PHY Covert Channels: Can you see the Idles?

Ki Suh Lee, Han Wang, Hakim Weatherspoon

—

Presentation by Rafał Sadziak

Covert channels

- channels that are not intended for information transfer, but can leak sensitive information
- provide the ability to hide the transmission of data within established network protocols, thus hiding their existence

Network covert channels

Covert channels over Internet

Sender:

- Has access to some part of network stack (network interface, kernel network stack or user application)
- Can use packets from other applications or generate its own

Adversary:

- Wants to detect and prevent covert channels
- Passive: Monitors packet information
- Active: employs network appliances such as jammers

Network covert channels

Storage channels

Sender:

- Changes values of packets to encode messages
- Uses unused bits or fields of protocol header like:
  - IP Identification field
  - IP Fragment Offset
  - TCP Sequence Number
  - TCP timestamps

Adversary:

- Can easily monitor specific fields of packet headers for detection
- Can sanitize fields for prevention

Network covert channels

Timing channels

Sender:

- Controls timing of transmission of packets to deliver messages
- Can mimic patterns of legitimate traffic (i.i.d random interpacket delays, spreading codes etc.)

Adversary:

- Can analyze traffic with statistical tests to detect covert channel

Physical layer (PHY) covert channels

Storage channel

Pros:

- Can use PHY special characters that are discarded before going to higher layers making it impossible to detect from kernel or application layers
- Transport of messages with high bandwidth

Cons:

- Will work only on one hop
- Unless sender has a whole path of compromised switches and routers that can forward hidden messages

Chupja - PHY timing channel

Chupja (`spy` in Korean) is faster than priori art - 10-100 Kb/s vs 10-100 b/s

- Implemented and evaluated over 10GbE physical layer
- Every interpacket gap will be used to transport one bit of message
- Ethernet idle characters (/I/) to control size of interpacket gap at ns scale (one /I/ is 700-800 picoseconds long)
- Will work over multiple hops

Chupja

Design goal

1. High bandwidth
    a. Goal of ~100 Kb/s
2. Robustness
    a. Bit error rate (BER) $<$10%
3. Undetectability
    a. Adversary employing usual methods of packets timestamping can't detect covert channel
    b. Usual methods mean passive adversary using commodity servers with commodity network interface cards

Chupja

Encoding/Decoding

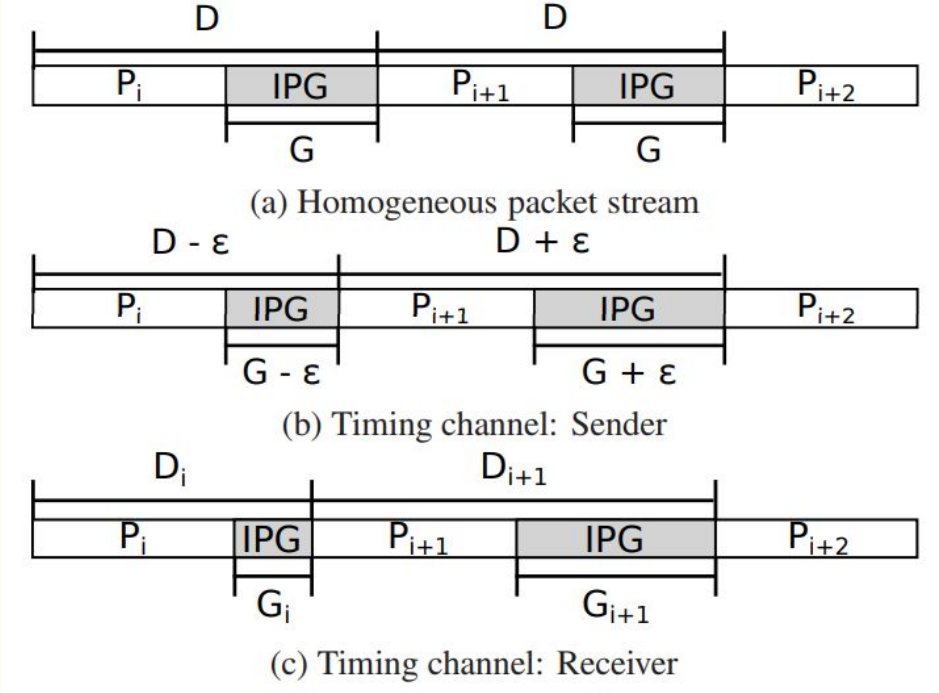
Sender and receiver share interpacket gap (G) and wait time (W) to start communication

Sender:

