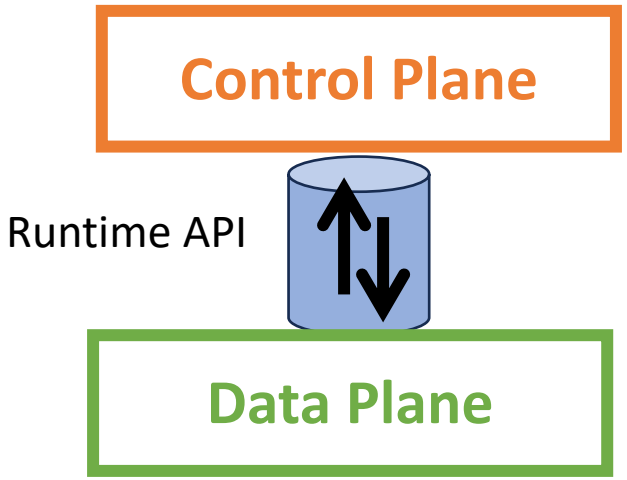
P4EAD: Securing the In-band Control Channels on Commodity Programmable Switches

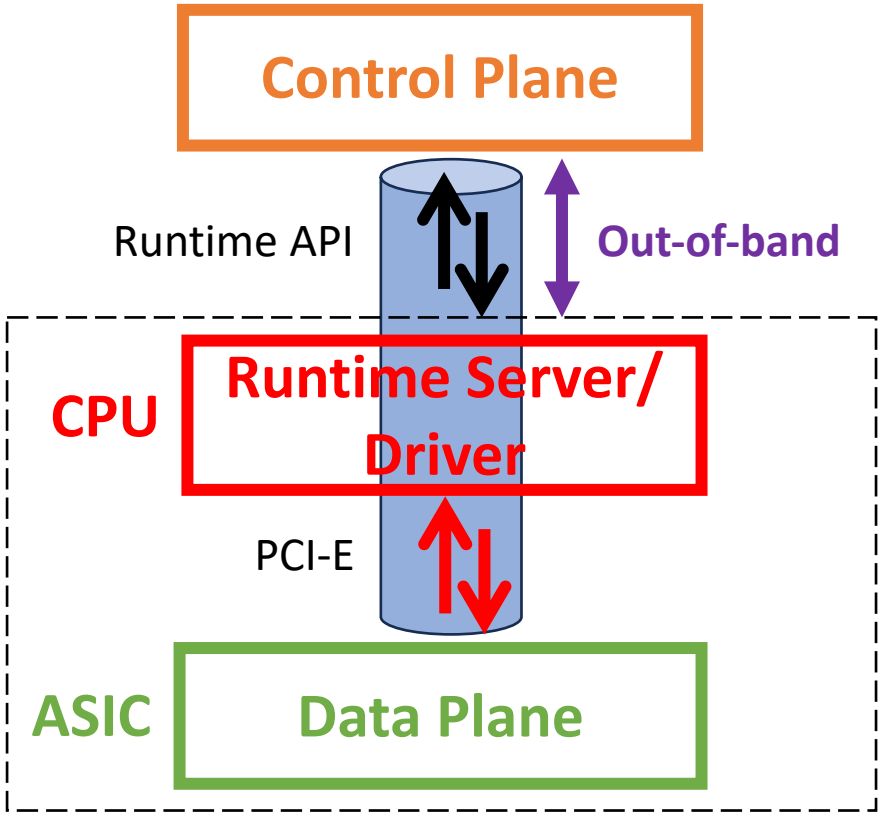
Archit Bhatnagar, Xin Zhe Khooi, Cha Hwan Song, Mun Choon Chan

National University of Singapore

What are *control channels*?



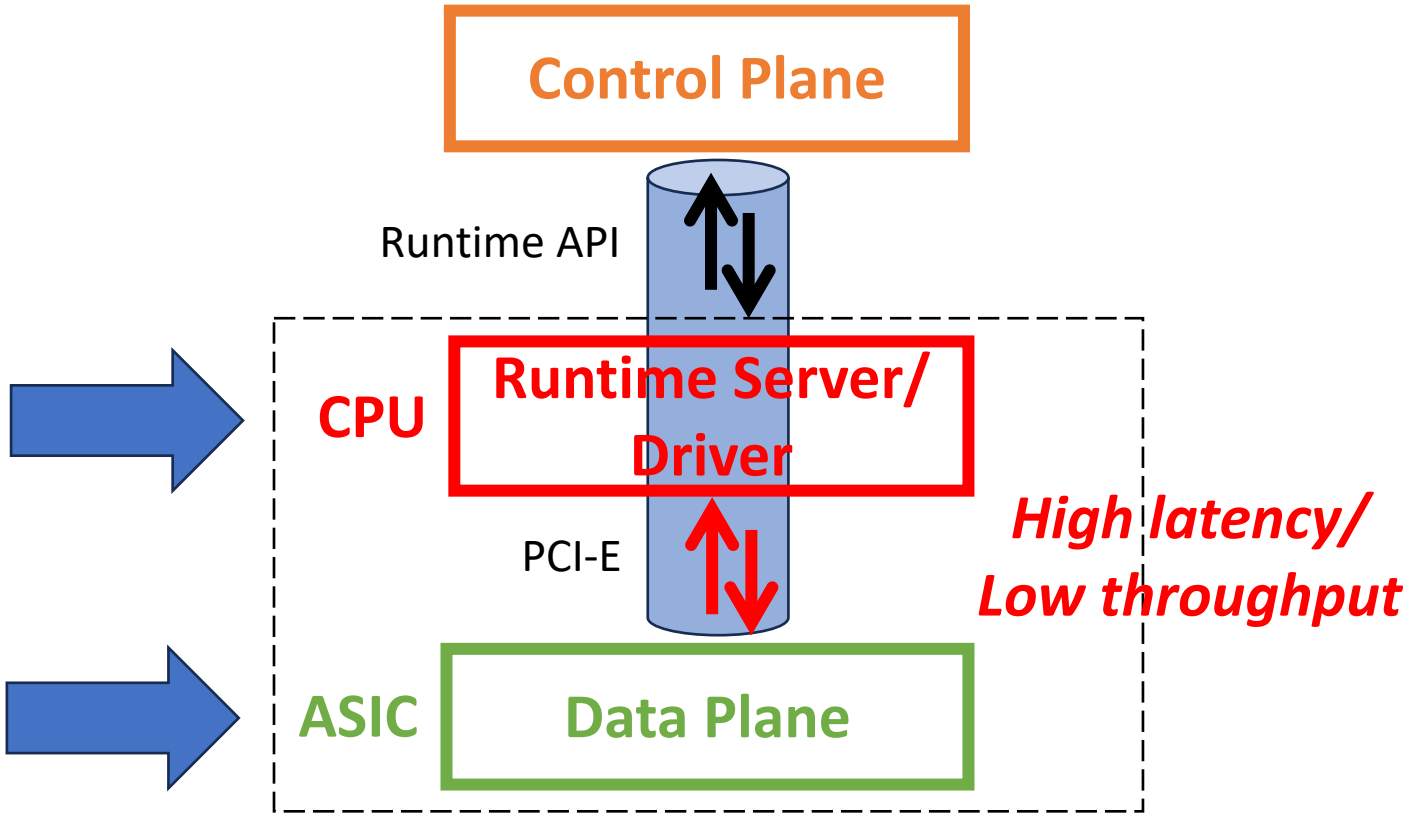
Example of Runtime APIs:
OpenFlow, P4Runtime



Existing out-of-band control channels are *slow*

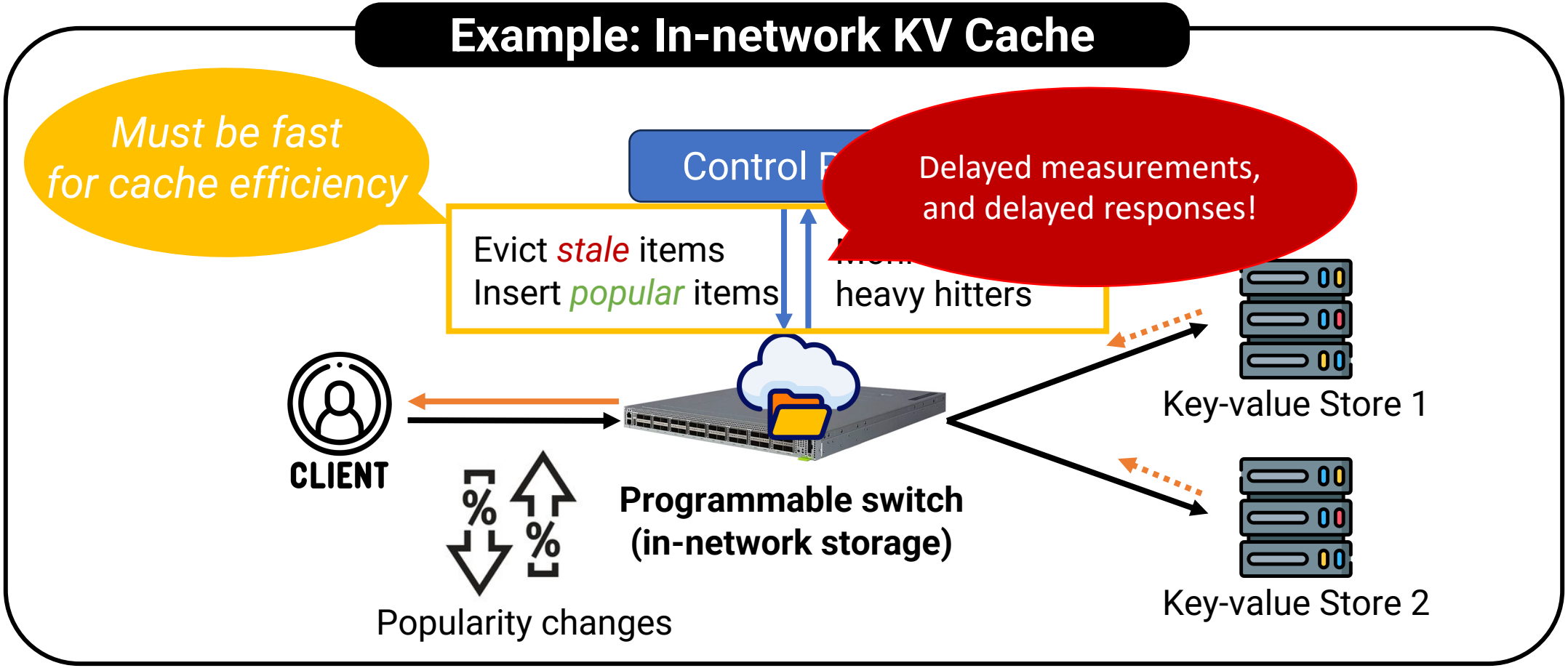
Control plane (out-of-band) read/write operation rate: **<10^5 entries/s**

