TRACER: Flexible and Responsible Distributed Active Network Measurements

Yi-Min Lin, Mark Csurgay, Jonghoon Kwon, Adrian Perrig
Department of Computer Science, ETH Zürich, Zürich, Switzerland

***Abstract*—Distributed active measurement platforms provide critical visibility into Internet behavior. However, they face a fundamental tension: flexible, user-defined measurements provide richer observations but create significant abuse risks. Platforms that admit untrusted users restrict measurement flexibility to mitigate those risks, while platforms supporting flexible measurements only open to trusted users. We present TRACER, a system that extends flexible active measurements to untrusted users through two complementary mechanisms: a per-target rate limiter that enforces pre-defined traffic policies and a cryptographic accountability framework that binds every received probe to its originating measurement and user. Together, these mechanisms deter abuse by ensuring that measurement traffic is both volume-bounded and attributable, without sacrificing measurement fidelity. Our work establishes a new design paradigm for open distributed measurement platforms: rather than restricting what users can measure, the system holds users cryptographically accountable for the traffic they generate, shifting the security burden from access control to attribution.**

***Index Terms*—Distributed measurement, programmable measurement, distributed rate-limit, accountability**

I. Introduction

Distributed active measurement platforms face a fundamental tension between *measurement flexibility* and *openness*. To capture broader network phenomena, systems have progressively increased programmability through virtual machines [4], containers [2], [29], shell access [3], scripting environments [24], [18], and WebAssembly [26]. However, since existing isolation mechanisms (e.g., virtualization and sandboxing) protect the measurement infrastructure rather than the external environment, enabling arbitrary measurement traffic introduces security risks: the measurement platform can be abused for large-scale reconnaissance or distributed denial-of-service attacks [21]. Consequently, platforms either restrict the programmability to only trusted, verified users [2], [4], [3], [17] or offer public access with limited, predefined measurements [5], [1], [24], as shown in Table I.

Prior mitigations using static, local rate limits, e.g., Scriptroute [24], remain effective for basic abuse prevention, but they fall short: independent local enforcement fails to prevent the aggregate impact of coordinated distributed traffic, and static rules cannot anticipate complex malicious behaviors.

In this paper, we argue for a different direction: rather than attempting to fully regulate measurements through infrastructure-level restrictions, distributed measurement platforms should make users *responsible* for the traffic they generate. Our design is motivated by recent discussions on responsible Internet measurements [9], which advocate stronger attribution and opt-out mechanisms for measurement infrastructures. Building on this idea, we propose TRACER, a framework for publicly accessible and programmable distributed measurements based on two principles: *global resource control* and *cryptographic accountability*.

First, we introduce *measurement profiles* with *target-oriented policies*. Instead of regulating traffic only from the perspective of the vantage point, users explicitly declare measurement properties such as targets, bandwidth requirements, or execution duration. The platform then enforces globally coordinated resource limits across all vantage points, ensuring that aggregate traffic remains bounded and compliant with target-specific policies. This shifts traffic regulation from coarse source-oriented restrictions toward explicit declarations of intended measurement behavior.

Second, we extend the concept of probe attribution [28] with cryptographic accountability. Existing attribution mechanisms attach identifying information to probes, but the information is neither cryptographically bound to the originating user nor resistant to forgery. Consequently, malicious users may deny responsibility for abusive traffic, while attackers can spoof probes to falsely implicate the platform. Such accountability is what allows a vantage point to support more flexible, higher-risk measurements without relying on restrictive heuristic rules. Responsibility is verifiably shifted to the user, letting an honest vantage point disprove involvement in abusive traffic, while any operator deviating from the protocol remains accountable for its own misbehavior. We design an accountability protocol that cryptographically binds probes to measurements and enables efficient verification using lightweight symmetric cryptography.

We implement a proof-of-concept programmable measurement platform supporting WebAssembly-based measurements and common tools such as `ping` and `iperf`. Our evaluation shows that the proposed mechanisms impose negligible bandwidth and computational overhead while preserving measurement fidelity. The design is compatible with existing distributed measurement infrastructures that rely on centralized coordination, such as RIPE Atlas.

Our contributions are as follows:

- We propose a centralized rate-limiting mechanism that enforces bandwidth constraints across vantage points while supporting prefix-based policies and fair resource sharing.

979-8-3195-0662-7/26/$31.00 ©2026 IEEE

- We design a lightweight attribution protocol based on delayed key disclosure that cryptographically binds probes to measurements. Our design achieves efficient verification using only symmetric cryptography and requires only a compact 16-bit attribution tag.
- We implement the proposed mechanisms within a programmable, distributed measurement platform that supports WebAssembly execution, and evaluate its practicality and measurement fidelity.

II. Motivation and Challenges

We present the design goals for a measurement platform that admits untrusted users without compromising the safety of operators or targets, and the challenges in achieving them.

A. Design Goals

Flexibility. The system should grant users control over probe types and transmission patterns, enabling experiments beyond the predefined templates offered by existing platforms. It should also be accessible to a wide range of users without requiring manual account or measurement vetting, reducing administrative overhead and fostering an open ecosystem.

Responsibility. All probes generated by the system should be attributed to their origin, including the platform, the operator, and the user. Such attribution enables identifying the source of measurement traffic, enabling affected targets to opt-out as a way to improve measurement ethics [9].

Fidelity. Security enforcement should not distort measurement results. In particular, enforcement mechanisms must introduce minimal overhead and avoid artifacts that interfere with timing, rate control, or protocol behavior.

B. Technical Challenges

Distributed Rate Limiting. Enforcing rate limit aggregated from a distributed measurement platform is challenging. Because each node only has partial visibility, bounding aggregated traffic requires coordination between nodes and introduces challenges regarding scalability. Existing systems such as Scriptroute and Planetlab rely on per-node rate limiting, which fails to prevent coordinated traffic surges across nodes.

This challenge is further complicated when considering measurement fidelity and resource utilization. Some measurements generate steady traffic while some generate bursty traffic. Conservative rate limiting can therefore compromise measurement fidelity through excessive throttling or poor resource utilization. The key challenge is to enforce rate limits while enabling efficient bandwidth allocation across measurements.

Lightweight Accountability. Providing accountability while maintaining measurement fidelity is a challenge. Digital signatures offer authentication and non-repudiation but are computationally prohibitive to distort measurement results and are infeasible at high packet rates. Symmetric-key approaches such as per-packet message authentication codes [19], [15] are lightweight but require the verifier to hold the signing key, which also enables the verifier to forge packets falsely attributed to the measurement. The challenge is therefore to achieve efficient per-packet authentication while preserving public verifiability.

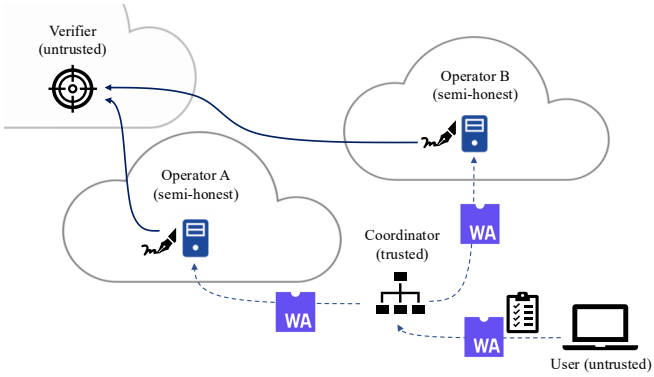


Figure 1: System architecture. Untrusted users submit measurement profiles to the trusted coordinator, which validates them against security policies and dispatches execution to operators. Operators enforce the declared profiles at runtime and cryptographically sign all outgoing traffic.

