

P4Chaskey: An Efficient MAC Algorithm for PISA Switches

Martim Francisco*, Bernardo Ferreira*, Fernando M. V. Ramos[†],
Eduard Marin[†], Salvatore Signorello^{‡*}

*LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

[†]INESC-ID, Instituto Superior Técnico, University of Lisbon

[‡]Telefonica Research, Spain

7th European P4 Workshop – EuroP4'24

October 28, 2024 – Charleroi, Belgium

In-Network Cryptography

The use of cryptography in the data plane has shown potential to improve anonymity, DDoS defense and BFT protocol design.

Programmable In-Network Obfuscation of Traffic

Liang Wang, Hyojoon Kim, Prateek Mittal, Jennifer Rexford
Princeton University

ABSTRACT

Recent advances in programmable switch hardware offer a fresh opportunity to protect user privacy. This paper presents PINOT, a lightweight in-network anonymity solution that runs at line rate within the memory and processing constraints of hardware switches. PINOT encrypts a client's IPv4 address with an efficient encryption scheme to hide the address from downstream ASes and the destination server. PINOT is readily deployable, requiring no end-user software or cooperation from networks other than the trusted network where it runs. We implement a PINOT prototype on the Barefoot Tofino switches, deploy PINOT in a campus network, and present results on protecting user identity against public DNS, NTP, and WireGuard VPN services.

1 INTRODUCTION

Network traffic contains privacy-sensitive information. While encryption protocols such as TLS provide confidentiality of data, they do not hide sensitive meta-data such as the identities of endpoints. Specifically, an Internet Protocol (IP) address can still be used to pinpoint and identify a user and device communicating on the Internet, putting privacy and security at risk [17–19, 22, 31, 43, 45]. However, existing approaches for anonymous communications either introduce high performance overhead (e.g., Tor) [12, 42] or face significant deployment challenges [4–6, 11, 24, 37].

Programmable switch hardware creates an opportunity to build a high-performance anonymity system by offloading privacy functionality to the network. Nevertheless, programmable switch hardware has limited memory and processing resources, posing challenges for implementing cryptographic algorithms that are commonly used in privacy applications. This begs the question: can we leverage programmable data planes to design a readily-deployable anonymity system that balances the privacy-performance trade-off?

In this paper, we push the boundaries of offloading privacy functionality to programmable data planes. We present PINOT (Programmable In-Network Obfuscation of Traffic), a lightweight anonymity system that runs in programmable switch hardware. In this preliminary work, we focus on protecting client IPv4 addresses against particular public services that provide DNS, NTP, and WireGuard VPN (a connections VPN services), and will extend PINOT to

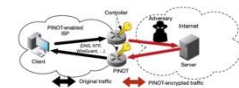


Figure 1: PINOT setup.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

leave the network, as shown in Figure 1. Meta-data is captured and

efficiently encrypted to 12.8% the size of the original IPv4 address to hide the source and destination IP addresses. This is achieved by using a state-of-the-art stream cipher, ChaCha20, to encrypt the IPv4 address. The resulting ciphertext is then XORed with the destination IP address to produce the final obfuscated IP address. This process is performed in hardware on the switch, ensuring that the obfuscation is performed at line rate. The switch also performs de-obfuscation of the traffic before it reaches the destination server.

RAVEN: Stateless Rapid IP Address Variation for Enterprise Networks

Liang Wang
Princeton University
lw19@princeton.edu

Prateek Mittal
Princeton University
pmittal@princeton.edu

ABSTRACT

Enterprise networks face increasing threats against the privacy of their clients. Existing enterprise services like Network Address Translation (NAT) offer limited privacy protection, at the cost of requiring per-flow state. In this paper, we introduce RAVEN (Rapid

Address Variation) to provide stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

How can an enterprise network protect the privacy of its clients? Most solutions today are client-driven. For example, clients can use services like Tor [16]. Apple also launched the iCloud Private Relay

service [1]. Defenses, such as Private Internet Exchange (PIX) [2], can be used to direct traffic to a private network. However, these solutions require stateful NAT, which is not suitable for stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

How can an enterprise network protect the privacy of its clients? Most solutions today are client-driven. For example, clients can use services like Tor [16]. Apple also launched the iCloud Private Relay service [1]. Defenses, such as Private Internet Exchange (PIX) [2], can be used to direct traffic to a private network. However, these solutions require stateful NAT, which is not suitable for stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

How can an enterprise network protect the privacy of its clients? Most solutions today are client-driven. For example, clients can use services like Tor [16]. Apple also launched the iCloud Private Relay service [1]. Defenses, such as Private Internet Exchange (PIX) [2], can be used to direct traffic to a private network. However, these solutions require stateful NAT, which is not suitable for stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

How can an enterprise network protect the privacy of its clients? Most solutions today are client-driven. For example, clients can use services like Tor [16]. Apple also launched the iCloud Private Relay service [1]. Defenses, such as Private Internet Exchange (PIX) [2], can be used to direct traffic to a private network. However, these solutions require stateful NAT, which is not suitable for stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

How can an enterprise network protect the privacy of its clients? Most solutions today are client-driven. For example, clients can use services like Tor [16]. Apple also launched the iCloud Private Relay service [1]. Defenses, such as Private Internet Exchange (PIX) [2], can be used to direct traffic to a private network. However, these solutions require stateful NAT, which is not suitable for stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

How can an enterprise network protect the privacy of its clients? Most solutions today are client-driven. For example, clients can use services like Tor [16]. Apple also launched the iCloud Private Relay service [1]. Defenses, such as Private Internet Exchange (PIX) [2], can be used to direct traffic to a private network. However, these solutions require stateful NAT, which is not suitable for stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