- $G_i = G - \varepsilon$ if $b_i = 0$
- $G_i = G + \varepsilon$ if $b_i = 1$

Receiver:

- $b'_i = 1$ if $G_i \geq G$
- $b'_i = 0$ if $G_i < G$



Evaluation setup

SoNIC (Software-defined Network Interface Card) was used to implement and evaluate Chupja

50 lines of code were added to support idle characters modulation control

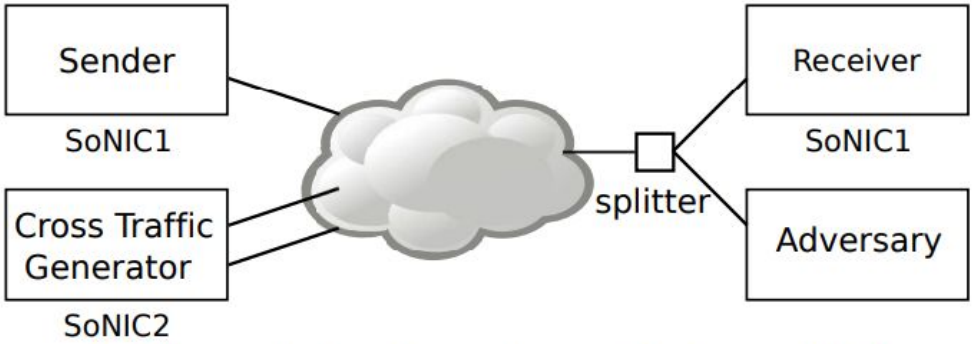


Figure 3: Network topology for evaluation. All lines are 10 gigabit fiber optic cables

Evaluation setup

Types of switches tested

	Type	40G	10G	1G	Full bandwidth	Forwarding
SW1	Core	0	8	0	160 Gbps	SF
SW2	ToR	4	48	0	1280 Gbps	CT
SW3	ToR	0	2	48	136 Gbps	SF
SW4	ToR	0	2	24	105.6 Gbps	SF

Table 1: Summary of evaluated network switches. "SF" is store-and-forward and "CT" is cut-through.

Evaluation setup

Tested packet sizes, data rates and number of /I/s

Packet size [Bytes]	Data Rate [Gbps]	Packet Rate [pps]	IPD [ns]	IPG [/I/]
1518	9	737028	1356.8	170
1518	6	491352	2035.2	1018
1518	3	245676	4070.4	3562
1518	1	81913	12211.2	13738
64	6	10416666	96.0	48
64	3	5208333	192.0	168
64	1	1736111	576.0	648

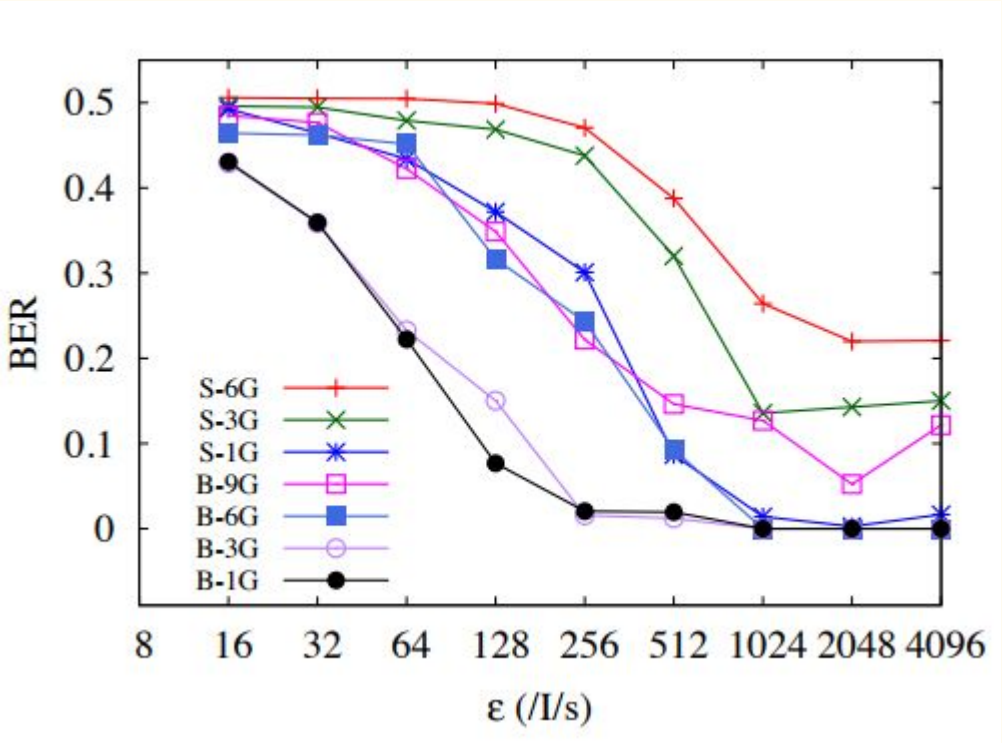
Table 2: IPD and IPG of homogeneous packet streams.