Data plane packet processing rate: **>10^9 pkts/s**



Orders of magnitude mismatch in performance!

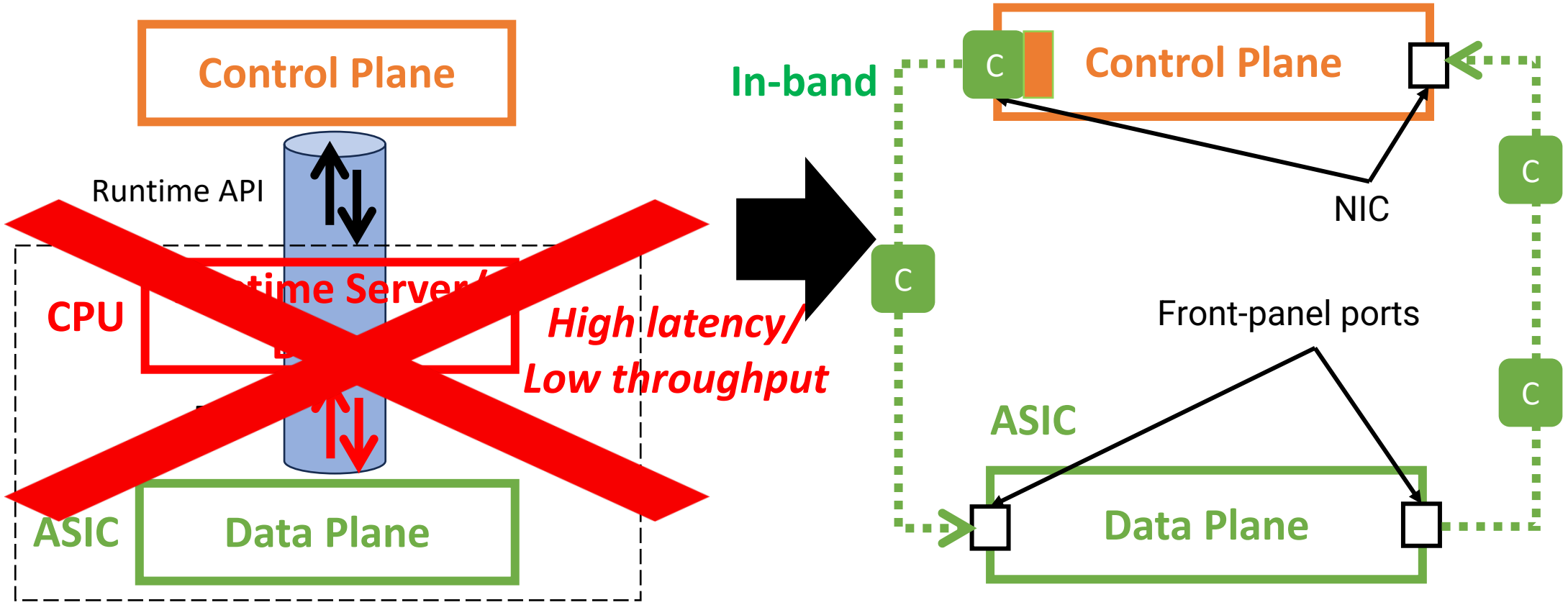
Issue with slow control channels



Delayed responses result in system performance degradation!

Emerging In-band control channels

- **Key idea**: Exploit the data plane's high-speed packet processing rates
  - Control plane uses control packets to update/ retrieve the data plane states



Revisiting Application Offloads on Programmable Switches

Cha Hwan Song[†], Xin Zhe Khooi[†], Dinil Mon Divakaran[‡], Mun Choon Chan[†]
[†]National University of Singapore, [‡]Trustwave

***Abstract*—Application offloads on modern high-speed programmable switches have been proposed in a variety of systems (e.g., key-value store systems and network middleboxes) so as to efficiently scale up the traditional server-oriented deployments. However, they largely achieve sub-optimal offloading efficiency due to the lack of (1) capability to perform control actions at sufficient rates, and (2) adaptability to workload changes.**

In this paper, we scrutinize the common stumbling blocks of existing frameworks with performance evaluations on real workloads. We present DySO (Dynamic State Offloading), a framework which enables expeditious on-demand control actions and self-tuning of management rules. Our software simulations show up to 100% performance improvement compared to existing systems for various real world traces. On top of that, we implement and evaluate DySO on a commodity programmable switch, showing two orders of magnitude faster responsiveness to sudden workload changes compared to the existing systems.

***Index Terms*—Programmable networks, P4, Framework, Application acceleration, In-network caching**

I. Introduction

The emergence of commodity programmable switches have induced a series of innovations by enabling hardware acceleration for a broad range of mission-critical data-center

Millions of Low-latency State Insertions on ASIC Switches

TOMMASO CAIAZZI, Roma Tre University, Italy and KTH Royal Institute of Technology, Sweden
MARIANO SCAZZARIELLO, KTH Royal Institute of Technology, Sweden
MARCO CHIESA, KTH Royal Institute of Technology, Sweden

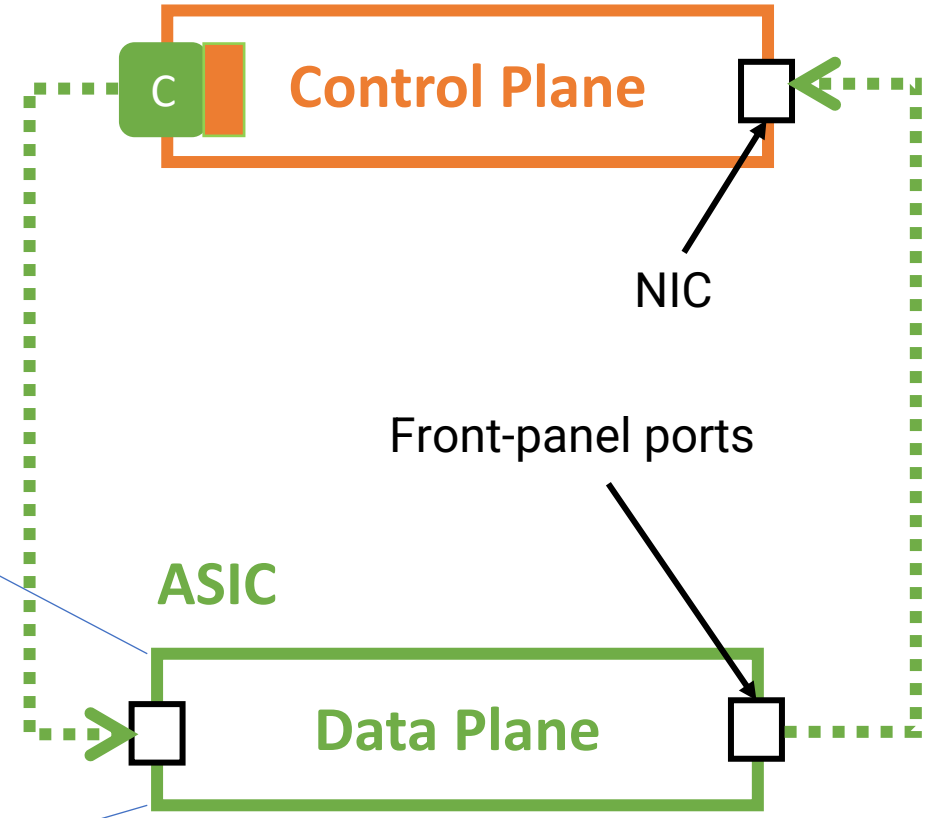
Key-value data structures are an essential component of today's stateful packet processors such as load balancers, packet schedulers, firewalls, and more. Supporting key-value data structures entirely in the data plane of an ASIC switch would result in high throughput, low-latency, and low energy consumption. Yet, today's key-value implementations on ASIC switches are ill-suited for stateful packet processing as they support only a limited amount of flow-state insertions per second into these data structures. In this paper, we present SwitchAroo, a mechanism for realizing key-value data structures on programmable ASIC switches that supports both high-frequency insertions and fast lookups *entirely* in the data-plane. We show that SwitchAroo can be realized on state-of-the-art programmable ASICs, supporting millions of flow-state insertions per second with only a limited amount of packet recirculation.