How can an enterprise network protect the privacy of its clients? Most solutions today are client-driven. For example, clients can use services like Tor [16]. Apple also launched the iCloud Private Relay service [1]. Defenses, such as Private Internet Exchange (PIX) [2], can be used to direct traffic to a private network. However, these solutions require stateful NAT, which is not suitable for stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

How can an enterprise network protect the privacy of its clients? Most solutions today are client-driven. For example, clients can use services like Tor [16]. Apple also launched the iCloud Private Relay service [1]. Defenses, such as Private Internet Exchange (PIX) [2], can be used to direct traffic to a private network. However, these solutions require stateful NAT, which is not suitable for stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

Hyojoon Kim
Princeton University
hyojoonk@cs.princeton.edu

Jennifer Rexford
Princeton University
jrex@cs.princeton.edu

ABSTRACT

Enterprise networks face increasing threats against the privacy of their clients. Existing enterprise services like Network Address Translation (NAT) offer limited privacy protection, at the cost of requiring per-flow state. In this paper, we introduce RAVEN (Rapid

Address Variation) to provide stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

How can an enterprise network protect the privacy of its clients? Most solutions today are client-driven. For example, clients can use services like Tor [16]. Apple also launched the iCloud Private Relay

service [1]. Defenses, such as Private Internet Exchange (PIX) [2], can be used to direct traffic to a private network. However, these solutions require stateful NAT, which is not suitable for stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

How can an enterprise network protect the privacy of its clients? Most solutions today are client-driven. For example, clients can use services like Tor [16]. Apple also launched the iCloud Private Relay service [1]. Defenses, such as Private Internet Exchange (PIX) [2], can be used to direct traffic to a private network. However, these solutions require stateful NAT, which is not suitable for stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

How can an enterprise network protect the privacy of its clients? Most solutions today are client-driven. For example, clients can use services like Tor [16]. Apple also launched the iCloud Private Relay service [1]. Defenses, such as Private Internet Exchange (PIX) [2], can be used to direct traffic to a private network. However, these solutions require stateful NAT, which is not suitable for stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

How can an enterprise network protect the privacy of its clients? Most solutions today are client-driven. For example, clients can use services like Tor [16]. Apple also launched the iCloud Private Relay service [1]. Defenses, such as Private Internet Exchange (PIX) [2], can be used to direct traffic to a private network. However, these solutions require stateful NAT, which is not suitable for stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

How can an enterprise network protect the privacy of its clients? Most solutions today are client-driven. For example, clients can use services like Tor [16]. Apple also launched the iCloud Private Relay service [1]. Defenses, such as Private Internet Exchange (PIX) [2], can be used to direct traffic to a private network. However, these solutions require stateful NAT, which is not suitable for stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

How can an enterprise network protect the privacy of its clients? Most solutions today are client-driven. For example, clients can use services like Tor [16]. Apple also launched the iCloud Private Relay service [1]. Defenses, such as Private Internet Exchange (PIX) [2], can be used to direct traffic to a private network. However, these solutions require stateful NAT, which is not suitable for stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

How can an enterprise network protect the privacy of its clients? Most solutions today are client-driven. For example, clients can use services like Tor [16]. Apple also launched the iCloud Private Relay service [1]. Defenses, such as Private Internet Exchange (PIX) [2], can be used to direct traffic to a private network. However, these solutions require stateful NAT, which is not suitable for stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

How can an enterprise network protect the privacy of its clients? Most solutions today are client-driven. For example, clients can use services like Tor [16]. Apple also launched the iCloud Private Relay service [1]. Defenses, such as Private Internet Exchange (PIX) [2], can be used to direct traffic to a private network. However, these solutions require stateful NAT, which is not suitable for stateless, publicly-routable IPv6 address that makes it easier for an adversary to locate a client and analyze its traffic [13, 27, 53].

SMARTCOOKIE: Blocking Large-Scale SYN Floods with a Split-Proxy Defense on Programmable Data Plans

Sophia Yoo
Princeton University
sophiayoo@princeton.edu

Xiaoqi Chen
Princeton University
xiaoqi@cs.princeton.edu

Jennifer Rexford
Princeton University
jrex@cs.princeton.edu

ABSTRACT

Despite decades of mitigation efforts, SYN flooding attacks

One of the most common DDoS attacks, namely SYN floods, consume server memory until the server is forced to drop benign traffic [37, 38, 58]. SYN floods constituted up to 94.7% of all DDoS attacks in 2020 [49] and continue to remain a critical threat today [50]. Additionally, benign traffic volumes also continue to grow exponentially, reaching staggering loads of up to hundreds of Gbps in cloud-provider networks [63].

To respond to growing threats and traffic volumes, network providers (e.g., cloud providers, enterprise networks, ISPs) urgently require scalable defenses that block volumetric attacks without compromising security against adaptive adversaries or degrading application performance [21, 40, 58]. In other words, they need defenses with three key requirements: security (blocking attacks from adaptive adversaries), scalability (handling large amounts of benign and attack traffic), and performance (low latency for benign clients).

While state-of-the-art SYN-flooding defenses using SYN cookies have been proposed and standardized for many years, existing solutions have failed to simultaneously provide security, scalability, and performance [2, 3, 51, 67, 72]. Designing and practically implementing modern SYN-flooding defenses that meet each of these requirements is particularly challenging. Compared to defenses against many other volumetric attacks (e.g., ACK floods, RST floods, UDP amplification), SYN-flooding defenses cannot simply drop, rate-limit, or ignore unsolicited traffic. Doing so might cause denial of benign client requests, particularly since adversaries often spoof attack packets using legitimate addresses. Instead, defenses must identify and block large-scale attacks (requiring costly compute) while keeping per-flow state to track large numbers of verified connections (requiring large chunks of memory).