III. System Model

We present the system model with adversarial assumptions and the desired properties to achieve.

A. Architecture

We consider a system model focusing on the interaction among the following entities: a centralized *coordinator*, vantage point *operators*, *users*, and *verifiers*. The coordinator is responsible for admitting and orchestrating measurements submitted by users and scheduling their execution on vantage points operated by the operators. Operators execute measurements on their nodes and interact with the targets on the Internet, as shown in Fig. 1. A verifier can verify the observed measurements and report abuses to the operator.

Trust Assumptions. We assume a trusted coordinator and semi-honest operators: they correctly execute the protocol but do not necessarily trust other operators. The users and the verifiers are not trusted.

Adversary Capabilities. An adversarial user may intentionally or unintentionally violate platform policies, such as generating volumetric traffic, crafting malformed packets (e.g., spoofed source addresses), or targeting prohibited destinations; attackers on the network may craft malicious attack traffic accountable to our system, and a dishonest verifier may also falsely report abuse.

B. Desired Properties

The design aims to satisfy the following properties.

- **Completeness:** Every probe generated by the system can be verified and attributed to its originating measurement.
- **Soundness:** An adversary cannot forge probes that are falsely attributed to the system.

Table I: Comparison of distributed active measurement platforms.

Platform	Probe Flexibility	User Trust	Rate Limiting	Probe Attribution	Post-hoc Verifiable
PlanetLab [4]	High (VM)	Vetted	Per-vantage point	Metadata-based [12]	✗
M-Lab [2]	High (Container)	Vetted	None	None	✗
NLNOG RING [3]	High (Shell)	Vetted	None	None	✗
RIPE Atlas [5]	Low (Static)	Open	Per-vantage point & per-target [1]	Unsigned	✗
CAIDA Ark [1]	Low (Static)	Open	Per-vantage point	None	✗
Scriptroute [24]	Medium (Script)	Open	Per-vantage point	None	✗
Debuglet [26]	Medium$^+$ (Wasm) [2]	Open	None	None	✗
TRACER	Medium$^+$ (Wasm)	Open	Per-target	Accountability	✓

[1] RIPE Atlas limits the number of concurrent measurements per target (25 periodic and 25 one-off by default) [6].
[2] Wasm can be compilation target for main-stream programming languages including C/C++, Golang, Rust, JavaScript, etc.

- **Non-repudiation:** The system cannot deny probes that it has legitimately generated.
- **Per-target rate limit:** The system should ensure a rate limit to a target regardless of the number of measurements or nodes.
- **Fairness:** The system should employ fair sharing for the limited bandwidth shared among the measurements.
- **Low overhead:** The system should incur low computational and bandwidth overhead to a measurement to preserve measurement fidelity.

IV. TRACER

We propose TRACER, a framework to safeguard a distributed measurement system. The system consists of two main components: (i) a resource management framework that enforces policy-compliant execution (Section IV-B), and (ii) an accountable probe attribution mechanism that enables verifiable measurement provenance (Section IV-C). Table I summarizes how TRACER compares to existing distributed measurement platforms across the properties defined above.

A. Overview

1) Measurement Profiles with Runtime Enforcement: Measurements are specified through *measurement profiles* that declare the intended targets, protocols, and resource requirements, most importantly, the required bandwidth. Profiles serve as the interface between user intent and system enforcement. Before execution, each profile is validated against applicable policies; non-compliant measurements are rejected, and accepted ones receive predictable resource allocations that operators can provision.

Pre-execution validation alone cannot guarantee that runtime behavior matches the declared profile. The system therefore enforces profiles at runtime, monitoring resource usage in real time and ensuring that execution remains within policy bounds.

2) Fair Resource Allocation: The rate limits imposed by the security policy introduce contention among concurrent measurements. To ensure fairness and prevent abuse, the system adopts max-min fair sharing for bandwidth allocation. Under this model, measurements with lower demand are satisfied first, while higher-demand measurements are progressively throttled only when contention arises. This approach prevents any single measurement from monopolizing resources and preserves the measurement fidelity for low-intensity workloads.

3) Accountability with Scheduled Key Disclosure: Lightweight accountability is achieved using symmetric cryptography with scheduled key disclosure [20]. Each outgoing probe is tagged with a message authentication code under a secret key that will be disclosed after a bounded delay. Before disclosure, only the originating operator possesses the key, so no external party can forge valid tags and the operator cannot deny probes it has signed. Targets store received probes and verify them once the key is published. This achieves per-probe accountability without per-packet digital signatures, providing public verifiability after the disclosure delay.

B. Distributed Rate Limiting

We present a system that enforces ingress and egress rate of both global policy and local policy, while achieving max-min fairness between operators and measurements. The system aims at two-level fairness: bandwidth should be distributed max-min fairly across all operators, and the allocated bandwidth at each node must be shared fairly among its running measurements to prevent any single task from monopolizing resources.

Goal. To model the hierarchical architecture of the platform, we define the system capacity and allocation constraints using a structured set of indices. Let n denote the number of active vantage points (nodes), and let m_i represent the number of active measurements running on node i, where $1 \leq i \leq n$. For each measurement j (where $1 \leq j \leq m_i$), the requested bandwidth floor and ceiling are denoted as $r_{i,j}^{\text{floor}}$ and $r_{i,j}^{\text{ceil}}$, respectively.

The system computes two allocation metrics: $A_{i,j,d}$, which is the bandwidth allocated to measurement j on node i for a specific destination d, and $A_{i,j}$, which represents the total aggregate bandwidth allocated to measurement j across all its destinations on node i, defined as $A_{i,j} = \sum_d A_{i,j,d}$.

Let C_i^{node} be the total capacity of node i, and C_d^{dest} be the capacity limit of destination d. The allocation matrix must satisfy the following constraints:

$$\sum_{j=1}^{m_i} A_{i,j} \le C_i^{\text{node}} \qquad \forall i \in 1,\dots,n \tag{1}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i} A_{i,j,d} \le C_d^{\text{dest}} \qquad \forall d \tag{2}$$

$$r_{i,j}^{\text{floor}} \le A_{i,j} \le r_{i,j}^{\text{ceil}} \qquad \forall i,j,d \tag{3}$$

The objective of the system is to calculate the allocation matrix A such that the resulting bandwidth distribution satisfies these constraints while achieving max-min fairness across the entire platform.

Distributed Two-Phase Allocation. The centralized calculation of $A_{i,j,d}$ for every individual measurement does not scale with increasing numbers of nodes n and measurements m. To improve scalability and reduce control-plane latency, the max-min allocation is split into two phases. This decomposition allows a global coordinator to manage destination-side bottlenecks while each vantage point independently enforces its local capacity constraints.