CCS Concepts: • **Networks** → **Programmable networks**; **In-network processing**; *Middle boxes / network appliances.*

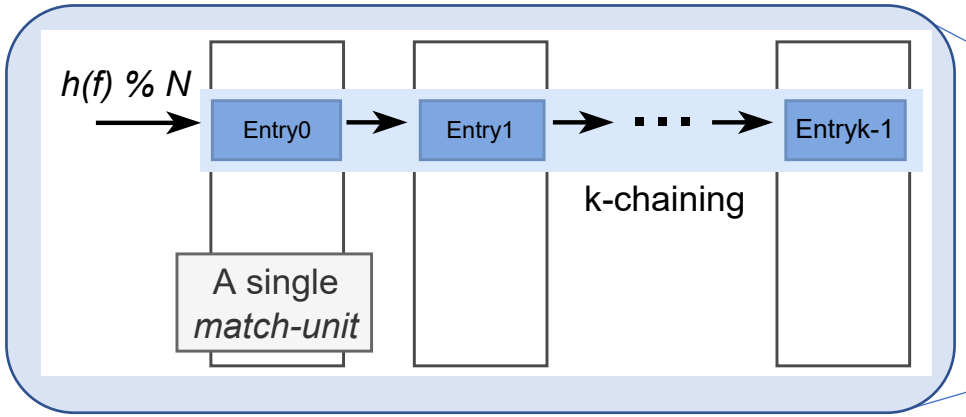
Additional Key Words and Phrases: stateful NFs, programmable switches, cuckoo hashing, ASIC switches, flowlet routing

Common designs for in-band control channels

- Substitute existing match-action tables (MAT) with hash tables constructed using register arrays
  - **Why?** Existing MATs cannot be updated through in-band control packets
- Enables updating entirely in the data plane with much faster updates (e.g., up to 100 times faster)

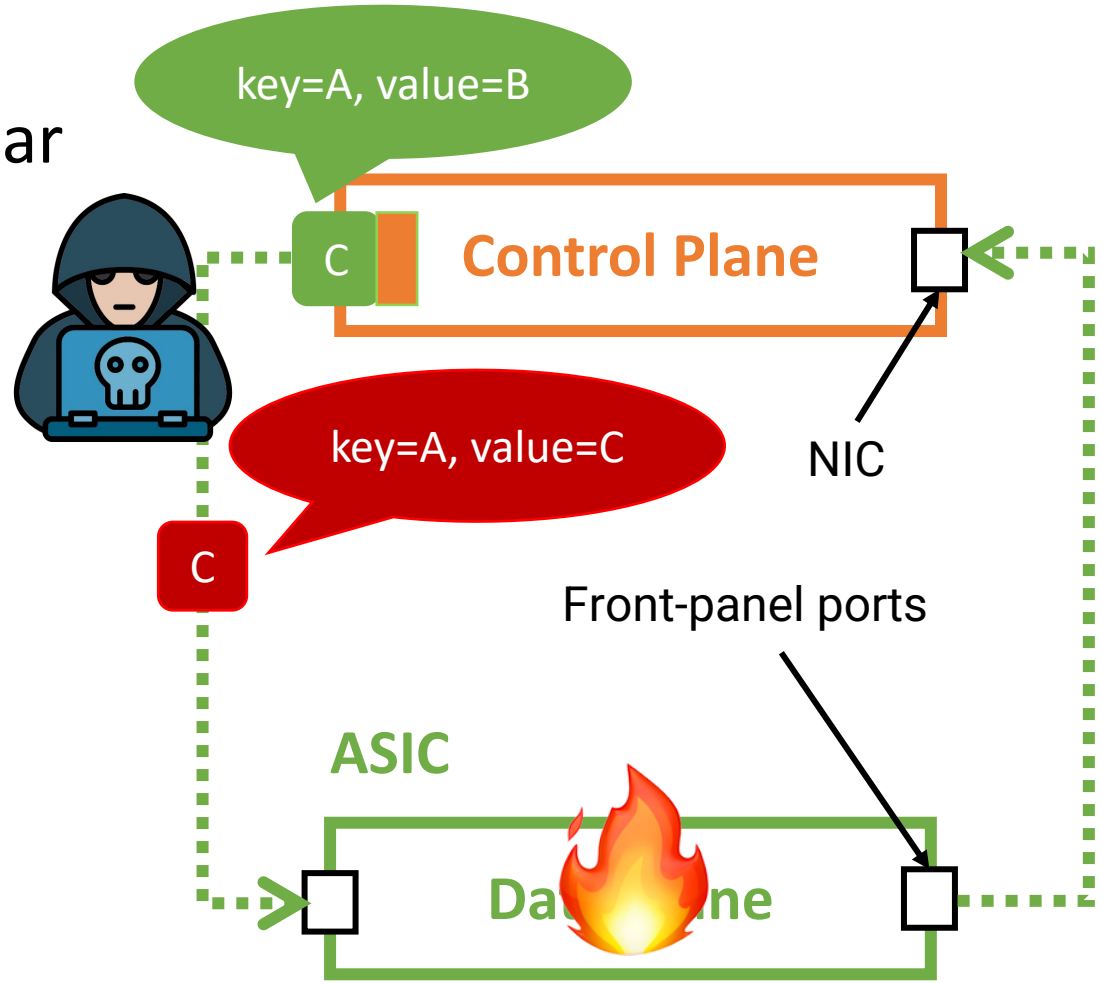


Example:



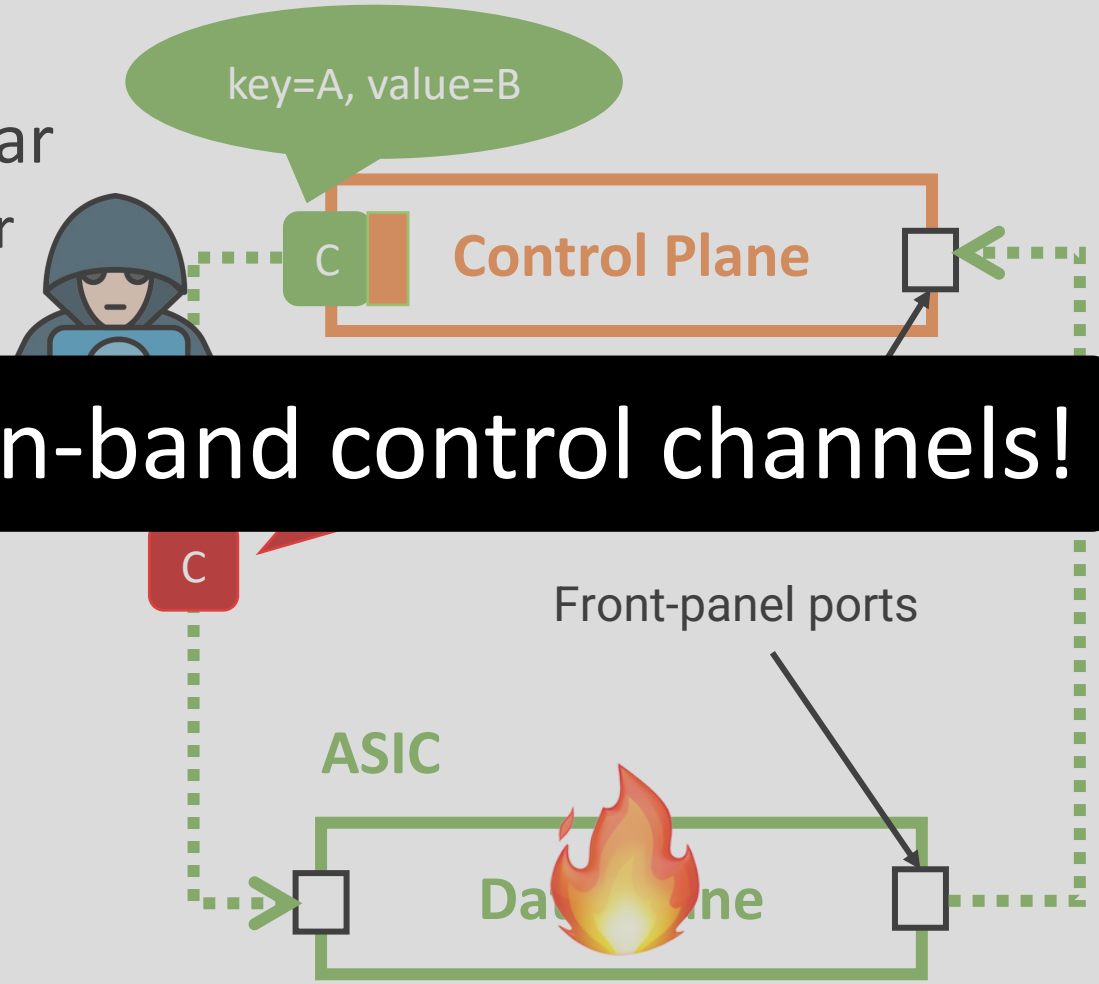
Existing in-band control channels are *vulnerable*!