Server-Based Solutions. Server-based SYN cookie defenses are the time-honored way to provide security and can scalably handle benign traffic by default [2, 3, 67]. However, today large-scale attacks, software-based packet-protection

SPINE: Surveillance Protection in the Network Elements

Trisha Datta, Nick Feamster, Jennifer Rexford, and Liang Wang
Princeton University

Abstract

Internet Protocol (IP) addresses can reveal information about communicating Internet users and devices, even when the rest of the traffic between them is encrypted. At the same time, IP addresses serve as endpoints for network-layer communication and, as a result, are typically visible to the intermediate routers to allow them to forward traffic to its ultimate destination. Previous approaches to obfuscate the IP addresses of the sender and receiver commonly depend on either custom user software (e.g., Tor browser) or significant modifications to network hardware along the end-to-end path (which has proved to be a major roadblock). SPINE, on the other hand, conceals

States [11]. Many countries rely on Western nations for Internet content [9], potentially raising these types of concerns for other countries. More generally, any network's ability to observe source and destination IP addresses puts privacy at risk by revealing the endpoints of the communications.

Previous work recognizes this privacy threat [4–6, 10, 19], but existing approaches face significant deployment challenges. IPSEC tunnels, for instance, provide protection by completely encrypting original IP packets. However, such operations are too expensive to deploy in the core of the Internet and redundant if payloads are already encrypted with TLS. Another well-studied approach is network-layer anonymity

In-Network Cryptography for SCION

Emerging Internet architecture designs advocate for per-packet cryptographic verifications.

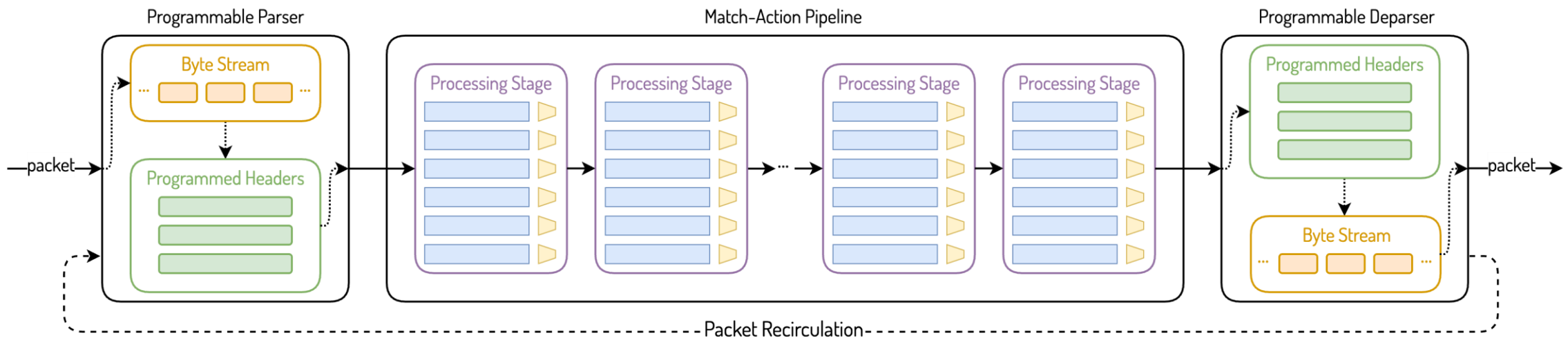


The lack of cryptographic accelerators in networking hardware has precluded cryptographic computations in the network core.

Architectures such as SCION are restricted to implementations in software with throughputs of tens of Gbps at best.

Programmable Terabit Switching Architecture

Terabit speed programmable switching architectures, such as PISA, offer a **clear opportunity** to implement cryptographic primitives in the data plane.



Cryptographic Primitives on PISA Switches

Minimum number of pipeline stages for a 128-bit input

Algorithm	Key Size (Bits)	# of Rounds	Stages per Round	Total Stages
ASCON ¹	128	24	8	192
AES - CMAC ²	128	10	5	50
SipHash ³	128	10	6	60
HalfSipHash ⁴	64	14	4	56

