ShortMAC: Efficient Data-plane Fault Localization

Xin Zhang, Zongwei Zhou, Hsu-Chun Hsiao, Tiffany Hyun-Jin Kim
Adrian Perrig and Patrick Tague
What is Fault Localization?

- **Problem definition**
  - Identify faulty links during packet forwarding

- **Attacker Model**
  - Drop, modify, misroute, or inject packets at data plane

- **Challenges**
  - Selective attack: break ping, traceroute, etc
  - High overhead
  - Slander & framing

Diagram:
- Source
- "Got it" nodes 1 to 4
- Node 3 is marked as faulty
- "Got it" node 5
- Only drop node 5’s ACKs
What is Fault Localization?

❖ Challenges (cont’d)
   ♦ Attacks against sampling
   ♦ Forgery attack: break Netflow, Bloom Filter, etc
   ♦ Natural packet loss

![Diagram](image-url)
Why is Fault Localization Important?

- The current Internet
  - Best effort, purely end-to-end

- Fault localization enables:
  - Data-plane accountability
  - Intelligent path selection
  - Linear path trial

Worst case: $3$ vs $2^3$ trials
Design Goals

❖ Security
  ❧ Against drop, modify, inject, and replay packets
  ❧ Against multiple colluding nodes

❖ Efficiency
  ❧ Low detection delay
  ❧ Low storage, communication and computation overhead

❖ Provable guarantees
  ❧ Upper bound of damage without being detected
  ❧ Lower bound of forwarding correctness if no fault detected
ShortMAC Key Insight #1

- Fault Localization $\rightarrow$ Packet authentication
  - Fault Localization $\rightarrow$ monitor packet count and content
  - W/ pkt authen, content $\rightarrow$ count
  - Only counts $\rightarrow$ small state, low bandwidth cost
ShortMAC Key Insight #2

- *Limiting* attacks instead of perfect detection
  - Detect every misbehavior? Costly! Error-prone!
  - Absorb low-impact attack: tolerance threshold
  - Trap the attacker into a *dilemma*
  - Enable probabilistic algorithms with provable bounds
ShortMAC Key Ideas

- The ShortMAC packet marking
  - Limiting instead of perfectly detecting fake packets
  - Source marks each packet with $k$ bits (with keyed PRF)

$k$-bit MAC, e.g., $k = 1$

**Diagram Explanation**

- Source marks each packet with $k$ bits (with keyed PRF).
- Forging is unlikely with a $50\%$ chance of inconsistency.
- Detectable with $k$-bit MAC, e.g., $k = 1$.

**Equations**

- $\text{Forge } m$? 50\% chance of inconsistency. Detectable!

- $\text{PRF}_{K_d}(\text{message}, \text{SN}, \text{TTL}_d)$
- $\text{PRF}_{K_2}(\text{message}, \text{SN}, \text{TTL}_2, \text{PRF})$
- $\text{PRF}_{K_1}(\text{message}, \text{SN}, \text{TTL}_1, \text{PRF}, \text{PRF})$
ShortMAC Key Ideas

- **High-level steps**
  - Each node maintains two counters (*counter only!*)
  - *Secure* reporting
  - Threshold-based detection robust to *natural errors*

More details: Onion ACK for reporting, threshold-based detection, etc.
Theoretical Bounds

- **The math**

\[ \alpha = 1 - (1 - T_{dr})^2 + \frac{\beta}{N(1-T_{dr})^d} \]

\[ \beta = \frac{T_{in}}{q} + \frac{\sqrt{(\ln \frac{2}{\delta})^2 + 8qT_{in} \ln \frac{2}{\delta} + \ln \frac{2}{\delta}}}{4q^2} \]

\[ \theta = (1 - T_{dr})^d - \frac{\beta}{N} \]

\[ N = \frac{\ln(\frac{2d}{\delta})}{2(T_{dr} - \rho)^2(1 - T_{dr})^d} \]

- **The numbers**

<table>
<thead>
<tr>
<th>Protocol</th>
<th>ShortMAC</th>
<th>PAAI-1</th>
<th>SSS</th>
<th>Sketch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay (pkt)</td>
<td>3.8×10^4</td>
<td>7.1×10^5</td>
<td>1.6×10^8</td>
<td>≈10^6</td>
</tr>
<tr>
<td>State (bytes)</td>
<td>21</td>
<td>2×10^5</td>
<td>4×10^3</td>
<td>≈500</td>
</tr>
</tbody>
</table>
Experimental Evaluation

- Average-case performance, proof of concept

- Simulation + Prototyping
  - Simulation: large-scale, security properties
  - Prototype: computational overhead

- SSF-net based simulation
  - Single 6-hop path
  - Malicious node in the middle
  - Independently dropping/injecting packets
Simulation Results

- False rates, detection delay, and comparison

- 2-bit-MAC

![Graph showing false rates and detection delay](image)
Prototyping Results

- Pure-software router prototype in Linux/Click

- Evaluation of fast path performance
  - Per-packet PRF computation
  - Different MACs with AES-ni

- Computational overhead
  - Throughput and latency
  - Linear path topology
  - Netperf benchmark
Prototyping Results

Throughput and latency
Phew... the end

- **Limiting** instead of perfectly detecting
  - Enables efficient algorithms

- **Provable security guarantee**
  - Theoretical bounds, against strong adversaries

- **High efficiency**
  - Low detection delay, router state, comm. overhead

- **Probabilistic packet authentication**
  - Building block for other applications
Thank you!
Questions?

Xin Zhang <xzhang1@cmu.edu>