- **Global phase (Destination Demand Allocation):** When a measurement is added or removed, the coordinator recalculates the global allocation shares for each affected destination d. It aggregates the ceiling requests across all nodes to evaluate destination saturation. If the aggregate demand exceeds the destination capacity C_d^{dest}, the coordinator computes a measurement and destination-specific saturation level $\theta_{j,d}$ satisfying:

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i} \min\left(r_{i,j}^{\text{ceil}}, \theta_{j,d}\right) = C_d^{\text{dest}} \tag{4}$$

  The coordinator then propagates the updated saturation matrix θ to the relevant vantage points. If a destination is unsaturated, $\theta_{j,d}$ defaults to ∞.
- **Local phase (Node-Level Allocation):** The local operator node recomputes its internal allocation shares upon receiving an updated saturation matrix θ, or when a local measurement lifecycle event occurs. To prevent a node from overcommitting its own resources or violating remote destination caps, the operator computes a local saturation level ϕ_i bounded by its capacity C_i^{node}

$$\sum_{j=1}^{m_i} \min\left(r_j^{\text{ceil}}, \phi_j\right) = C_i^{\text{node}} \tag{5}$$

  The final total rate for each measurement is established as $A_{i,j} = \min(r_{i,j}, \phi_j)$, while the node limits the rate per destination as $A_{i,j,d} = \min(\theta_j, \phi_j)$.

C. Measurement Accountability

We propose a measurement accountability scheme that extends probe attribution [22] and Internet source accountability [14] with cryptographic guarantees: TRACER's probes are cryptographically bound to its originating measurement, operator, and epoch, ensuring both the authenticity and integrity of probe attribution. Figure 2 illustrates the end-to-end accountability protocol between the operator, the bulletin board, and the target.

Table II: Notation

O_j	The j-th operator in the distributed measurement platform.
M_i	The i-th measurement running on an operator.
$k_{O_j,t}$	Symmetric key of operator O_j valid during time interval t, derived from its hash chain.
$k_{M_iO_j,t}$	Symmetric measurement-specific key derived for measurement M_i in operator O_j at time t.
p	The probe payload.
$\mathsf{H}(\cdot)$	A one-way function used to construct the cryptographic hash chain.
$\mathsf{PRF}_k(x)$	A pseudorandom function evaluated on x using symmetric key k.
$\mathsf{MAC}_k(m)$	Message Authentication Code (MAC) of message m using symmetric key k.
$K_{O_j}^+, K_{O_j}^-$	Public and private key pair of the j-th operator.
$\{m\}_{K^-}$	Message m with its digital signature signed with K^-.

1) Notation and preliminaries: Table II summarizes the cryptographic notations and system parameters used in the protocol description. We use $t \in \mathbb{N}$ to denote a discrete time interval (epoch), and $\|$ for concatenation.

TESLA key schedule. We leverage the TESLA [20] protocol to achieve public verification with symmetric cryptography. TESLA provides delayed authentication by scheduled key reveal. An entity generates a one-way key chain and uses the keys in reverse order to compute MACs for outgoing packets. A key used in interval t is kept secret until a pre-defined disclosure delay d has elapsed. Once the key is publicly revealed, verifiers can verify the MACs of the packets received during interval t.

2) Key Schedule Setup: Each operator O_j initializes a TESLA-style key schedule and publishes a commitment to a publicly verifiable bulletin board (e.g., a transparency log or blockchain). It selects the schedule parameters (L, I, d), where L is the chain length, I is the interval duration, and d is the disclosure delay. The operator O_j generates a hash chain $\{k_{O_j,0}, \dots, k_{O_j,L}\}$ such that $k_{O_j,\tau} = \mathsf{H}(k_{O_j,\tau+1})$.

The operator authenticates itself to the platform P and publishes the signed parameters $\{L, I, d, k_{O_j,0}, t_0\}_{K_{O_j}^-}$, where t_0 denotes the start time and $k_{O_j,0}$ serves as the public anchor. During execution, the operator O_j uses key $k_{O_j,t}$ in interval t, advances keys every I time units, and reveals $k_{O_j,t}$ after d intervals. Upon disclosure, a verifier checks correctness by validating $\mathsf{H}^t(k_{O_j,t}) = k_{O_j,0}$, taking into account the temporal offset from t_0.

3) Probe Tagging: For each probe p of measurement M_i generated at interval t by operator O_j, the operator derives a measurement-specific key $k_{M_iO_j,t} = \mathsf{PRF}_{k_{O_j,t}}(M_i)$ and computes the authentication tag $\sigma = \mathsf{MAC}_{k_{M_iO_j,t}}(p)$. The transmitted probe is $P = p \,\|\, \sigma$, which cryptographically binds the payload, operator, measurement, and epoch.

4) Verification: Upon receiving a probe P, the verifier retrieves the published parameters $\{L, I, d, k_{O_j,0}, t_0\}_{K_{O_j}^-}$, the

operator's public key $K^+_{O_j}$, and the latest disclosed key $k_{O_j,\tau}$. The verifier identifies the epoch t, the operator O_j and the measurement M_i using the out-of-band attribution method described in Section VI-A4.

The verifier first validates the digital signature of the parameters using $K^+_{O_j}$. If the key corresponding to interval t has been disclosed (i.e., $\tau - t \geq d$), the verifier reconstructs the interval key $k_{O_j,t} = \mathsf{H}^{\tau-t}(k_{O_j,\tau})$, checks its consistency via $\mathsf{H}^t(k_{O_j,t}) = k_{O_j,0}$, derives the measurement-specific key $k_{M_iO_j,t} = \mathsf{PRF}_{k_{O_j,t}}(M_i)$, and verifies the tag by checking $\mathsf{MAC}_{k_{M_iO_j,t}}(p) \stackrel{?}{=} \sigma$.

If the key $k_{O_j,t}$ has not yet been disclosed ($\tau - t < d$), the verifier cannot independently validate the probe and must either query the operator O_j for confirmation or defer verification until the scheduled disclosure. In the former case, the operator O_j verifies the probe using its currently undisclosed key $k_{O_j,t}$ and returns a response indicating whether the probe is attributable to measurement M_i at interval t.

5) Time Synchronization: The protocol does not require tight clock synchronization. The target maps probe arrival timestamps to epochs using the published parameters t_0 and I; a clock skew of δ shifts the computed epoch by at most $\lceil \delta / I \rceil$, which is zero for NTP-synchronized hosts ($\delta < 10$ms) and epoch intervals on the order of seconds as least. Probes signed near an epoch boundary may arrive in the adjacent epoch from the target's perspective; the verifier handles this by attempting verification against both the computed epoch and its immediate predecessor. The epoch interval I should be chosen large enough that network delay remains a small fraction of I.

V. Security Analysis

A. Distributed Rate Limiting

In steady state, the coordinator enforces a separate global capacity $\mathbb{C}$ for each relevant destination: for any active measurement set $\mathcal{M}$, the allocated fair shares satisfy $\sum_{m \in \mathcal{M}} A_m \leq \mathbb{C}$. To avoid blocking when operators disconnect, the system adopts a *simultaneous start and update* strategy—new measurements begin immediately, introducing a transient convergence window Δ before all operators adopt revised fair shares.