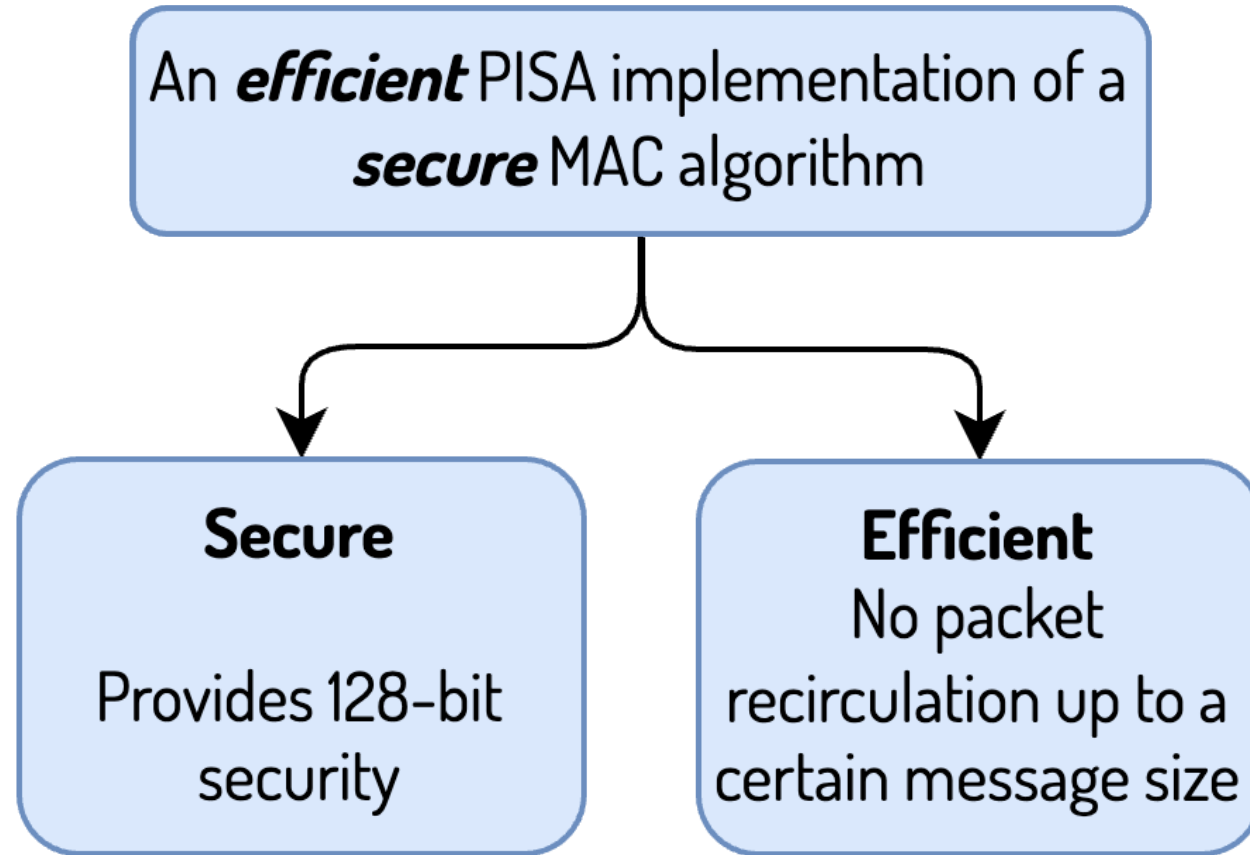
[1] Archit Bhatnagar, Xin Zhe Khooi, Cha Hwan Song, Mun Choon Chan. 2023. P4EAD: Securing the In-band Control Channels on Commodity Programmable Switches;

[2] Lars-Christian Schulz, Robin Wehner, David Hausheer. 2023. Cryptographic Path Validation for SCION in P4;

[3] Yutaro Yoshinaka, Mio Kochiyama, Yuki Koizumi, Junji Takemasa, Toru Hasegawa. 2024. A Lightweight Anonymity Protocol at Terabit Speeds on Programmable Switches;

[4] Sophia Yoo, Xiaoqi Chen. 2021. Secure Keyed Hashing on Programmable Switches.

Target Properties for a MAC on PISA



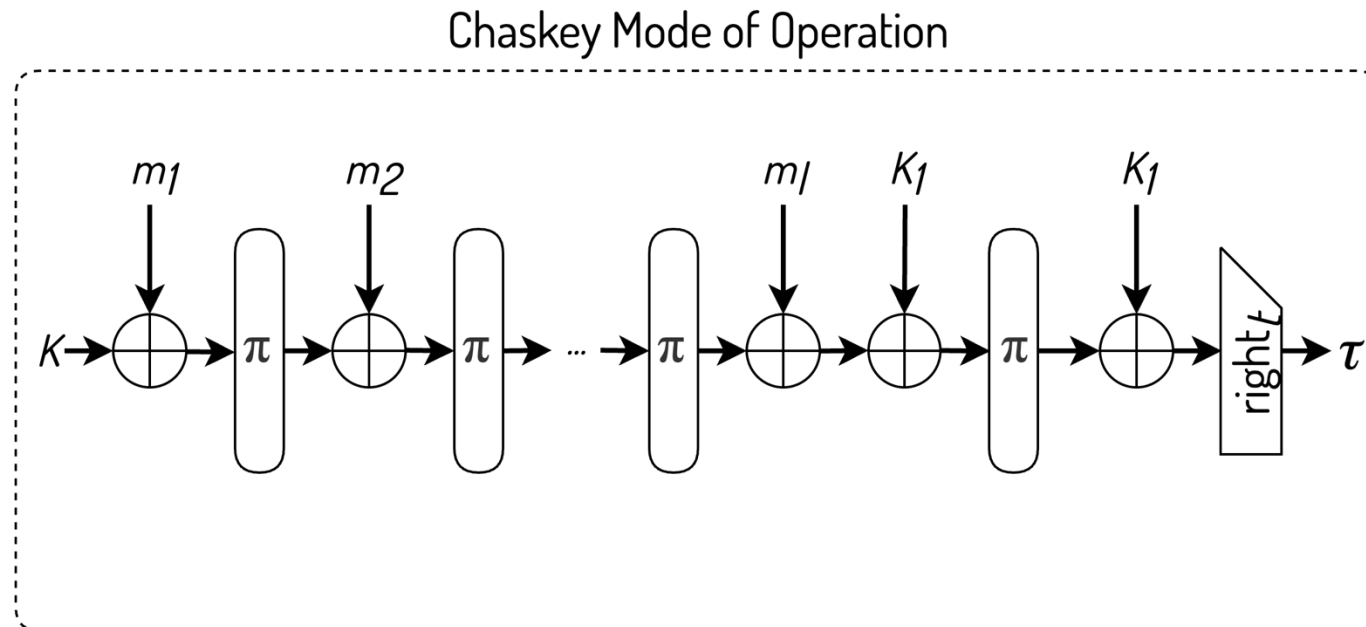


Our answer is P4Chaskey!