ε (/I/s)	16	32	64	128	256	512	1024	2048	4096
ns	12.8	25.6	51.2	102.4	204.8	409.6	819.2	1638.4	3276.8

Table 3: Evaluated ε values in the number of /I/s and their corresponding time values in nanosecond.

Results

Small network without cross traffic



Takeaways:

- Chupja is more efficient with bigger packets
- When there is no cross traffic modulating small number of /I/s (128) is sufficient to create a timing channel

Results

Small network with cross traffic

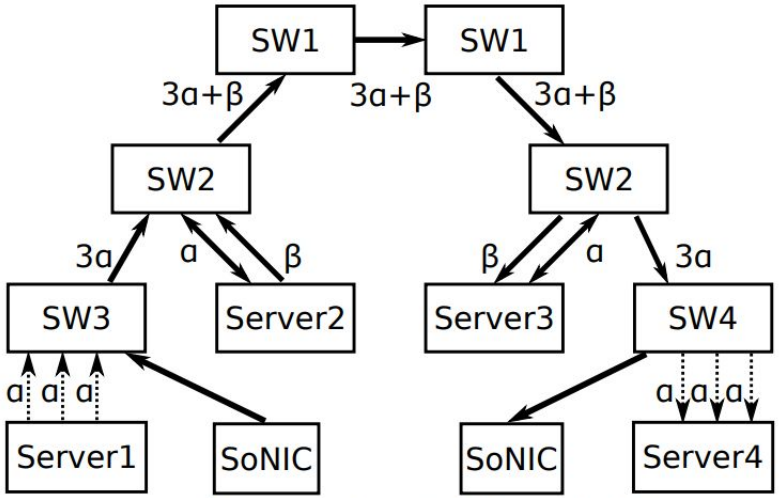
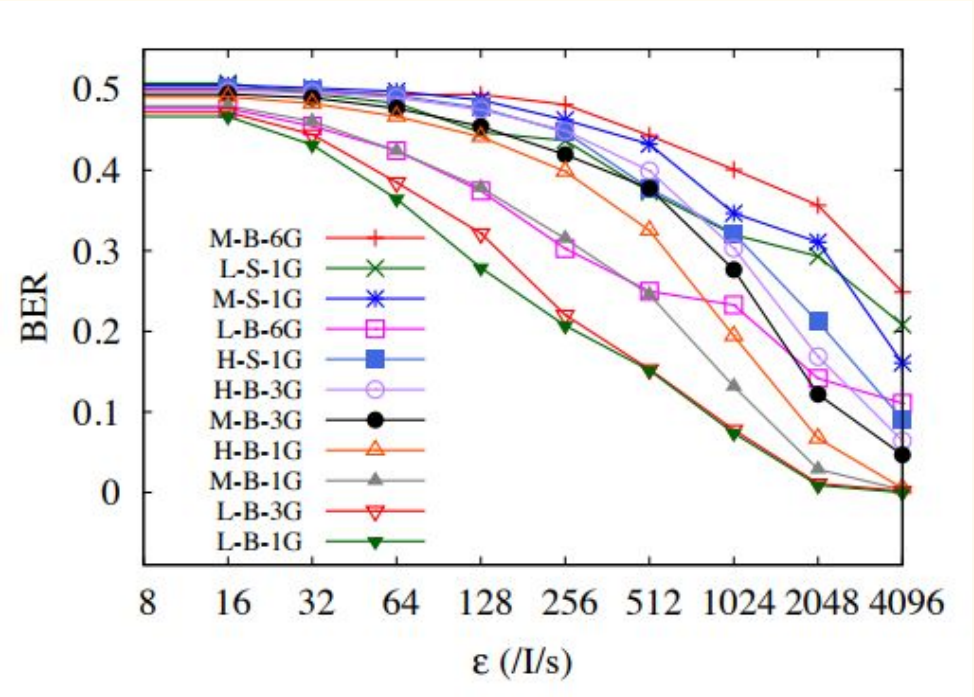


Figure 4: A small network. Thick solid lines are 10G connections while dotted lines are 1G connections.