- Control packets are sent in the clear
  - **Threat Model**: Can be sniffed modified, and/or replayed by potential adversaries!
- Systems are thus vulnerable to attacks like:
  - man-in-the-middle
  - denial-of-service
  - replay attacks
    - encryption is not enough



Existing in-band control channels are *vulnerable*!

- Control packets are sent in the clear
  - **Threat Model**: Can be sniffed and/or modified by potential adversaries!



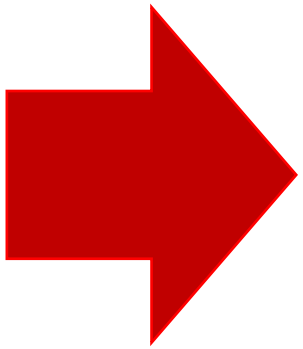
attacks like:

- man-in-the-middle
- denial-of-service
- replay attacks

Securing the in-band control channels

Requirements:

- Confidentiality
- Authenticity
- Integrity

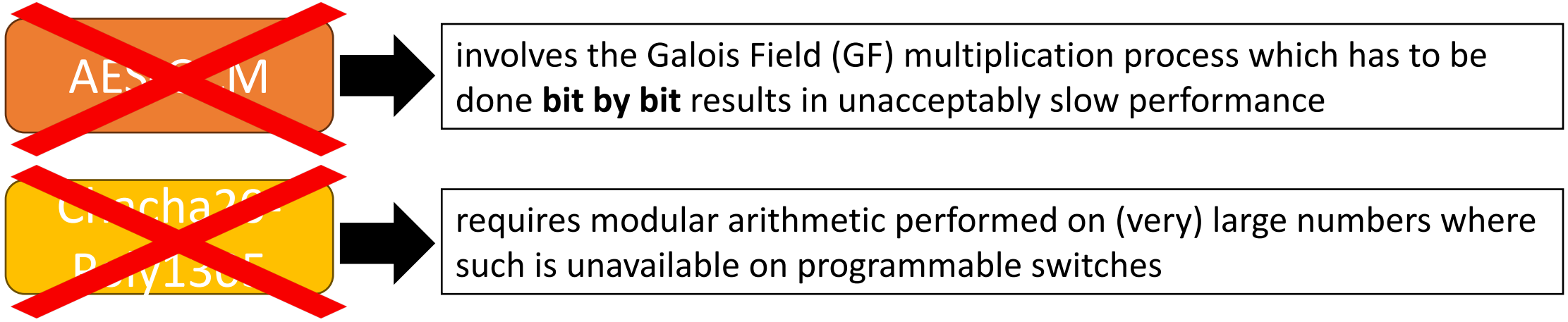


Authenticated Encryption (AE)

Can AE be done on programmable switches?

Authenticated Encryption on Programmable Switches?

- No existing AE implementation exists for programmable switches
  - Combining **P4-AES** and **SipHash** to form an AE-like scheme, it is not secure[1]!
  - It is also expensive – requires sufficient hardware resources for both
- How about implementing well-known AE schemes?



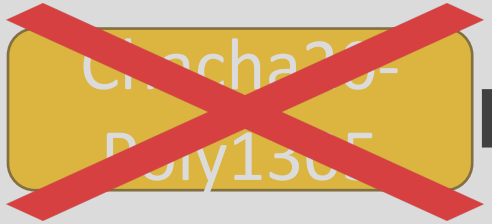
[1] T. Kohno; J. Viega & D. Whiting. NIST. Retrieved March 12, 2013. it is very easy to accidentally combine secure encryption schemes with secure MACs and still get insecure authenticated encryption schemes.

Authenticated Encryption on Programmable Switches?

- No existing AE implementation exists for programmable switches
  - Combining **P4-AES** and **SipHash** to form an AE-like scheme, it is not secure[1]!
  - It is also expensive – requires sufficient hardware resources for both
- How about implementing well-known AE schemes?

Is there any alternative?

hich has to be done **bit by bit** results in unacceptably slow performance



requires modular arithmetic performed on (very) large numbers where such is unavailable on programmable switches

[1] T. Kohno; J. Viega & D. Whiting. NIST. Retrieved March 12, 2013. it is very easy to accidentally combine secure encryption schemes with secure MACs and still get insecure authenticated encryption schemes.

Lightweight Crypto Suite: *ASCON*

UPDATES 2023

Lightweight Cryptography Standardization Process: NIST Selects Ascon

February 07, 2023

The NIST Lightweight Cryptography Team has reviewed the finalists based on their submission packages, status updates, third-party security analysis papers, and implementation and benchmarking results, as well as the feedback received during workshops and through the lwc-forum. The decision was challenging since most of the finalists exhibited performance advantages over NIST standards on various target platforms without introducing security concerns.

The team has decided to standardize the **Ascon** family for lightweight cryptography applications as it meets the needs of most use cases where lightweight cryptography is required. Congratulations to the Ascon team! NIST thanks all of the finalist teams and the community members who provided feedback that contributed to the selection.

NIST's next steps will be to:

PARENT PROJECT

See: Lightweight Cryptography

RELATED TOPICS

Security and Privacy: lightweight cryptography

Activities and Products: standards development

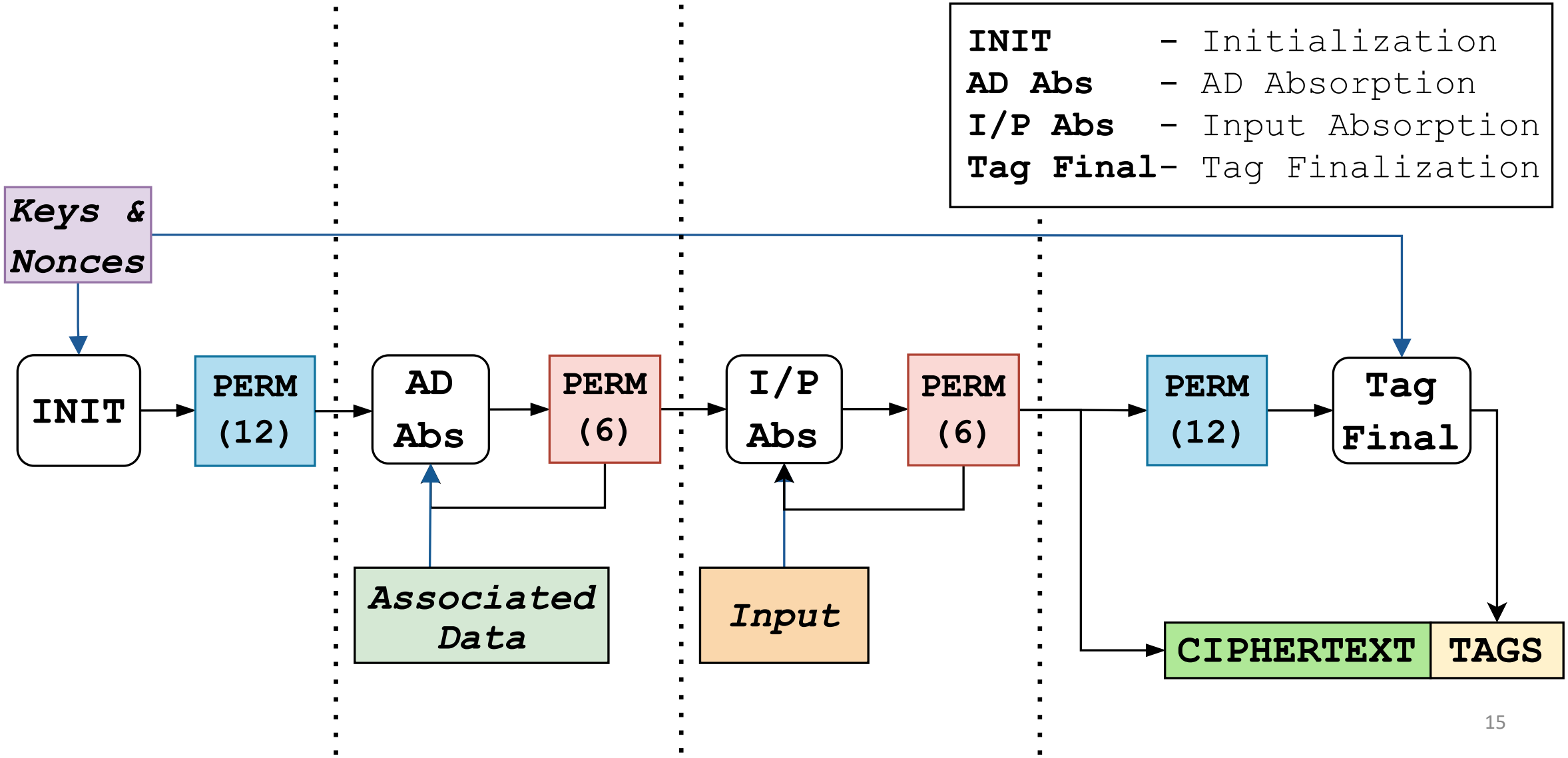
ASCON Cipher Suite

- Family of authentic
  - **ASCON-AEAD**
  - ASCON-HASH
- Lightweight, and hardv re-friendly algorithm design:
  - requires only simple binary operations like **XOR**, **ROR** and **AND**
  - computations operate over a constant 320-bit state vector

Key contribution of the paper:
P4EAD

ASCON fits the bill for programmable switches!