The Chaskey MAC Algorithm

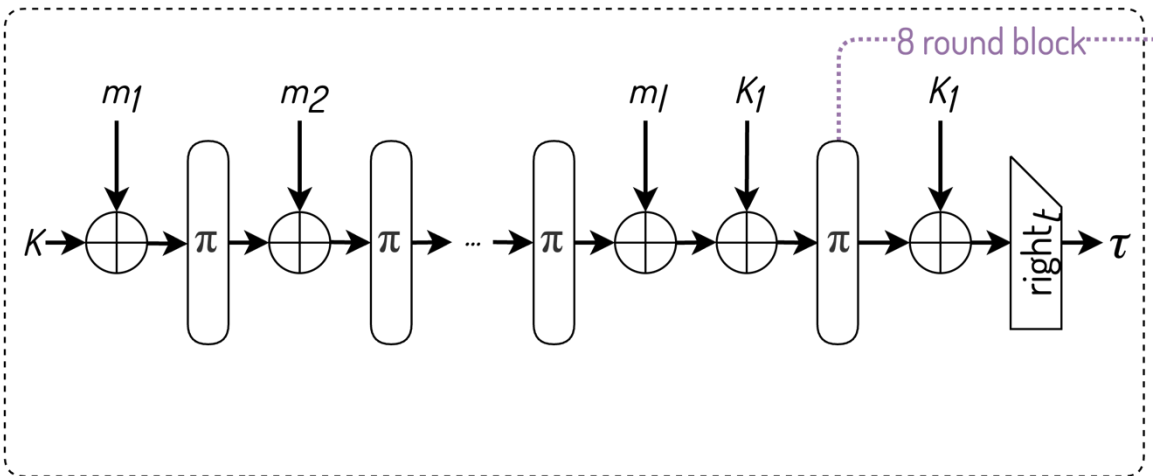
Permutation-based MAC algorithm for 32-bit microcontrollers:

- Processes a message m into l blocks of $n = 128$ bits each through π permutation rounds.