Seven-hop timing channel with 0.154 ms round trip time delay on average

Results

Small network with cross traffic



Takeaways:

- ε needs to be bigger for same BER
- Channels at data rate higher than 6G are not efficient
- Channels work well for 1518B packet size and 1-3Gbps data rates

Results

Over National Lambda Rail

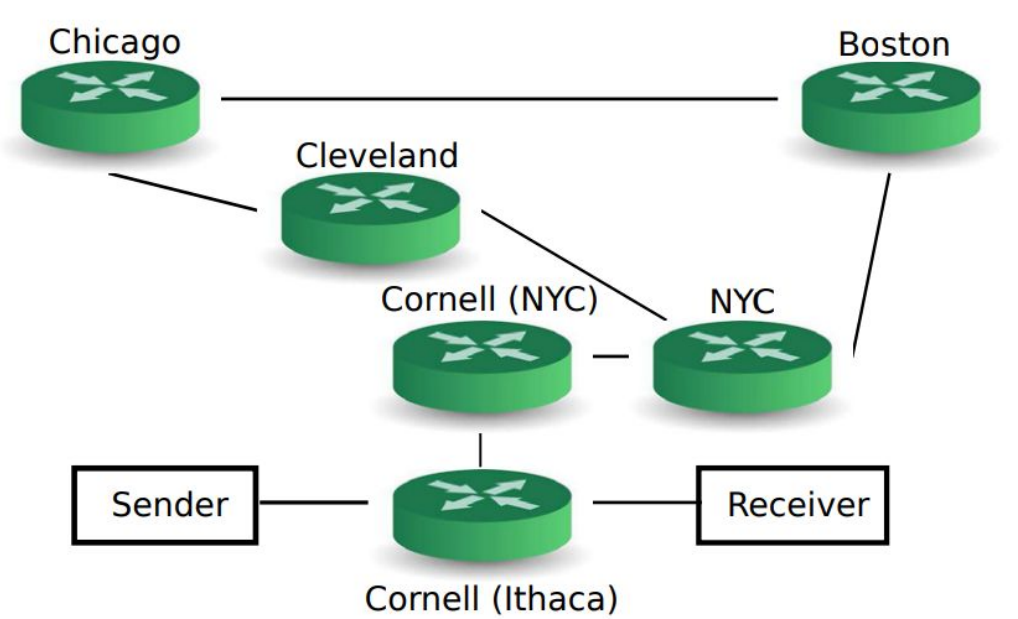


Figure 6: Our path on the National Lambda Rail

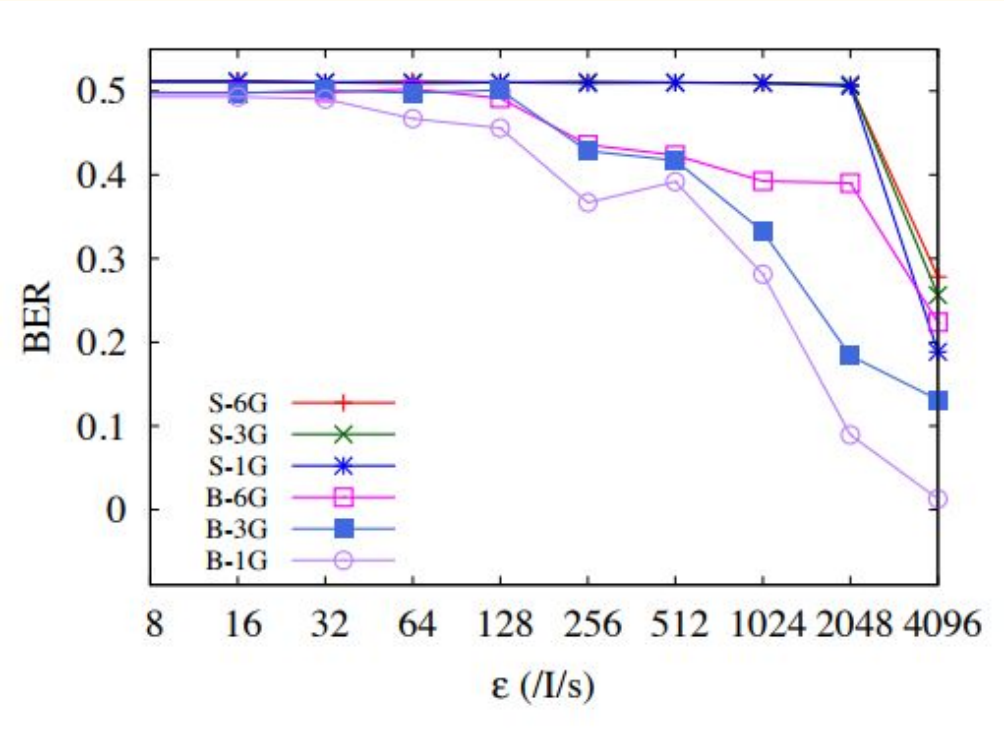
NLR is a wide-area network designed for research and has significant cross traffic