1) Worst-case transient overshoot: Suppose a destination is fully saturated by $\mathcal{M}_{\text{old}}$ with $\sum_{m \in \mathcal{M}_{\text{old}}} A_m = \mathbb{C}$, and a batch of new measurements $\mathcal{M}_{\text{new}}$ arrives. The coordinator compresses $\mathcal{M}_{\text{old}}$ to their minimum floors and allocates the freed capacity to the newcomers:

$$\sum_{m \in \mathcal{M}_{\text{new}}} A'_m = \mathbb{C} - \sum_{m \in \mathcal{M}_{\text{old}}} r_m^{\text{floor}}. \tag{6}$$

During Δ, operators in $\mathcal{M}_{\text{new}}$ transmit at their new allocation while stale operators in $\mathcal{M}_{\text{old}}$ continue at the pre-update rate $\mathbb{C}$. The peak transient traffic is therefore:

$$\Gamma_{\max} \leq \mathbb{C} + \Big(\mathbb{C} - \sum_{m \in \mathcal{M}_{\text{old}}} r_m^{\text{floor}}\Big) = 2\mathbb{C} - \sum_{m \in \mathcal{M}_{\text{old}}} r_m^{\text{floor}} < 2\mathbb{C}. \tag{7}$$

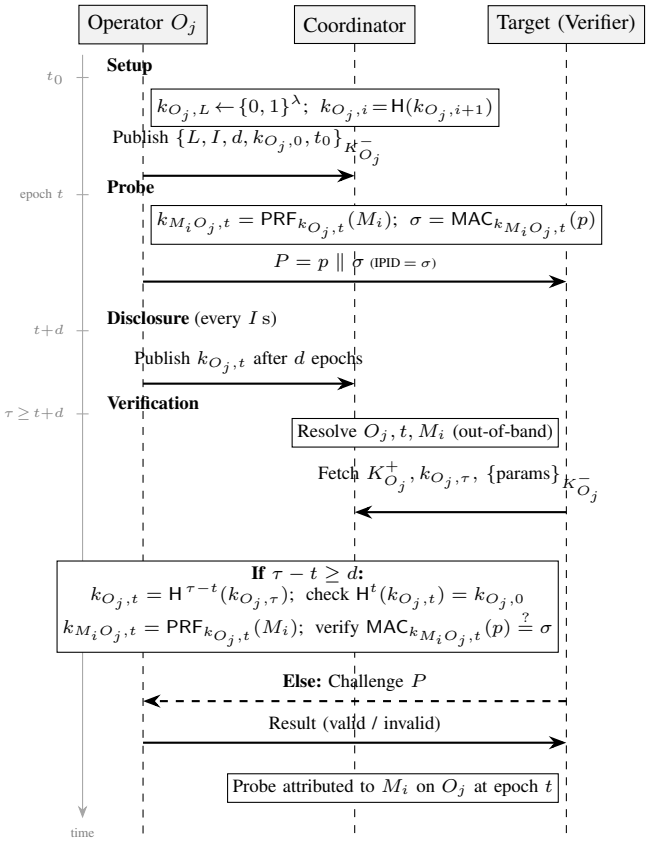


Figure 2: The measurement accountability protocol.

The overshoot is thus strictly sub-$2\mathbb{C}$, and enforcing a positive minimum floor ($f_m > 0$) tightens this bound further.

2) Bounding Cascading Arrivals: The $2\mathbb{C}$ bound assumes a single, isolated reallocation event. If new measurements arrive *continuously* within Δ, continuous reallocation compounds out-of-sync operators, causing transient traffic to scale as $(k{+}1)\mathbb{C}$ for k arrivals—an exposure an adversary could exploit through rapid measurement churn to mount a control-plane amplification attack.

We address this with *epoch-based admission control*: the coordinator batches all incoming requests and triggers a single global reallocation at fixed epoch boundaries (e.g., $E = 100$ ms). Provided $E > \Delta$, every operator has converged before the next reallocation fires, so at most one out-of-sync window can be active at any time. This reduces the worst-case transient to $2 \cdot \mathbb{C}$ event per epoch, regardless of arrival rate.

B. Probe Accountability

1) Completeness: Every probe from an honest operator verifies. For $P = p \parallel \sigma$ produced under $K_{M_i,O_j,t}$, once $K_{O_j,t}$ is disclosed and M_i identified, the verifier recomputes $K_{M_i,O_j,t} = \mathsf{PRF}_{K_{O_j,t}}(M_i)$ and checks σ; correctness follows from MAC correctness and the deterministic key derivation.

2) Soundness and Non-repudiation: Both hold within the pre-disclosure window and share the same basis: the one-wayness of the hash chain, which prevents deriving $K_{O_j,t}$

from earlier disclosed keys, and the existential unforgeability of the MAC under chosen-message attack. Before $K_{O_j,t}$ is revealed, no adversary can forge a valid tag for a fresh payload in interval t (soundness), and O_j is the sole party able to produce valid tags for t, even if $K_{O_j,t-1}$ is already disclosed (non-repudiation). Any probe observed in this window is therefore attributable to O_j.

3) Temporal Guarantee: Soundness and non-repudiation hold only within the pre-disclosure window. Once $K_{O_j,t}$ is revealed, any party can derive $K_{M_i,O_j,t} = \mathsf{PRF}_{K_{O_j,t}}(M_i)$ and forge valid tags for interval t, an inherent limitation of delayed-disclosure schemes. The guarantee remains meaningful to honest verifiers because attribution is anchored to the verifier's observation time, not to a timestamp in the probe: assume the disclosure delay exceeds the maximum transmission time, any probe an honest verifier records while $K_{O_j,t}$ is still secret must originate from O_j, whereas probes seen after disclosure could be forged. A dishonest verifier can also forge an abuse claim, but similarly the system can only consider claims issued before key disclosure, assuming a long enough disclosure delay for the verifiers to react.

4) Replay Attacks: An attacker can replay observed probes to violate soundness but such attack can be mitigated using replay suppression techniques. Since it cannot forge a valid tag σ, it must replay the entire probe; assuming each probe is unique, duplicates are detectable with an efficient structure such as a Bloom filter [16], so only the first copy survives.

C. Accountability with Short Tag

We show that a short tag, e.g. 16 bits, suffices to achieve the properties above. The core idea is that the verifier can only verify *interactively*—online, before the key is revealed—and that combining this with rate-limited, all-or-nothing verification makes forging a full flow of valid tags negligible, while the tag itself fits in the IP header at no bandwidth cost.

We model an adversary that queries the verifier R times before key disclosure. Each query submits an arbitrary number of probes, and the verifier returns *true* only if *all* of them verify; this all-or-nothing response is essential, as per-probe answers would leak far more and ease forgery. Assuming the MAC is a secure PRF under a uniformly random key, each fresh probe's 16-bit tag is uniform and independent, so a single guess succeeds with probability $p = 2^{-16}$. The adversary's goal is to forge N verifying probes.

Submitting one probe per query is the adversary's best strategy: batching all N into one query requires every tag correct in a single trial, succeeding with only $R \cdot 2^{-16N}$ (e.g. $10^4 \cdot 2^{-160}$ for $N = 10$, $R = 10^4$). Under the individual strategy, over R Bernoulli trials with mean $\lambda = Rp$, the probability of obtaining at least N forgeries is

$$P_{\text{indiv}} = \sum_{j=N}^{R} \binom{R}{j} p^j (1-p)^{R-j} \approx 1 - \sum_{j=0}^{N-1} \frac{e^{-\lambda}\lambda^j}{j!}, \qquad (8)$$

which for $N = 10$, $R = 10^4$ gives $P_{\text{indiv}} \approx 1.2 \times 10^{-14}$. Thus, with a global budget of R queries before key disclosure, the probability of forging a complete flow of N packets stays below 10^{-13} for modest N (Fig. 3). While per-packet non-repudiation is deferred until disclosure, the scheme achieves strict *aggregate* non-repudiation over the flow.