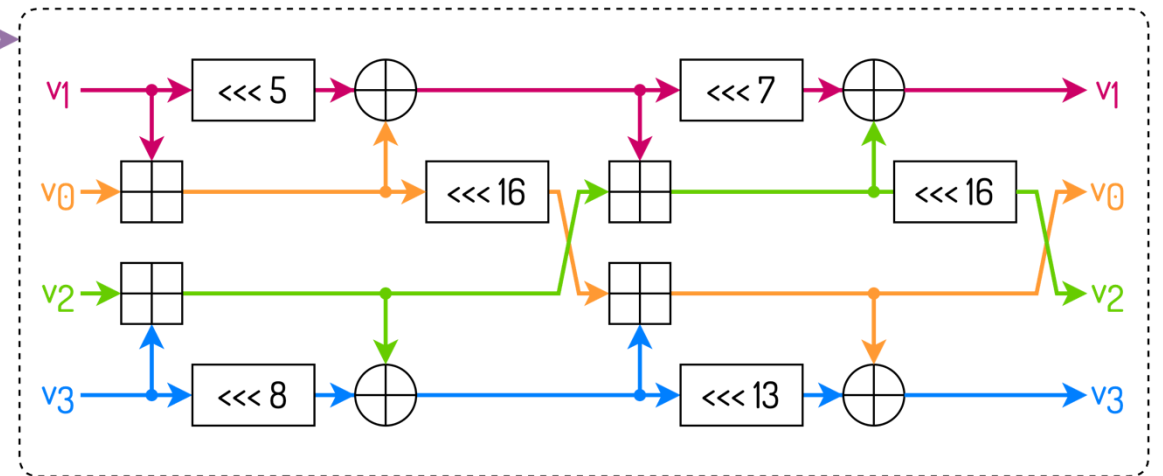


Chaskey's Operations

Chaskey Mode of Operation

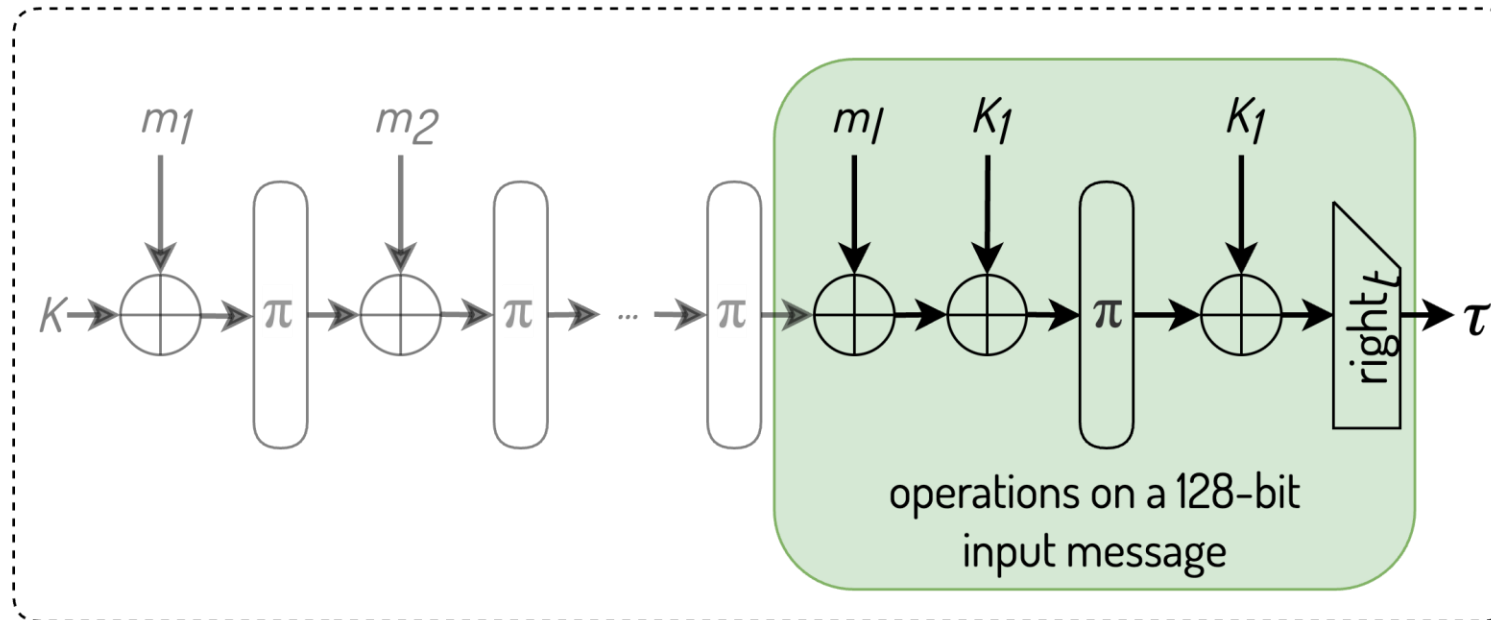


Chaskey's Permutation Round Operations



Chaskey for a 128-bit Input Message

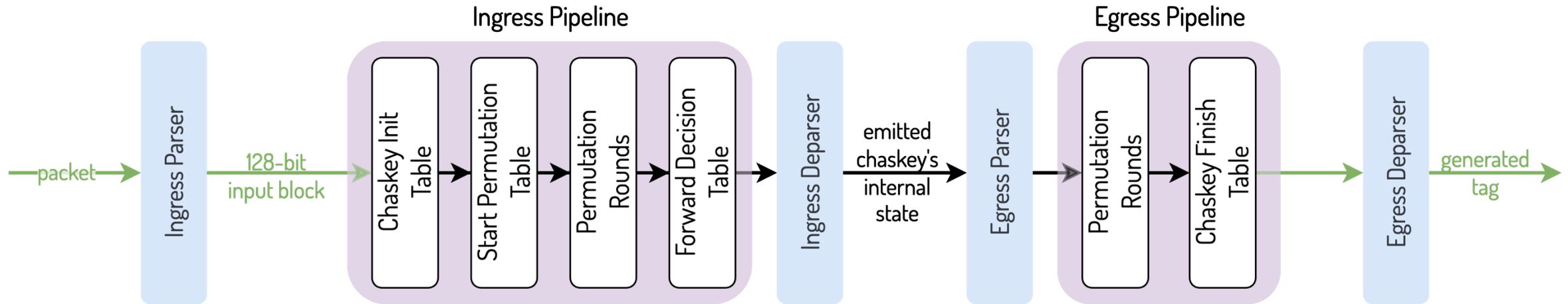
Chaskey Mode of Operation





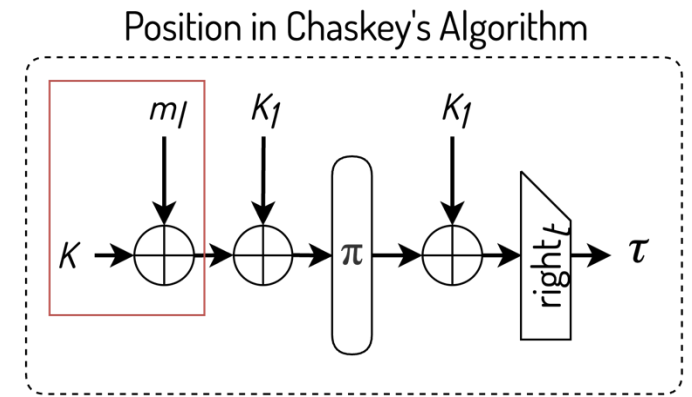
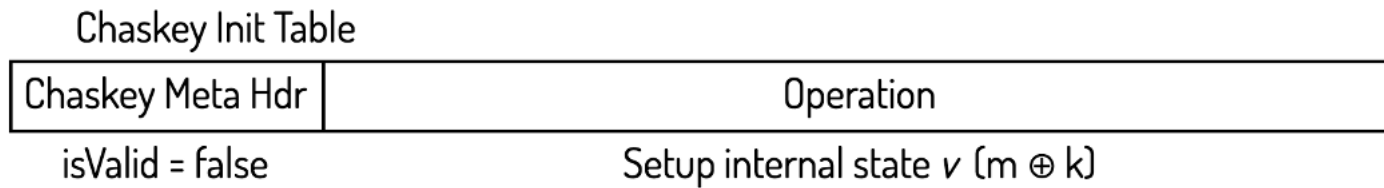
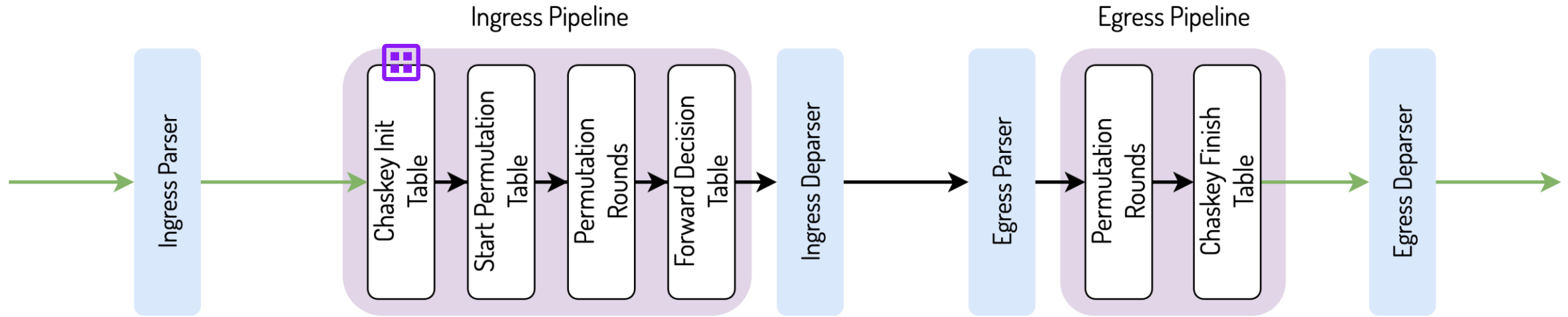
P4Chaskey Design & Implementation