ASCON-AEAD


INIT - Initialization
AD Abs - AD Absorption
I/P Abs - Input Absorption
Tag Final- Tag Finalization
Keys & Nonces
INIT
PERM (12)
AD Abs
PERM (6)
I/P Abs
PERM (6)
PERM (12)
Tag Final
Associated Data
Input
CIPHERTEXT
TAGS

What are *PERM* rounds (P-RND) in ASCON?

- **PERM** rounds (**P-RND**) are used repeatedly in ASCON
- Each **P-RND** consists of
  - Round Constant Addition
  - Substitution Layer
  - Linear Diffusion Layer

Round Constant Addition

$$x_2 = x_2 \oplus c_r$$

Substitution Layer

$$x_0 = x_0 \oplus x_4$$
$$x_4 = x_4 \oplus x_3$$
$$x_2 = x_2 \oplus x_1$$

Start of Keccak box

$$t_0 = x_0 \oplus (\neg x_1 \wedge x_2)$$
$$t_1 = x_1 \oplus (\neg x_2 \wedge x_3)$$
$$t_2 = x_2 \oplus (\neg x_3 \wedge x_4)$$
$$t_3 = x_3 \oplus (\neg x_4 \wedge x_0)$$
$$t_4 = x_4 \oplus (\neg x_0 \wedge x_1)$$

End of Keccak box

$$t_1 = t_1 \oplus t_0$$
$$t_0 = t_0 \oplus t_4$$
$$t_3 = t_3 \oplus t_2$$
$$t_2 = \neg t_2$$

Linear Diffusion Layer

$$x_0 = t_0 \oplus (t_0 \ggg 19) \oplus (t_0 \ggg 28)$$
$$x_1 = t_1 \oplus (t_1 \ggg 61) \oplus (t_1 \ggg 39)$$
$$x_2 = t_2 \oplus (t_2 \ggg 1) \oplus (t_2 \ggg 6)$$
$$x_3 = t_3 \oplus (t_3 \ggg 10) \oplus (t_3 \ggg 17)$$
$$x_4 = t_4 \oplus (t_4 \ggg 7) \oplus (t_4 \ggg 41)$$

$\oplus$ XOR $\wedge$ AND $\neg$ NOT $\ggg$ ROR

Mapping *P-RNDs* on Tofino

Use MATs for to perform the constant lookup for each P-RND

1 stage

Round Constant Addition

$x_2 = x_2 \oplus c_r$

Substitution Layer

4 stage

$x_0 = x_0 \oplus x_4$
$x_4 = x_4 \oplus x_3$
$x_2 = x_2 \oplus x_1$

Start of Keccak box

$t_0 = x_0 \oplus (\neg x_1 \wedge x_2)$
$t_1 = x_1 \oplus (\neg x_2 \wedge x_3)$
$t_2 = x_2 \oplus (\neg x_3 \wedge x_4)$
$t_3 = x_3 \oplus (\neg x_4 \wedge x_0)$
$t_4 = x_4 \oplus (\neg x_0 \wedge x_1)$

End of Keccak box

$t_1 = t_1 \oplus t_0$
$t_0 = t_0 \oplus t_4$
$t_3 = t_3 \oplus t_2$
$t_2 = \neg t_2$

Do ROR using bit slicing and concatenation

Linear Diffusion Layer

3 stage

$x_0 = t_0 \oplus (t_0 \ggg 19) \oplus (t_0 \ggg 28)$
$x_1 = t_1 \oplus (t_1 \ggg 61) \oplus (t_1 \ggg 39)$
$x_2 = t_2 \oplus (t_2 \ggg 1) \oplus (t_2 \ggg 6)$
$x_3 = t_3 \oplus (t_3 \ggg 10) \oplus (t_3 \ggg 17)$
$x_4 = t_4 \oplus (t_4 \ggg 7) \oplus (t_4 \ggg 41)$

$\oplus$ XOR $\wedge$ AND $\neg$ NOT $\ggg$

Split operation into 2 parts

Combine operations using built in hash engines instead of splitting them

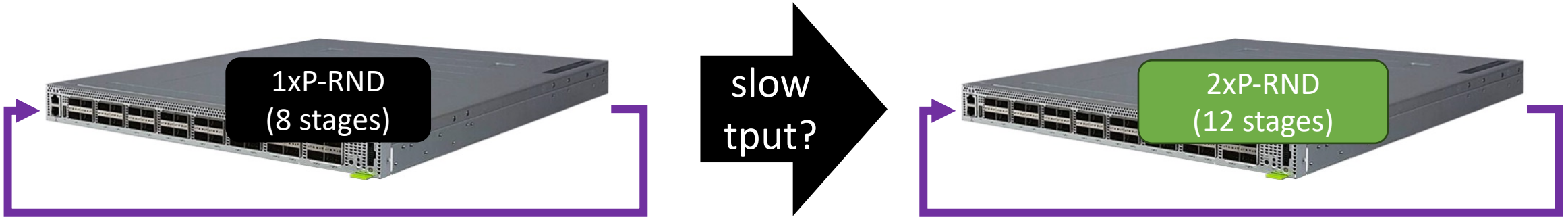
In total, **8 stages** required per P-RND!

*More details can be found in the paper and released code.

Implementing P4EAD

Context: Programmable switches have limited pipeline stages (e.g., 12)

How to implement ASCON-AEAD on Tofino? (recall 8-stage per P-RND)

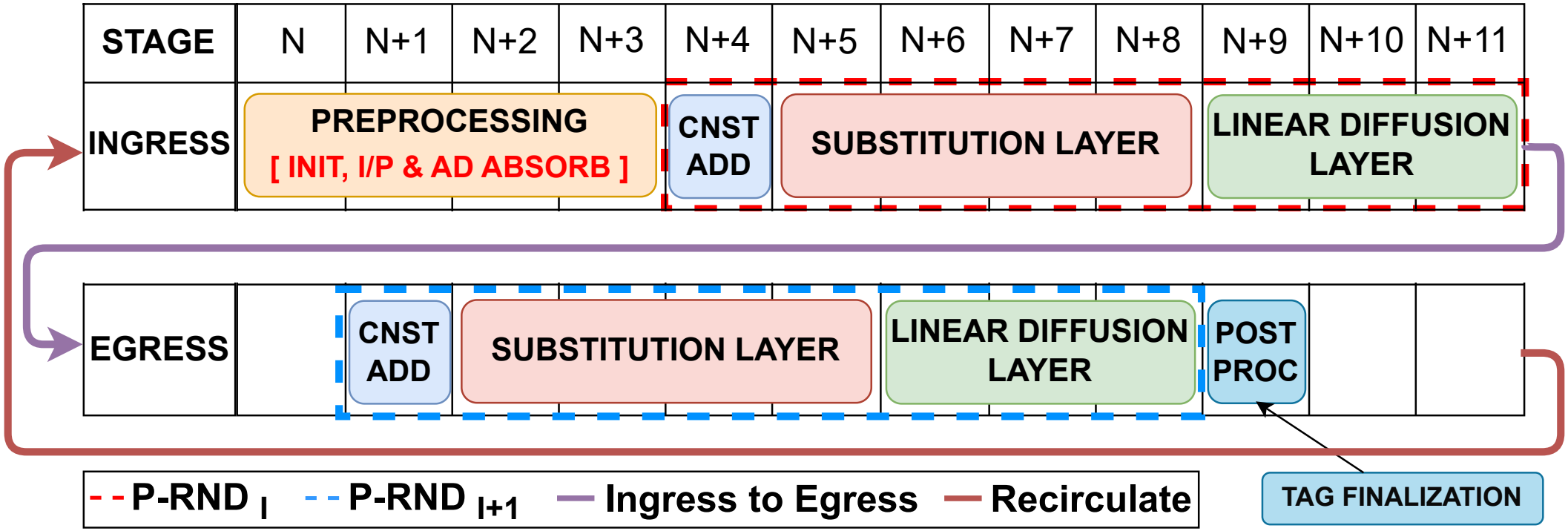


packet recirculation: add *special* metadata header to the packet to store the current P-RND number and relevant states.

double the P-RNDs: mutually exclusive operations can be parallelized while sharing the pipeline stages