Round trip delay is 67.6 ms on average

During evaluation cross traffic was around 1~2 Gbps on many links

Results

Over National Lambda Rail



Takeaways:

- Only packet size 1518B and data rate 1Gbps are feasible but only for very large ε (>2048 /I/s = 1638.4 ns)

Sensitivity analysis

Perturbation

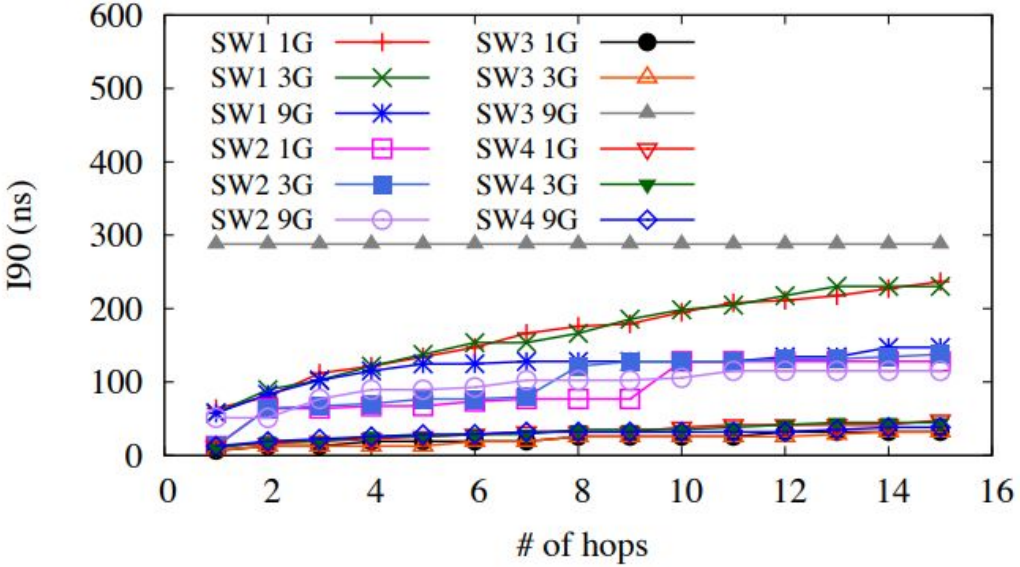


Figure 8: I_{90} comparison between various switches

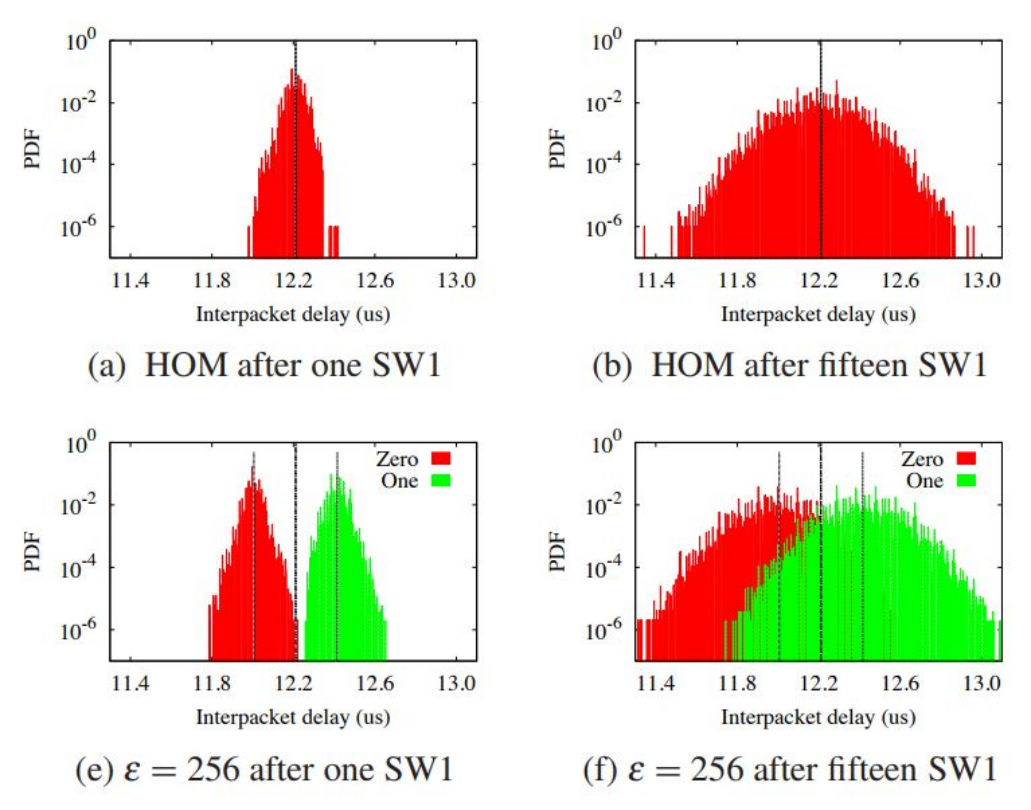
I_{90} = smallest ε that satisfies $P(\mu-\varepsilon \leq D \leq \mu+\varepsilon) \geq 0.90$