P4Chaskey Design



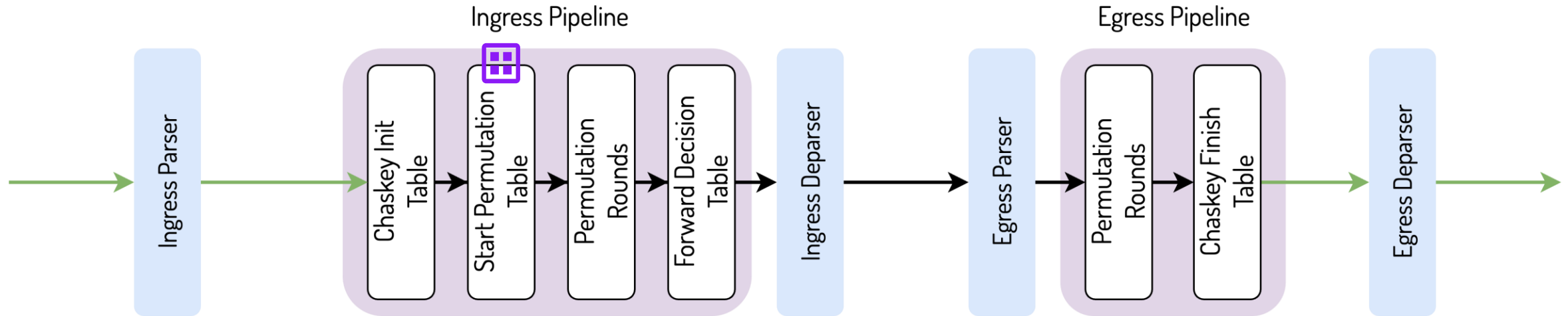
P4Chaskey Control Flow

Current Round: 0

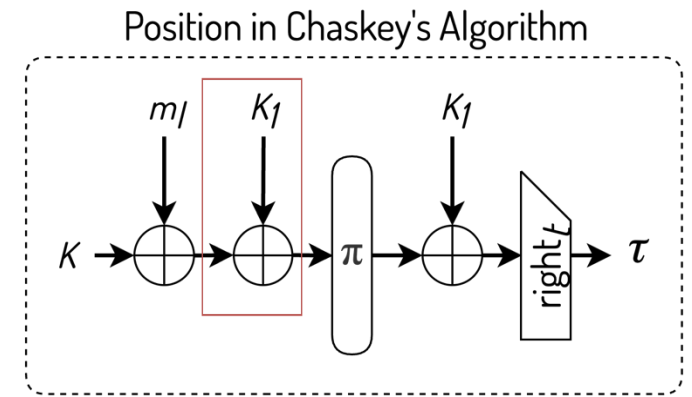


P4Chaskey Control Flow

Current Round: 0

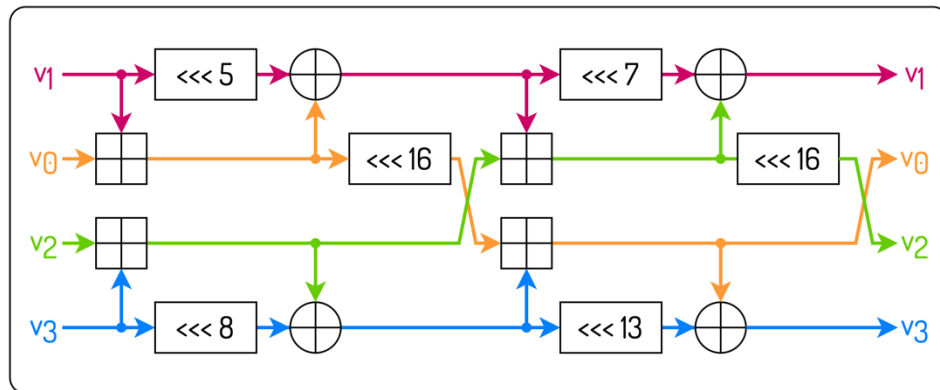
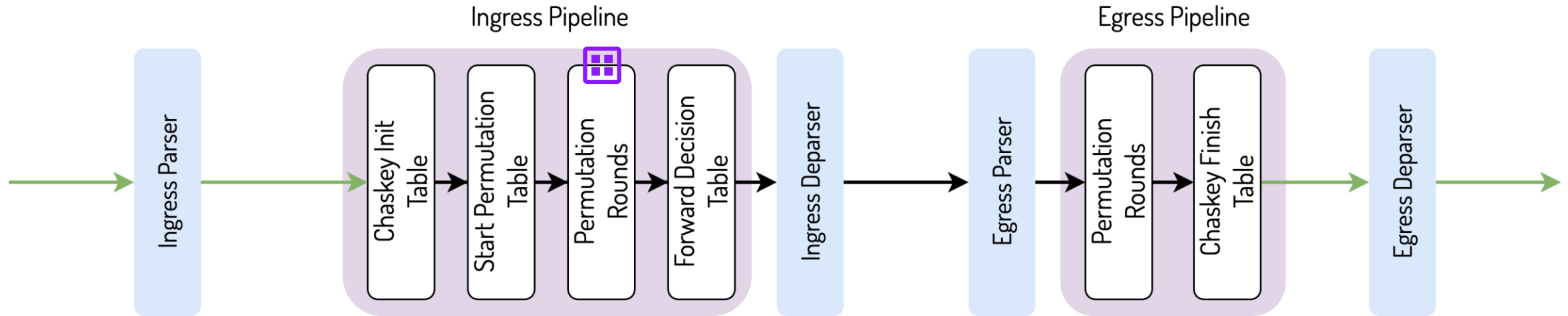


