 1996. She is co-author of the book "Web Protocols and Practice" (Addison-Wesley, May 2001). She served as the chair of ACM SIGCOMM from 2003 to 2007. Jennifer was the 2004 winner of ACM's Grace Murray Hopper Award for outstanding young computer professional. She is an ACM Fellow (2008), and a member of the American Academy of Arts and Sciences (2013) and the National Academy of Engineering (2014).