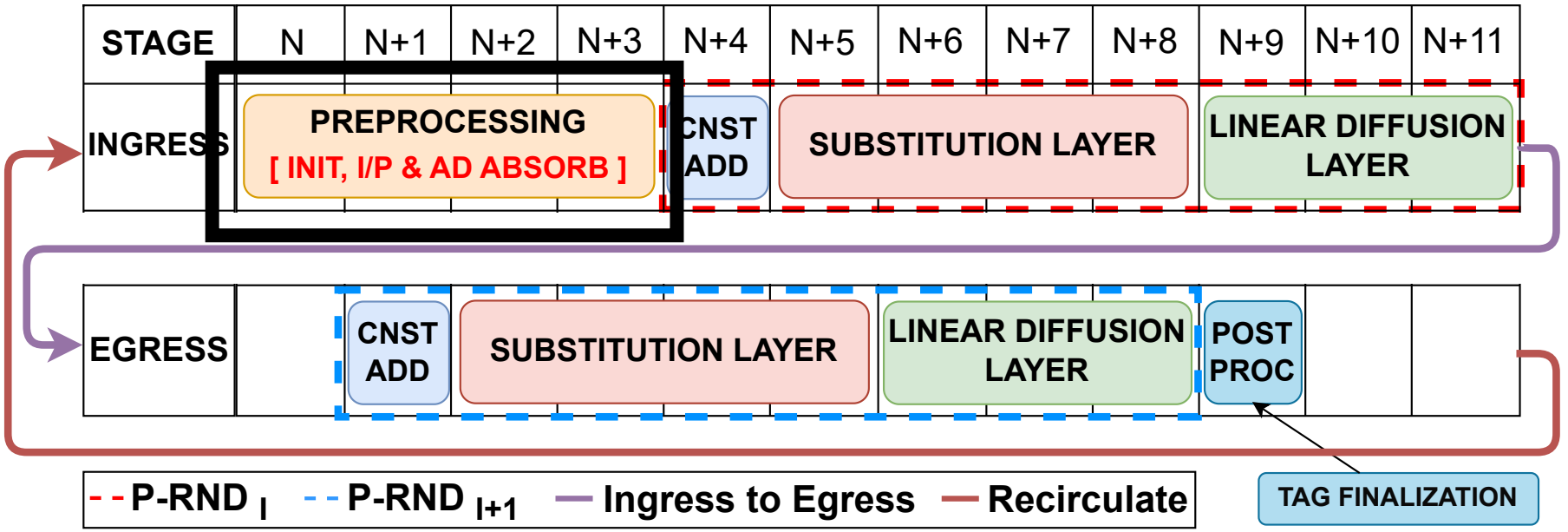
P4EAD: Pipeline Layout

- Example layout on Intel Tofino
  - 2 P-RND per pipeline pass
  - Encryption and decryption are symmetrical



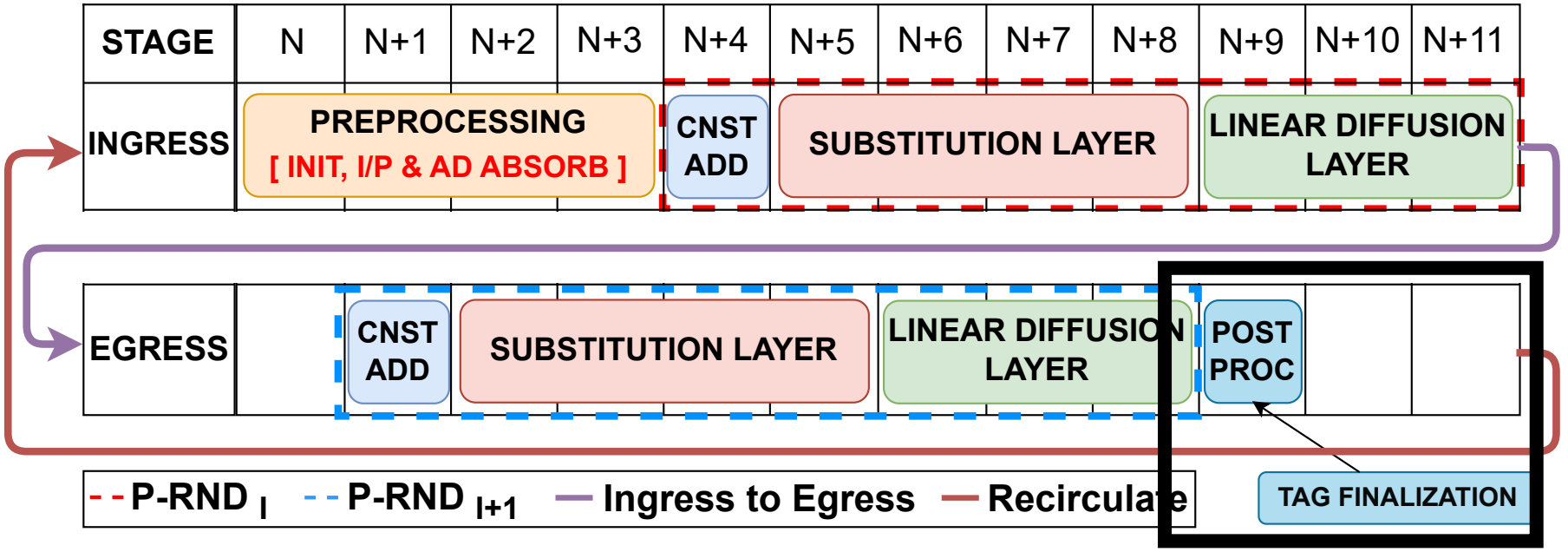
P4EAD: Pre-processing

- Pre-processing:
  - Handle 3 kinds of packets:
    - (i) newly received packets that needs to be encrypted/ decrypted, or
      (ii) recirculated packets in the middle of a PERM, or
      (iii) recirculated packets at the end of a PERM.
  - Performs state vector initialization & input absorption.



P4EAD: Post-processing

- Post-processing:
  - Perform tag finalization (encryption)
  - additionally, perform tag verification (decryption)
  - Strip off the metadata header



P4EAD: Implementation

- **Intel Tofino**: Up to 2 P-RND per pipeline pass
- **Intel Tofino2**: Up to 4 P-RND per pipeline pass

Resource	tf1_2rnd.p4	tf2_4rnd.p4
SRAM	0.8%	0.8%
Hash Bits	13.2%	15.9%
Hash Dist. Unit	55.6%	66.7%
VLIW Ins.	6.5%	6.3%
PHV	29.3%	33.9%

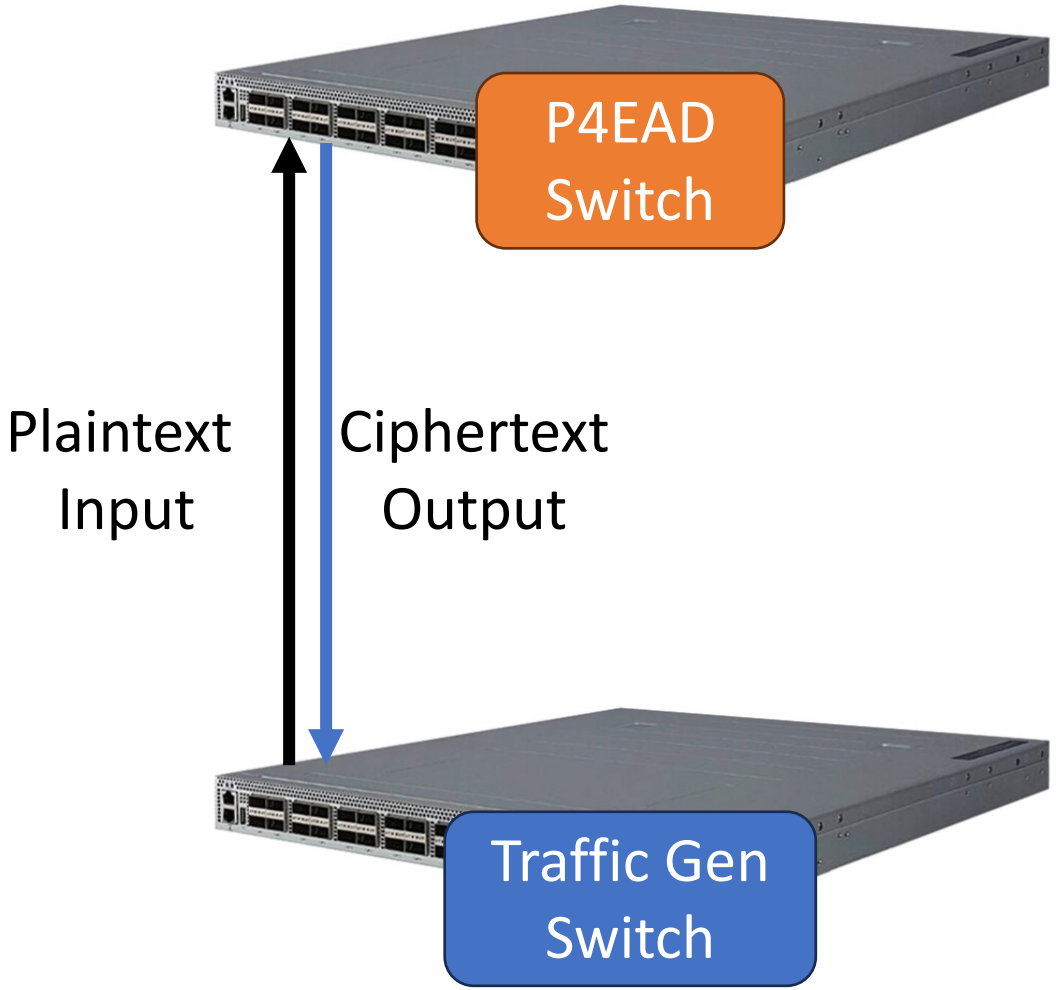
$$s = \frac{30 + 6(\lfloor \frac{p}{8} \rfloor + \lfloor \frac{q}{8} \rfloor)}{r}$$

p - length of input in bytes
q - length of associated data in bytes
r - the number of P-RNDs per pipeline pass
s - the number of recirculations needed

P4EAD is publicly available!

