LightningFilter: Traffic Filtering at 100 Gbps

Presented by:
Benjamin Rothenberger

In collaboration with:
Prof. Adrian Perrig, Juan Garcia Pardo, Dominik Roos, Jonas Gude, Pascal Sprenger, Florian Jacky
Project Goals

- High-speed packet processing requires nanosecond operations
  - Example: 64-byte packets @ 100Gbps: ~5ns processing time

- Nanosecond scale key establishment
- Nanosecond scale packet authentication

- Trivia: how “long” is a nanosecond?
  - Answer: light travels about 30cm in 1ns
Use Case:
Network Error Message Authentication

Problem: only short time frame
→ Only possible using symmetric cryptography

Solution: DRKey!
DRKey

• Novel protocol based on symmetric cryptography

• Intel AES-NI instructions enable key derivation within \(~50\) cycles
  \(\rightarrow\) Nanosecond scale!

• Key computation is up to 3 times faster than DRAM lookup!
  \(\rightarrow\) Computing the key is faster than storing it in memory!

\(\rightarrow\) Foundation for many DDOS defense mechanisms
DRKey Performance

Authentication / Signing times averaged over 100000 runs:
DRKey: 84.8 ns
Ed25519: 125.5 µs

Factor: ~ 1450x
Lightning Filter

Traffic Filtering at 100 Gbps
Overview

- **Internet**
- **Border Router**
- **Lightning Filter**
  - SCION traffic
  - Authenticated traffic
- **Standard Firewall**
  - Firewall traffic
  - Invalid traffic
  - normal traffic

- **SCiON**
1. Attack scenario
   • Attacker located anywhere in Internet → Source authentication

2. Bandwidth capacity
   • 120 Gbps traffic volume

3. Filtering based on source authentication
   • Alternate between filtering and bypass every 30s

4. Duplicate suppression
   • 80 Gbps duplicates traffic, 40 Gbps legitimate traffic
Attack Scenario: Internet Attacker

Internet

AS A

100 Mbps

120 Gbps

100 Mbps
Attack Scenario: Internet Attacker

Internet

AS A

100 Mbps

120 Gbps

100 Mbps
Questions?
Backup Slides
**DRKey Scenario**

- Communication between clients and server is *authenticated* using DRKey
- Key derivation for L2 keys is delegated to server
DRKey Exchange Demo

1. Client requests the L2 key to communicate to the server from its local CS
2. L1 key has not been prefetched $\rightarrow$ L1 key exchange
3. Server fetches the derivation secret for its delegation from CS
4. Server then derives the same L2 key locally
5. Do 100 runs and calculate average execution time
DRKey Hierarchy

• Key establishment using a multi-level key hierarchy
  
• **L0**: per-AS local secret key & per-AS public/private key pair
  
• **L1**: AS-level key establishment (typically prefetched!)
  
• **L2**: *locally* derive symmetric keys for end hosts
DRKey Key Exchange

L1 key exchange

Fetch L2 key

Locally derive L2 Key
Key Rollover

Key Rollover

Grace Period

Grace Period

Active Key $DS_t$

Fetching Key $DS_{t+3}$

Active Key $DS_{t+1}$

Fetching Key $DS_{t+4}$

Key $DS_{t+2}$

Active Key $DS_{t+2}$
Rate Limiting

I) aggregate

Used tokens in last slice

II) recompute

Refill rate

allocation for next slice

III) distribute

Data Plane

a) Packet processing