Takeaway:

- switches do not add significant perturbations to IPDs
- perturbation is not related to data rate

Sensitivity analysis

Correlation



(a) HOM after one SW1

(b) HOM after fifteen SW1

(e) $\varepsilon = 256$ after one SW1

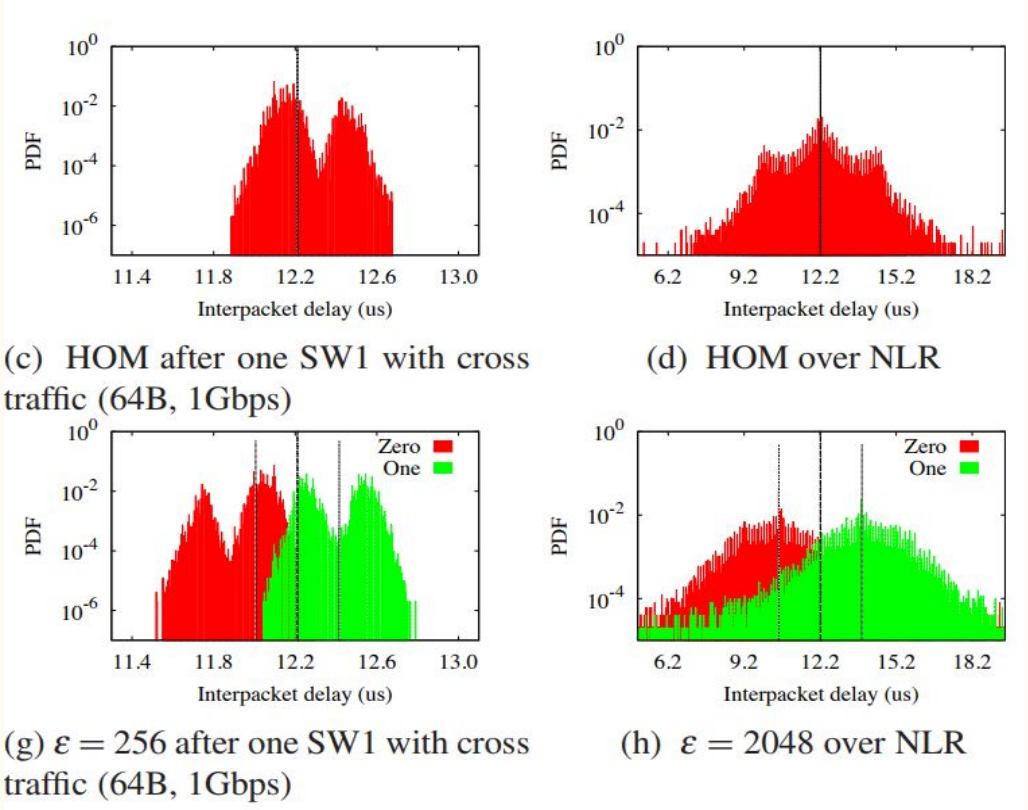
(f) $\varepsilon = 256$ after fifteen SW1

Takeaway:

- switches treat IPDs of an encoded ‘zero’ bit and those of an encoded ‘one’ bit as uncorrelated distributions even after multiple hops

Sensitivity analysis

Correlation



(c) HOM after one SW1 with cross traffic (64B, 1Gbps)

(d) HOM over NLR

(g) $\varepsilon = 256$ after one SW1 with cross traffic (64B, 1Gbps)