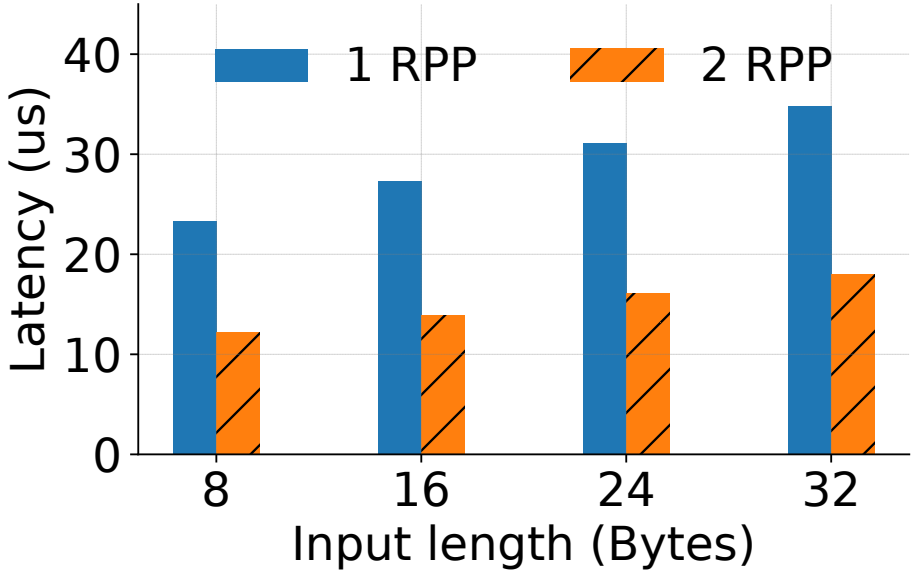
https://github.com/NUS-CIR/tofino-ascon/

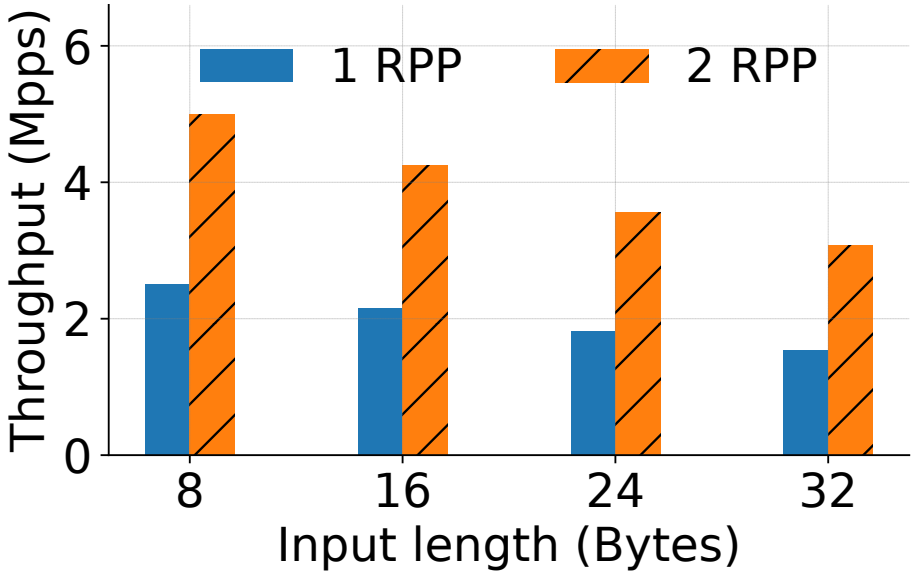
Evaluation Methodology



- Search for maximum achievable throughput by P4EAD switch
  - gradually increase traffic generation rate at the Traffic Gen Switch
- Compare generated and received number of packets
  - run experiment until packet losses are observed
- Experiment repeated 1000 times to ensure consistency

Performance Benchmarks: Tofino

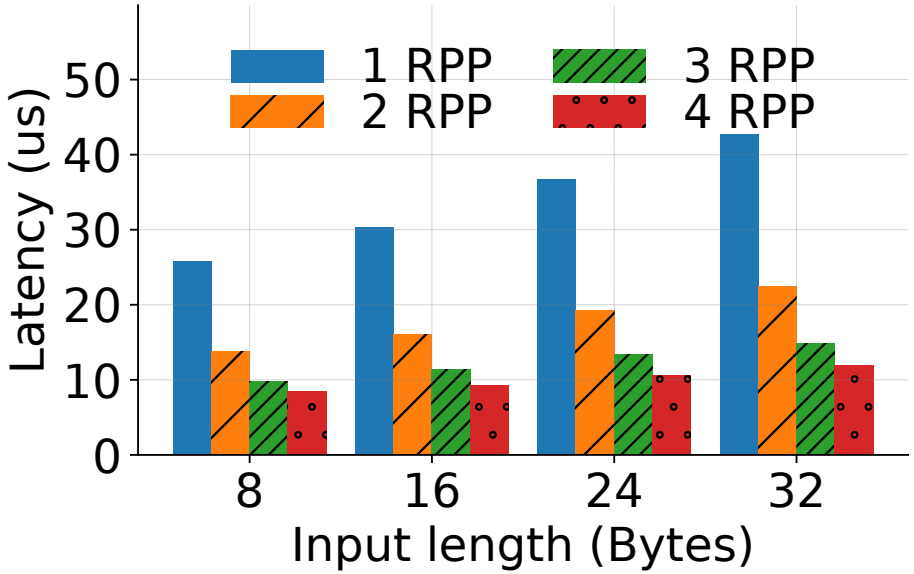


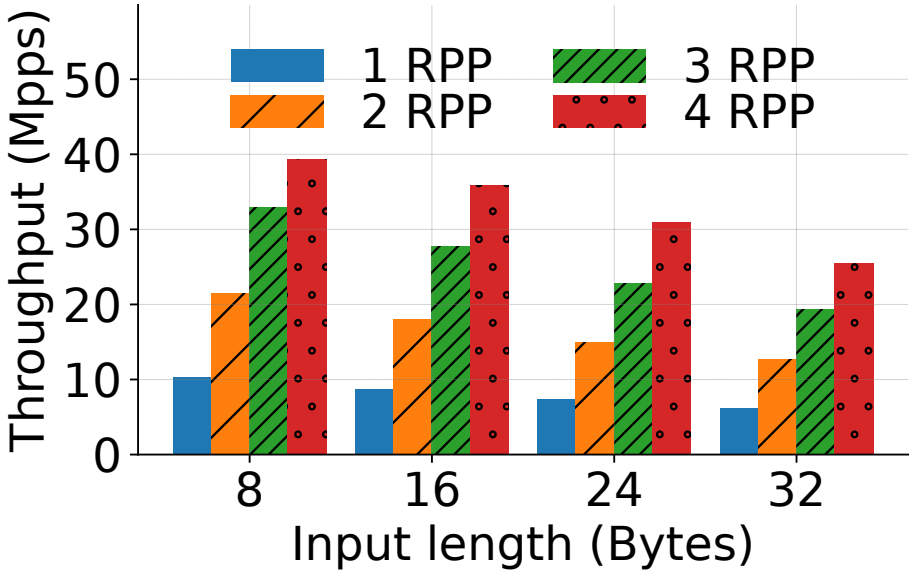


- **Assumption**:
  - 4-byte AD
  - One recirculation port
- **Latency**:
  - **1 RPP**: 23~35 us
  - **2 RPP**: 12~18 us
- **Throughput**:
  - **1 RPP**: 2.5~1.5 Mpps
  - **2 RPP**: 5~3.1 Mpps

RPP: P-RND per pipeline pass

Performance Benchmarks: Tofino2

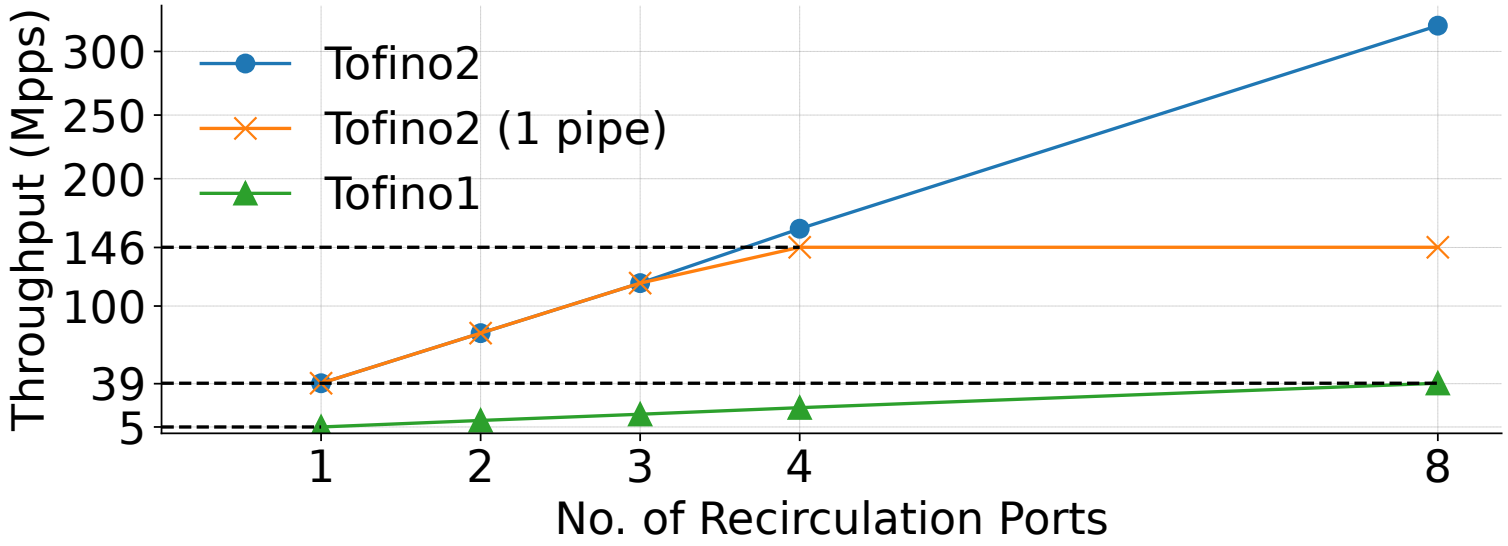




- **Assumption**:
  - 4-byte AD
  - One recirculation port
- **Latency**:
  - **3 RPP**: 9~15 us
  - **4 RPP**: 8.4~12 us
- **Throughput**:
  - **3 RPP**: 33~19 Mpps
  - **4 RPP**: 39~25 Mpps