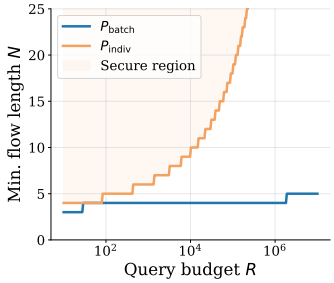


Figure 3: Parameter selection: minimal N under different R to reach the 10^{-13} forgery bound.

VI. Implementation and Evaluation

A. Implementation

Our implementation comprises approximately 9K lines of Go; the source[1] and a public deployment[2] are publicly available. We now describe the measurement implementation and the core components.

1) User Measurements: For our benchmarks, guest measurements are authored as Go functions compiled into WASI preview1-compatible WebAssembly (Wasm) binaries. Because Wasm executes in an isolated sandbox, we expose minimal host functions to the runtime, enabling guest code to establish TCP or ICMP connections and handle network I/O. Users submit these binaries to the coordinator via HTTPS, accompanied by a configuration policy containing four critical parameters:

1) **Targeted Destinations:** A declared set of destination IP addresses the measurement intends to probe. Any attempt to connect to an undeclared address is rejected at the host boundary.
2) **Expected Duration:** The anticipated runtime of the measurement, used by the coordinator to schedule and bound execution.
3) **Bandwidth Floor:** The minimum guaranteed bandwidth allocated to the measurement during execution, ensuring a baseline quality of service regardless of contention from other measurements.
4) **Bandwidth Ceiling:** The maximum bandwidth a measurement may utilize, applied on a best-effort basis when operator or destination capacity is available.

2) Coordinator: The coordinator manages global system state and orchestrates operators via long-lived, bidirectional gRPC streams, eliminating the need for operators to expose public network endpoints. To minimize control-plane latency, the coordinator implements a reactive admission control mechanism. Max-min fairness allocations are re-evaluated exclu-

[1]https://github.com/netsec-ethz/debuglet
[2]https://debuglet.netsec.ethz.ch/

sively upon measurement registration or deregistration events rather than via periodic polling.

To calculate allocations in $\mathcal{O}(\log n)$ time, the coordinator maintains an AVL tree for every active destination prefix. To enforce a baseline minimum allocation, each node in the tree stores a measurement's excess capacity demand $D = r^{\text{ceil}} - r^{\text{floor}}$ rather than its absolute limits. Upon a lifecycle event, the coordinator computes a fair-shared excess bandwidth value f_D for the affected destination prefix. The absolute maximum bandwidth ceiling for a given measurement is then derived as $r^{\max} = \min\left(r^{\text{ceil}}, r^{\text{floor}} + f_D\right)$.

When a new measurement is registered, the coordinator executes a three-step admission pipeline: (1) validates the request against current destination and operator capacities; (2) updates the destination AVL trees; and (3) derives and serializes the state transitions into update tuples defined as $U = (M, D, C)$, representing the measurement identifier, destination prefix, and computed rate limit, respectively. These tuples are immediately dispatched to the respective operators via the open gRPC stream.

3) Operator: Each operator runs as an independent service that registers its local system capacities (C_i^{node}) with the coordinator. Operators can dynamically adjust these capacities at runtime. Every active measurement runs concurrently within an isolated Goroutine. Before initializing a guest binary, the operator evaluates local resource constraints to guarantee that C_i^{node} is never overcommitted.

Network I/O operations crossing the guest-host boundary specify a buffer size parameter C, which the host uses to account for the corresponding traffic volume. The operator enforces a hard buffer ceiling of 32 KiB per call to guarantee high-resolution rate limiting. Traffic shaping is managed via Token Buckets. The operator instantiates a token bucket for each destination listed in the measurement's policy (the set D_m) alongside a singular bucket for the operator node limit (the singleton set E_m).

Each bucket $b \in D_m \cup E_m$ is configured with a fill rate r_b matching its max-min allocated bandwidth and initialized with a burst capacity of $M_b = 32{,}768$ tokens. This allows small initial network requests to bypass shaping, while continuous or large transfers trigger throttling. For any given network invocation requesting a buffer size C, the operator clamps the input to $C' = \min(C, M_b)$ and evaluates the mandatory backoff delay $T_{\max}$ (in seconds) to satisfy all intersecting rate-limits:

$$T_{\max} = \max_{b \in D_m \cup E_m} \begin{cases} 0 & \text{if } C' \leq M_b \\ C'/r_b & \text{otherwise} \end{cases} \tag{9}$$

4) Probe Attribution: We implement probe attribution using eBPF attached to the TC egress hook, enabling efficient in-kernel packet tagging without userspace overhead.

In-Band Attribution. We embed the attribution tag in the IPv4 Identification (IPID) field to avoid inflating the packet size. When the do-not-fragment (DF) flag is set, this field serves no purpose and is conventionally randomized per RFC 6864 [27]; a pseudorandom authentication tag is therefore indistinguishable from normal traffic. This gives us a 16-bit tag space with no additional header bytes and no payload overhead. In Section V-C, we show that 16 bits are sufficient for meaningful accountability under rate-limited verification.

For IPv6, we use the Destination Options extension header to carry tags with at most 32 bits with the cost of 8 bytes inflation per packet. By pre-allocating the extension header at the socket layer before packet construction, the kernel accounts for the additional bytes in path MTU calculations, and the eBPF tagger can overwrite the tag field in place.

Out-of-Band Resolution of Verification Context. Since the 16-bit tag carries no inline metadata, the verifier recovers the verification context—the originating operator O, the measurement M, and the epoch t—out of band from the coordinator, which logs the interval each operator ran each measurement and publishes every operator's latest disclosed key $k_{O,\tau}$. The probe's source address identifies O, and its observation time yields the epoch $t = \lfloor (t_a - t_0)/I \rfloor$, from which the verifier reconstructs the chain key and authenticates it using the protocol described in Fig. 2. The epoch restricts the candidates to the measurements was running at t, and the verifier identifies M by trial verification: the tag only verifies using the correct measurement key derived from M.

eBPF Tagging. At measurement start, the operator stamps the transport socket with a 32-bit identifier via `SO_MARK`. On each egress packet the eBPF program reads `skb->mark`, looks up the per-measurement epoch key (k_0, k_1) in a `BPF_MAP_TYPE_HASH` map, and tags the packet; `mark == 0` bypasses the tagger at no overhead. The tag is a PRF over the canonicalized first 64 bytes (IP header and initial payload, mutable fields zeroed); since HMAC-SHA256 exceeds the BPF verifier's instruction limits, we use an inlined SipHash-2-4 keyed with (k_0, k_1) and truncate the digest to 16 bits. The carrier differs by address family. For IPv4, the tag overwrites the Identification field (zeroed with the header checksum before hashing), and the L3 checksum is repaired via `bpf_l3_csum_replace`. IPv6 has no comparable mutable field, so the program inserts an 8-byte Destination Options header via `bpf_skb_adjust_room`, writes the tag, and updates Next Header and Payload Length; canonicalization zeroes Traffic Class, Flow Label, and Hop Limit, and no checksum fixup is needed since extension headers lie outside the transport checksum.