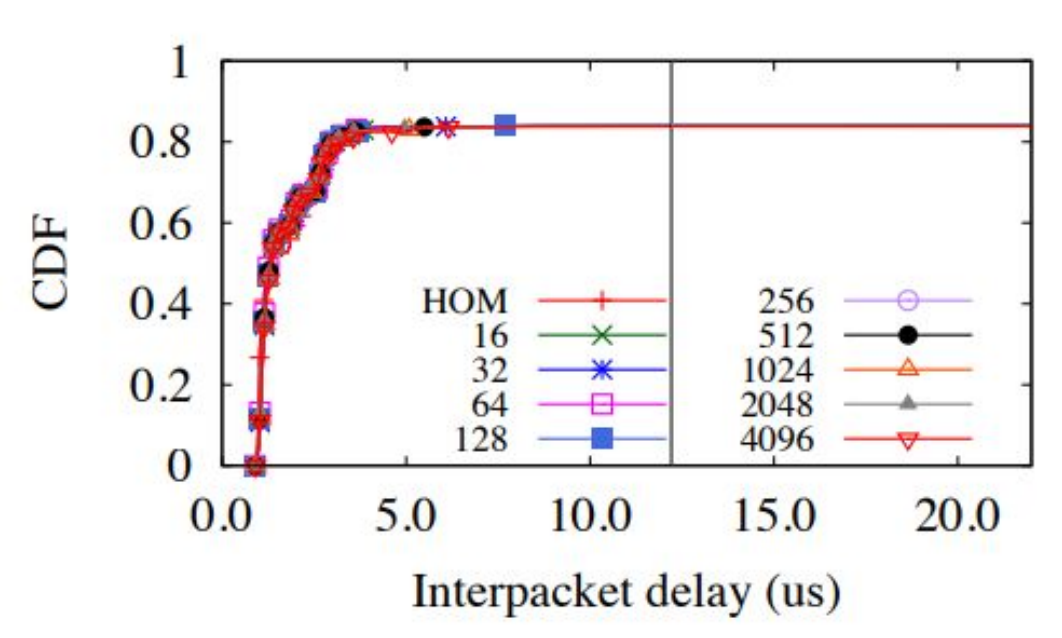
(h) $\varepsilon = 2048$ over NLR

Takeaway:

- switches treat IPDs of an encoded ‘zero’ bit and those of an encoded ‘one’ bit as uncorrelated distributions even with cross traffic

Detection

Kernel timestamping

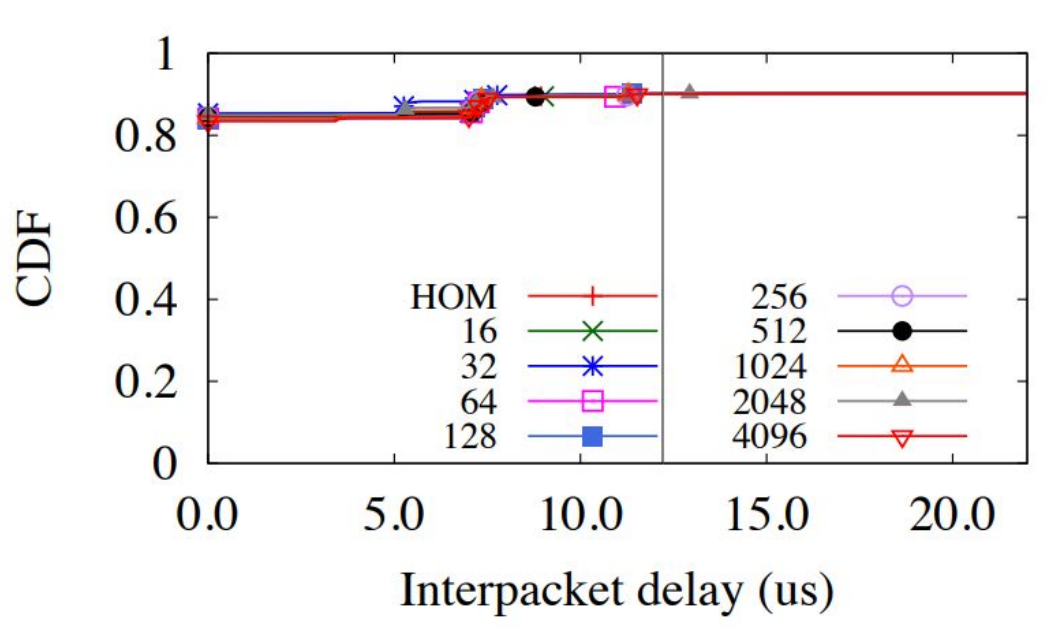


All measured kernel timestamps were nowhere near the vertical line regardless of ε values