RPP: P-RND per pipeline pass

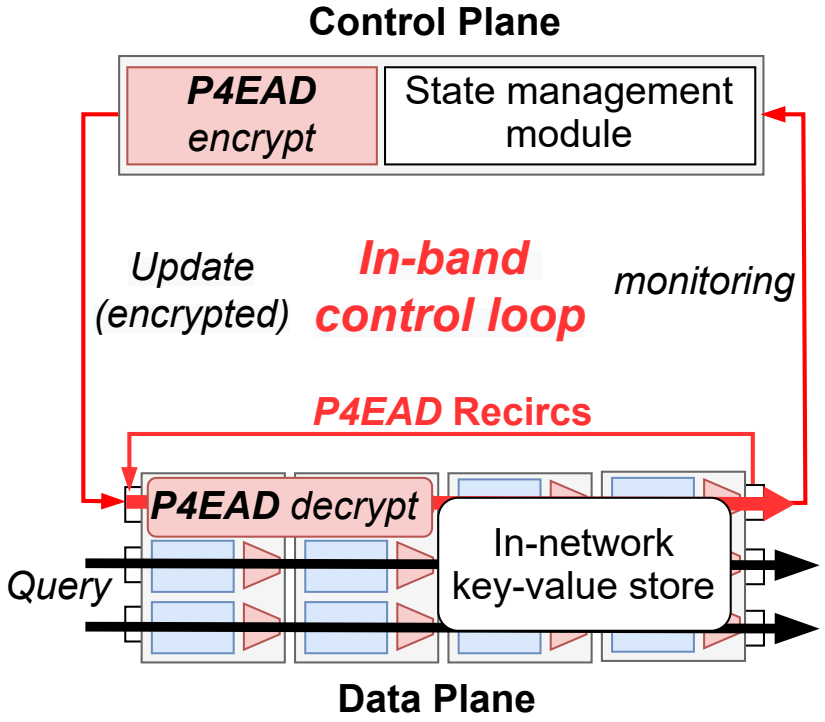
Scalability

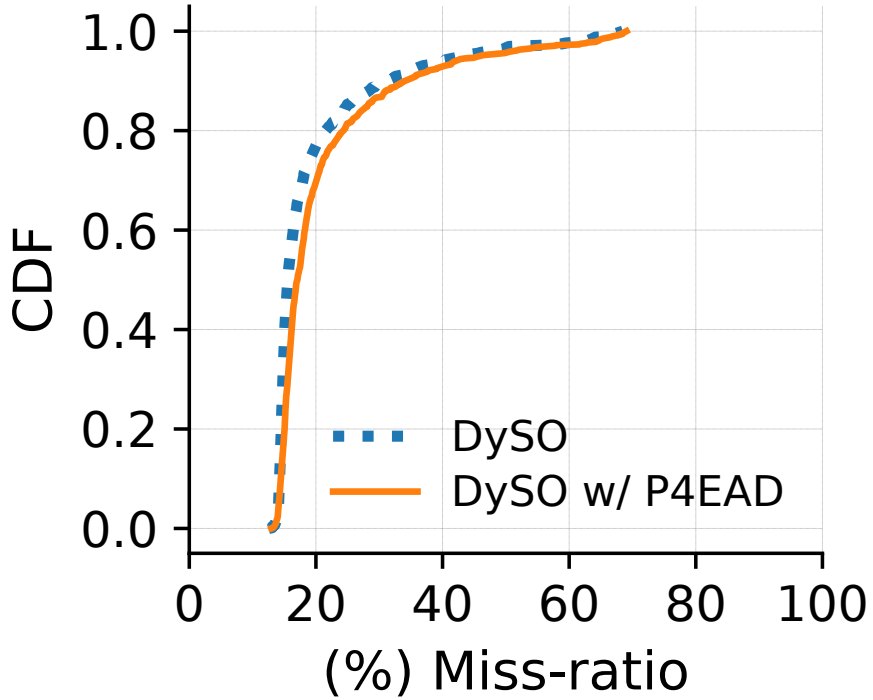


- **Assumption**: 4-byte AD, 8-byte input
- Scale up by increasing the number of recirculation ports
  - Up to **39** Mpps for Tofino using **8** ports
  - Up to **146** Mpps for Tofino2 using **8** ports (same pipeline)
  - Up to **320** Mpps for Tofino2 using **8** ports (different pipeline)

Integration with In-network KV Cache

- In-network KV store accelerated with in-band control channel[1]
  - two variants are compared: with (1 P-RND) and without P4EAD
  - popularity changes every 5s; cache statistics sampled every 1ms





[1] CH Song et al., Revisiting Application Offloads on Programmable Switches. IFIP Networking 2022.

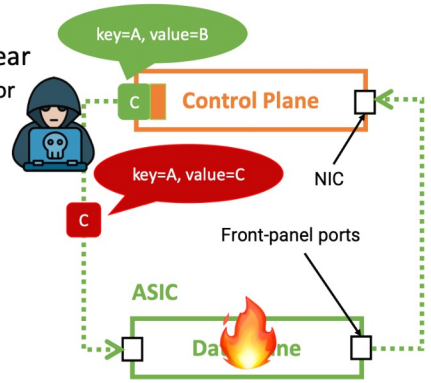
Discussion and Future Work

- Fast and responsive control channels is essential for emerging high-performance/ in-network computing applications
  - The security of these applications must be investigated
- Apart from control channels, **inter-switch communications** will need to be secured too!
- While P4EAD can secure existing solutions…
  - Future chip/switch designs should consider:
    - introducing dedicated hardware crypto accelerators
    - built-in fast and responsive control channels

Summary

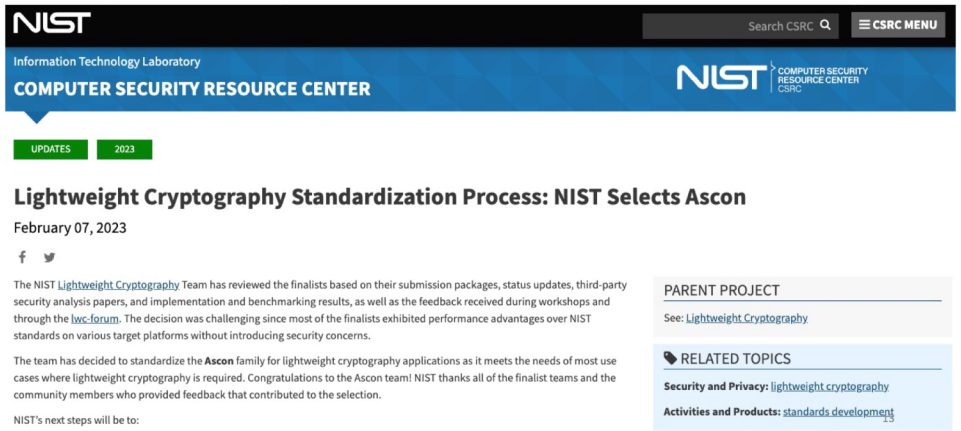
Existing in-band control channels are *vulnerable*!

- Control packets are sent in the clear
  - **Threat Model**: Can be sniffed and/or modified by potential adversaries!
- Systems are thus vulnerable to attacks like:
  - man-in-the-middle
  - denial-of-service
  - replay attacks



8

Lightweight Crypto Suite: *ASCON*



Implementing P4EAD

Context: Programmable switches have limited pipeline stages (e.g., 12)

How to implement ASCON-AEAD on Tofino? (recall 8-stage per P-RND)

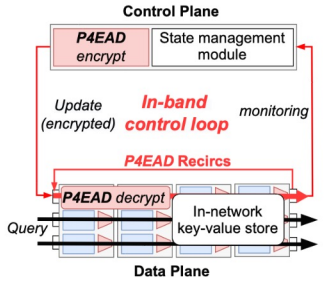
packet recirculation: add *special* metadata header to the packet to store the current P-RND number and relevant states.

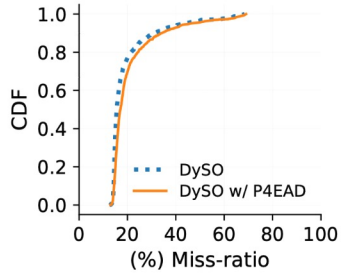
double the P-RNDs: mutually exclusive operations can be parallelized while sharing the pipeline stages

18

Integration with In-network KV Cache

- In-network KV store accelerated with in-band control channel[1]
  - two variants are compared: with and without P4EAD (1 P-RND per pass)





[1] CH Song et al., DySO: Enhancing application offload efficiency on programmable switches. Computer Networks 224 (2023).

26

Talk to me!