TESLA Key chain parameters. Of the schedule parameters—epoch interval I, disclosure delay $d \cdot I$, chain length L, and pre-disclosure query budget R—only R affects soundness, since the aggregate forgery probability scales as $R \cdot 2^{-16N}$ for a N-probe flight (§V-C) and is independent of I. We therefore treat I and $d \cdot I$ as operational choices: I trades verification granularity against boundary ambiguity ($\approx \mathrm{RTT}/I$ of probes fall near an epoch boundary), and $d \cdot I$ must be long enough for a verifier to observe a probe flight before disclosure yet short enough to bound unilateral verification latency. We set $I = 10\,\mathrm{s}$ and $d \cdot I \approx 10$–$30\,\mathrm{min}$, and enforce

R as a hard budget per (operator, epoch), so that lengthening d never weakens soundness. The chain length determines the re-anchoring period $L \cdot I$: for a 30-day lifetime at $L \approx 2.6 \times 10^5$, yet the operator stores a single seed and derives keys on demand via fractal traversal [13] using only $\lceil \log_2 L \rceil \approx 18$ pebbles in $O(\log L)$ time. On exhaustion it re-anchors with a fresh self-signed root over a brief overlap window so in-flight verification is unaffected.

B. System Overhead

We evaluate the computational and communication overhead on an AMD Ryzen AI 9 HX 370 (coordinator) and AMD EPYC 7702P (operator), both running Linux.

1) Coordinator Overhead and Message Footprint: We measure the worst-case registration overhead under a maximum-collision model where all N active measurements share a destination, triggering N updates per event.

Measurement insertion scales logarithmically, requiring $\approx$ 553ns at $N = 10^3$ and rising marginally to $\approx$ 1118ns at $N = 10^6$. Because registration is synchronous, it bounds the critical path before execution handoff.

Conversely, computing the update matrix scales linearly with N, adding approximately 128 μs at $N = 10^3$ and 454.5 ms at $N = 10^6$. Asynchronous dispatch to operators (Section V-A) prevents this computation from delaying measurement initiation, although the sub-second cost at large N motivates batching. The wire overhead remains modest: update messages average 59 bytes per entry, resulting in a 59 MB payload at $N = 10^6$.

2) Operator Overhead: To establish a baseline for host-function invocation overhead, we design two microbenchmarks using a minimalist function that instantiates and returns a Wasmer `int32` value. First, we measure the baseline latency by invoking this function directly within the Go host runtime. Second, we benchmark the latency when invoking the same function from within a WebAssembly (Wasm) binary. Calling the function directly in the Go host runtime takes $\approx$ 59ns, while invoking the function from within a Wasm binary takes $\approx$ 758ns.

We then evaluate the performance of our underlying send and receive network utilities. As detailed in Table III, the underlying host networking primitives introduce an execution cost of $\approx$ 750ns under unthrottled conditions. Activating token-bucket rate limiting introduces an additional computational overhead of ≈ 3.5 μs per network invocation due to state validation and map lookups. This small sub-microsecond penalty proves negligible when weighed against typical network transfer latencies.

Table III: Operator microbenchmarks isolating runtime shaping latencies.

	W/o rate limiting (ns)	W/ rate limiting (ns)
Send time	740 ± 6	4324 ± 34
Receive time	759 ± 9	4283 ± 43

Probe tagging. We benchmark the TC egress hook under two conditions—*bypass* (no socket mark, forwarded immediately) and *tagging* (SipHash-2-4 plus tag emission)—driving it with `BPF_PROG_TEST_RUN` and reporting kernel-measured execution time over several hundred runs per configuration (Table IV). Bypass costs ≈18 ns regardless of packet size or address family. Tagging adds ≈141 ns for IPv4 and ≈294 ns for IPv6—about twice as much, since IPv6 inserts a Destination Options header rather than overwriting the Identification field in place—and is independent of packet size, confirming that the cost is dominated by the fixed 64-byte hash. The resulting ≈160 ns total per IPv4 packet lets a single core sustain ≈6.25 Mpkt/s, exceeding 10 Gbps at 256-byte packets; the IPv6 path sustains roughly half.

Table IV: eBPF tagging overhead in under 95% confidence interval.

	IPv4		IPv6	
Size (B)	Bypass	Tagging	Bypass	Tagging
64	17.6 ± 0.3	159.1 ± 0.8	17.5 ± 0.3	311.5 ± 1.5
256	17.8 ± 0.3	158.6 ± 0.9	17.5 ± 0.3	313.0 ± 1.5
1500	17.9 ± 0.4	158.0 ± 0.9	17.6 ± 0.4	311.3 ± 1.5

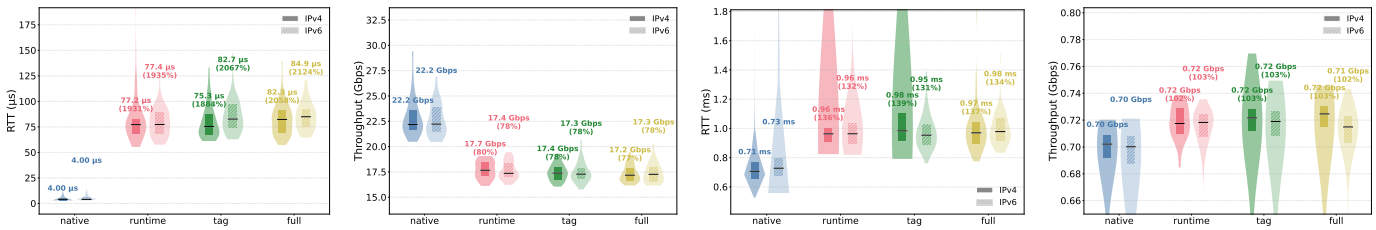
Probe Validation. Our Go verifier caches the epoch chain key and per-measurement authentication keys, invalidating them at each epoch. This amortizes the HKDF-SHA-256 derivation cost (≈1,840 ns), leaving only $2N_m$ SipHash-2-4 evaluations (≈395 ns each) per packet in steady state. It sustains ≈1.9 Mpkt/s on a single core, independent of chain depth, so longer key-chain lifetimes incur no additional verification cost.

C. Measurement Fidelity

We quantify how our instrumentation perturbs measurements using four execution conditions: *native* (the standard tool run directly), *runtime* (the equivalent Wasm probe with no instrumentation), *tag* (Wasm plus eBPF packet tagging), and *full* (tagging and rate limiting). *native* vs. *runtime* isolates the cost of Wasm sandboxing; *runtime* vs. *tag/full* isolates our instrumentation. Both probes—an ICMP ping and a TCP throughput probe—are written in C and compiled to Wasm.

We evaluate IPv4 and IPv6 on two environments: the local loopback interface, a stable, low-noise baseline that magnifies host-side costs, and the public Internet, reflecting realistic conditions. Public RTT targets four well-known DNS resolvers (we report Quad9); public throughput uses a public `iperf3` server in Amsterdam. As tagging acts on individual wire packets, we disable segmentation offload for the public throughput runs, which lowers absolute throughput uniformly across conditions and hence does not affect their comparison. Each condition is measured over 60 rounds (30 for loopback throughput), issuing one probe per condition per round with a brief inter-round cooldown; throughput runs last 5 s. Figure 4 reports medians and interquartile ranges for both families.