Kernel timestamping cannot distinguish a PHY covert channel like Chupja

Detection

Zero-copy timestamping

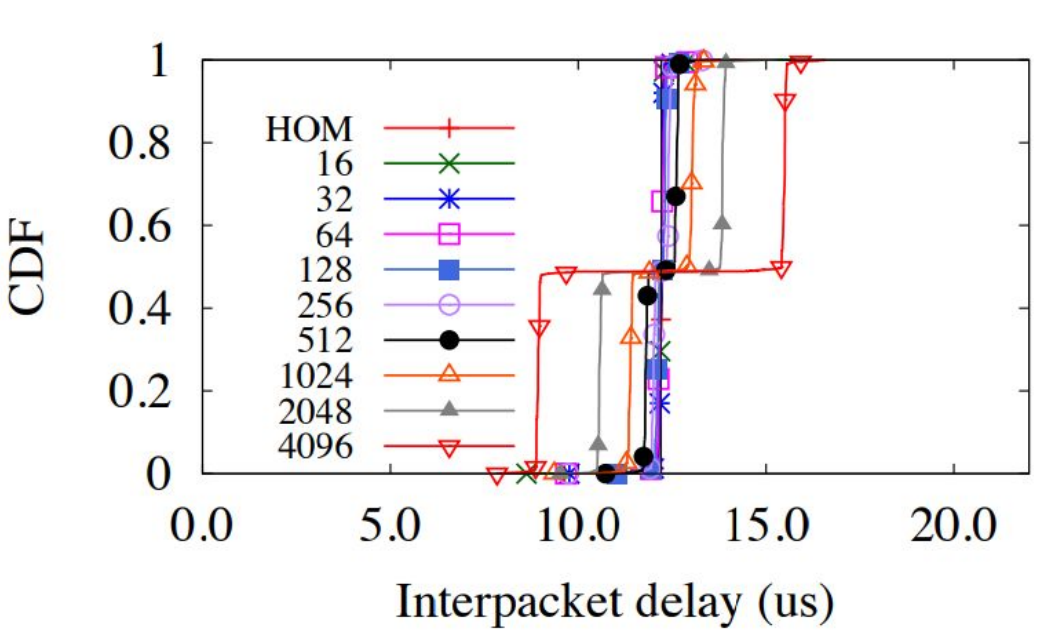


Netmap was used for zero-copy timestamping, which removes expensive memory operations and bypasses the kernel network stack

Netmap still depends on underlying system's timestamping, which is not capable of detecting Chupja

Detection

Hardware timestamping

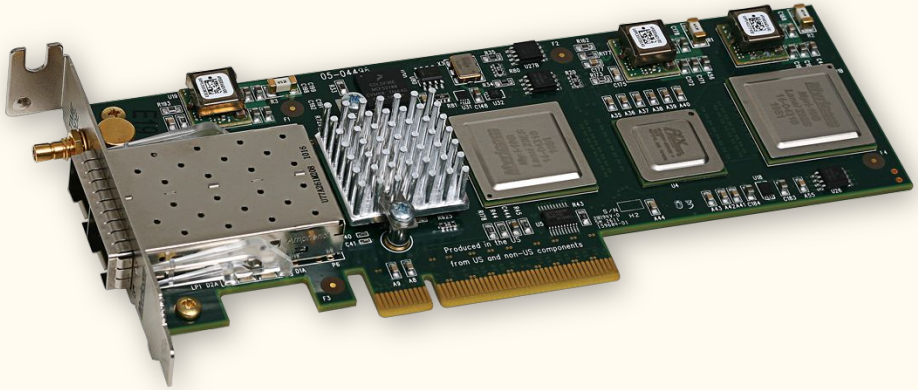


Hardware used (Sniffer 10G) demonstrates enough fidelity to detect Chupja when modulation is larger than 128 /I/s

Countermeasures

Covert timing channels implemented in the physical layer can leak secret information without being detected and as such are great threats to a system's security.

PHY-enhanced network jammers or monitoring appliances could potentially prevent or detect the existence of a covert channel.



Conclusion

Chupja PHY covert timing channel:

- Can effectively deliver 81 Kb/s with BER $<10\%$
- Is undetectable by software endhosts since they are not capable of detecting such small modulations in interpacket gaps

Future direction should focus on efficient methods to prevent or detect such covert channels

Authors' paper source:

https://www.usenix.org/node/179744

Thank you

Q&A