The families differ only in how the tag is carried. IPv4 reuses the 16-bit IP-ID field and adds no bytes; for IPv6 we



(a) RTT on loopback. (b) Throughput on loopback. (c) RTT on Quad9 (9.9.9.9). (d) Throughput on public Internet.

Figure 4: Measurement fidelity results on local loopback interface and on public Internet.

insert an 8-byte Destination Options extension header (a 2-byte tag plus framing). For bulk TCP these bytes are drawn from the segment payload, an analytically bounded ≈0.6% of goodput that lies well below our measurement resolution. One caveat is specific to IPv6: the extension header must survive the path. Every tested resolver accepts it except Google, which silently drops packets carrying extension headers [10].

1) Loopback Interface: On loopback, native tools are far faster than any Wasm condition (Figs. 4a and 4b). The gap stems from the sandbox: each probe crosses the Wasm–host boundary several times (socket open, send, receive, and timestamps) and reaches the network through Go's `net` stack rather than direct syscalls. For RTT this fixed per-call cost dwarfs the sub-microsecond native latency; for throughput it appears as a sub-microsecond per-packet cost that bounds the achievable rate. At 1,500-byte packets, a 1μs per-packet processing delay caps effective throughput at $1{,}500 \times 8/1\,\mu\text{s} \approx 12$ Gbps. In both cases the three operator conditions are statistically indistinguishable and behave identically across IPv4 and IPv6, showing that tagging and rate limiting add negligibly on top of the runtime itself.

2) Public Internet: At Internet scale, network latency dominates the sandbox overhead (Figs. 4c and 4d). RTT increases by at most a few percent relative to native execution, with no distinguishable difference among the operator configurations, indicating negligible latency overhead from eBPF tagging and rate limiting. Throughput similarly shows no systematic difference between Wasm and native execution beyond normal run-to-run variation in wide-area TCP. IPv4 and IPv6 exhibit comparable performance throughout.

D. Probe Survivability

We evaluate whether attribution tags survive network traversal without modification. We use self-describing probes whose payload echoes the tag encoded in the header, and pair each tagged probe with an untagged control sent over the same path.

IPv4. Across 17 sender–receiver pairs spanning five access classes (public IP, consumer broadband, institutional VPN, cellular, and AWS, in both directions), all 6,599 delivered probes preserved the 16-bit IPID tag, with no delivery penalty relative to the controls. The only exception was cellular, where the Don't Fragment (DF) flag was cleared on every probe while the tag remained intact. Because RFC 6864 permits arbitrary IPID values only for DF-set datagrams, this behavior is a deployment caveat for cellular networks rather than a failure of the attribution mechanism.

IPv6. The 8-byte Destination Options header was never rewritten: all 168 observed tags remained intact. However, approximately one-third of the 28 destinations dropped packets carrying the header altogether. This is consistent with the known deployment challenges of IPv6 extension headers; RFC 7872 reports a 10.9% drop rate for such packets among Alexa Top-1M web servers [10]. Our higher rate likely reflects the smaller sample and the use of a single vantage point.

VII. Discussion

A. Limitations

Our implementation has two limitations. First, IPv4 attribution reuses the IPID field, which has no defined meaning for atomic datagrams under RFC 6864 [27]. We therefore set DF and disable generic segmentation offload. IPv6 instead uses a Destination Options header, which avoids reusing IPID but is dropped on some paths. Second, the SipHash tag covers only the first 64 bytes of a probe, leaving the remaining payload unauthenticated. An attacker can modify this tail without invalidating the tag, but cannot modify the authenticated prefix, which includes the IP header. Assuming the prefix is unique per probe, such a modified probe is detected as a duplicate using the same mechanism as replay detection (Section V-B4). We enforce prefix uniqueness in IPv6 using a 16-bit nonce stored in the spare extension-header space; IPv4 has no equivalent space and instead relies on naturally varying fields, such as the TCP sequence number.

B. Trust Assumptions and Failure Modes

Our centralized design introduces the coordinator as a potential point of failure. However, neither coordinator failure nor compromise breaks probe-to-operator accountability. If the coordinator is unavailable, it admits no new measurements, while in-flight measurements continue under their existing allocations. The per-destination rate limit remains enforced (Section V-A), and probes remain verifiable after recovery using only the operators' published key chains. If the coordinator is compromised, it can violate global rate limits but cannot bypass local limits or forge accountability tags for honest operators, since it has access only to disclosed operator keys and cannot forge their signed key chains. A compromised operator can generate unattributed traffic or misattribute probes

within its own measurements, but cannot generate traffic that verifies as originating from another operator.

C. Out-of-Band Accountability

Although we implement probe accountability using in-band attribution, out-of-band approaches are also feasible. An operator could use cryptographic accumulators to produce definitive proofs of transmitted probes, or Bloom filters for probabilistic proofs over large probe volumes. However, both require persistent per-probe storage and more expensive verification operations. We consider in-band attribution with delayed key disclosure to be the more practical trade-off between overhead and verifiability.

D. Per-Target vs. Per-Source Rate Limiting

Per-target rate limiting is essential for bounding aggregate traffic toward a destination, but introduces coordination overhead that grows with the number of distinct targets. Per-source limiting avoids this cost entirely, since each vantage point enforces its own budget independently, but cannot constrain aggregate traffic from multiple vantage points toward a single target. Our globally coordinated per-target mechanism addresses the harder case: measurements with high throughput demands or targeting sensitive destinations. A hybrid deployment is straightforward—low-intensity measurements can operate under lightweight per-source limits, while only measurements exceeding a threshold or targeting policy-sensitive prefixes are subject to coordinated per-target enforcement, reducing the coordination burden in the common case.

VIII. Related Work

Distributed active measurement platforms. Existing platforms occupy two poles of the flexibility–openness spectrum. PlanetLab [4], M-Lab [2], [29], NLNOG RING [3], PacketLab [17] offer expressive interfaces but admit only vetted researchers. RIPE Atlas [5] and CAIDA Ark [1] accept public use but restrict users to a fixed set of measurement primitives. Scriptroute [24] occupies a middle ground via script-based measurements regulated by static traffic policies, but its per-vantage-point enforcement does not bound aggregate traffic, and its policy-based approach is brittle against evolving abuse patterns. TRACER enforces per-target rate limiting and takes a complementary way to regulate traffic: the malicious traffic will be attributable to the originating users.

Serverless computation platform such as AWS Lambda [7] offers on-demand, auto-scaling, multi-region code execution, but is not well suited as a measurement platform: it lacks raw-socket privileges for protocol-based probes (e.g., ICMP, UDP), caps execution at 15 minutes, and spans only $\sim$30 regions, biasing vantage-point coverage. Serverless execution can, moreover, be abused for coordinated attacks [8], yet to our knowledge, no prior work addresses regulating the traffic it generates.

Probe attribution. RFC 9511 [28] formalizes out-of-band and in-band probe attribution, but neither method is cryptographically protected: attribution can be forged and senders can deny their own probes. Fiebig [9] argues that attribution is necessary to enable target opt-out, motivating our design direction. Operational platforms such as RIPE Atlas identify probes implicitly via registered IP ranges and protocol fields, providing neither integrity nor non-repudiation. PlanetFlow [12] supports out-of-band probe attribution via a metadata database without cryptographic verifiability. TRACER instead provides probe accountability via in-band probe tagging and out-of-band key disclosure.

Lightweight source accountability in line-rate using symmetric cryptography was proposed [15], [14] but the approaches require verifier to share information before sending traffic. Our solution enables the verifier to verify the probes without sharing any information with the sender in advance.

Distributed rate limiting. Raghavan et al. [23] introduce distributed rate limiting for cloud services, proposing algorithms that enforce a global rate limit across distributed nodes while trading off accuracy against communication overhead, and demonstrating scalability to hundreds of limiters. Follow-on work has unified rate limiting with load balancing [25] and extended these ideas to programmable in-network hardware [11]. These systems target trusted participants enforcing service-level agreements over traffic aggregates. TRACER instead enforces rate limits as a security property against untrusted users, with per-destination-prefix policies and max-min fairness across concurrent measurements.

IX. Conclusion

Distributed active measurement platforms have long faced a binary choice: offer flexible interfaces only to trusted users, or provide open access with restricted expressiveness. TRACER resolves this tension by shifting the security burden from access control to accountability. Rather than restricting what untrusted users can measure, the platform makes them cryptographically accountable for the traffic they generate.

We realize this design through two complementary mechanisms: a globally coordinated rate limiter that enforces per-target bandwidth policies across vantage points, and a delayed key-disclosure protocol that provides non-repudiable probe attribution using symmetric cryptography alone. Both mechanisms impose negligible overhead while preserving measurement fidelity. Together, they demonstrate a new design paradigm for open measurement infrastructure: flexible access with enforced responsibility.

X. Acknowledgments

We would like to thank Ronaldo Alves Ferreira from Federal University of Mato Grosso do Sul for his feedback on the early draft. We also thank our shepherd and the anonymous reviewers for their insightful feedback and suggestions. We gratefully acknowledge support from ETH Zurich, and from the Zurich Information Security and Privacy Center (ZISC). This work was also supported by KREONET (Korea Research Environment Open NETwork) managed by KISTI (Korea Institute of Science and Technology Information).

REFERENCES

[1] Archipelago (Ark) Measurement Infrastructure — caida.org. https://www.caida.org/projects/ark/.

[2] Home - M-Lab — measurementlab.net. https://www.measurementlab.net/.

[3] Introduction — ring.nlnog.net. https://ring.nlnog.net/introduction/.

[4] Network Requirements | PlanetLabEurope — planet-lab.eu. https://planet-lab.eu/NodeRequirements/NetworkRequirements.

[5] RIPE Atlas - RIPE Network Coordination Centre — atlas.ripe.net. https://atlas.ripe.net/.

[6] User-defined Measurements | RIPE Atlas Documentation — atlas.ripe.net. https://atlas.ripe.net/docs/getting-started/user-defined-measurements.

[7] Amazon Web Services. AWS Lambda — serverless compute, 2026.

[8] Eyal Arazi. Radware Captcha Page — radware.com. https://www.radware.com/blog/ddos-protection/the-new-frontier-of-ddos-attacks-is-serverless/, 2023.

[9] Tobias Fiebig. Crisis, Ethics, Reliability & a measurement.network: Reflections on Active Network Measurements in Academia. In *Proceedings of the Applied Networking Research Workshop*, 2023.

[10] F. Gont, J. Linkova, T. Chown, and W. Liu. Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World. RFC 7872, June 2016.

[11] Yongchao He, Wenfei Wu, Xuemin Wen, Haifeng Li, and Yongqiang Yang. Scalable On-Switch Rate Limiters for the Cloud. In *IEEE INFOCOM*, 2021.

[12] Mark Huang, Andy Bavier, and Larry Peterson. PlanetFlow: maintaining accountability for network services. *ACM SIGOPS Operating Systems Review*, 40(1), 2006.

[13] Markus Jakobsson. Fractal hash sequence representation and traversal. *IACR Cryptol. ePrint Arch.*, 2002:1, 2002.

[14] Tiffany Hyun-Jin Kim, Cristina Basescu, Limin Jia, Soo Bum Lee, Yih-Chun Hu, and Adrian Perrig. Lightweight source authentication and path validation. In *Proceedings of ACM SIGCOMM*, 2014.

[15] Taeho Lee, Christos Pappas, David Barrera, Pawel Szalachowski, and Adrian Perrig. Source Accountability with Domain-brokered Privacy. In *Proceedings of the ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2016.

[16] Taeho Lee, Christos Pappas, Adrian Perrig, Virgil Gligor, and Yih-Chun Hu. The Case for In-Network Replay Suppression. In *Proceedings of the ACM ASIA Conference on Computer and Communications Security (AsiaCCS)*, 2017.

[17] Kirill Levchenko, Amogh Dhamdhere, Bradley Huffaker, Kc Claffy, Mark Allman, and Vern Paxson. Packetlab: a universal measurement endpoint interface. In *Proceedings of ACM Internet Measurement Conference (IMC)*, 2017.

[18] Matthew Luckie, Shivani Hariprasad, Raffaele Sommese, Brendon Jones, Ken Keys, Ricky Mok, and K. Claffy. An Integrated Active Measurement Programming Environment. In *Passive and Active Measurement*, volume 15567. 2025.

[19] Christos Pappas, Taeho Lee, Raphael M. Reischuk, Pawel Szalachowski, and Adrian Perrig. Network Transparency for Better Internet Security. *IEEE/ACM Transactions on Networking*, 27(5), 2019.

[20] Adrian Perrig and J Doug Tygar. Tesla broadcast authentication. In *Secure Broadcast Communication: In Wired and Wireless Networks*. 2003.

[21] Larry Peterson. It's Been a Fun Ride. https://systemsapproach.org/2020/03/19/its-been-a-fun-ride/, 2020.

[22] Giuseppe Raffa, Jorge Blasco, Daniel O'Keeffe, and Santanu Kumar Dash. Cloudflow: Identifying security-sensitive data flows in serverless applications. In *34th USENIX Security Symposium*, 2025.

[23] Barath Raghavan, Kashi Venkatesh Vishwanath, Sriram Ramabhadran, Ken Yocum, and Alex C. Snoeren. Cloud control with distributed rate limiting. In *Proceedings of ACM SIGCOMM*, 2007.

[24] Neil Spring. Scriptroute: A public internet measurement facility. In *Symposium on Internet Technologies and Systems (USITS)*, 2003.

[25] R. Stanojevic and R. Shorten. Load Balancing vs. Distributed Rate Limiting: An Unifying Framework for Cloud Control. In *IEEE International Conference on Communications (ICC)*, 2009.

[26] Seyedali Tabaeiaghdaei, Filippo Costa, Jonghoon Kwon, Patrick Bamert, Yih-Chun Hu, and Adrian Perrig. Debuglet: Programmable and Verifiable Inter-Domain Network Telemetry. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2024.

[27] Dr. Joseph D. Touch. Updated Specification of the IPv4 ID Field. RFC 6864, 2013.

[28] Éric Vyncke, Benoit Donnet, and Justin Iurman. Attribution of Internet Probes. RFC 9511, 2023.

[29] Berat Can Şenel, Maxime Mouchet, Justin Cappos, Olivier Fourmaux, Timur Friedman, and Rick McGeer. EdgeNet: A Multi-Tenant and Multi-Provider Edge Cloud. In *Proceedings of the International Workshop on Edge Systems, Analytics and Networking*, 2021.