Start Permutation Table	
Current Round	Operation
0	Setup (Final) Permutation ($v \oplus k_1$)

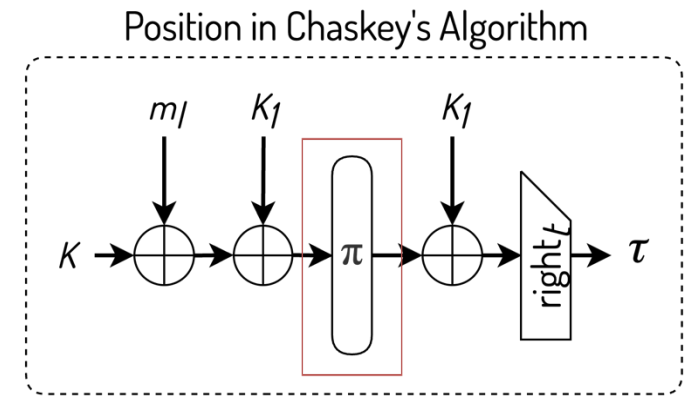


P4Chaskey Control Flow

Current Round: 0

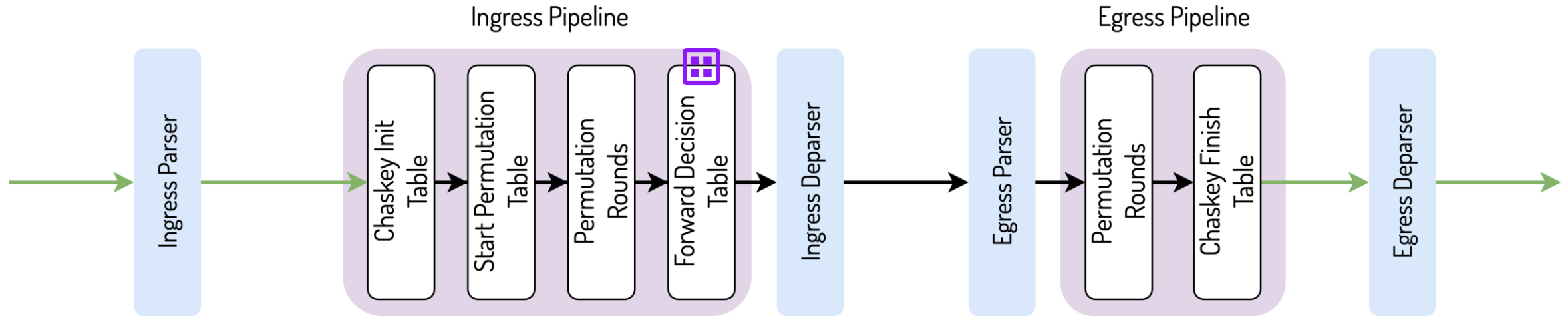


X 4



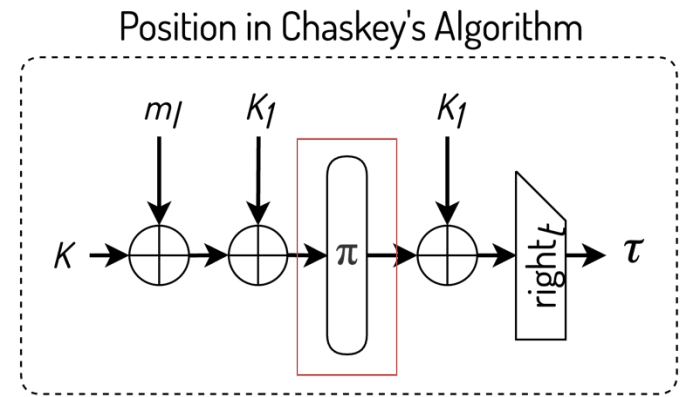
P4Chaskey Control Flow

Current Round: 4



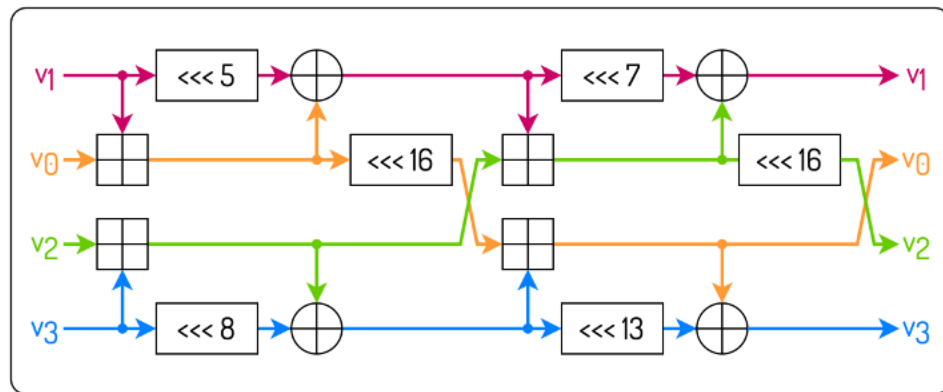
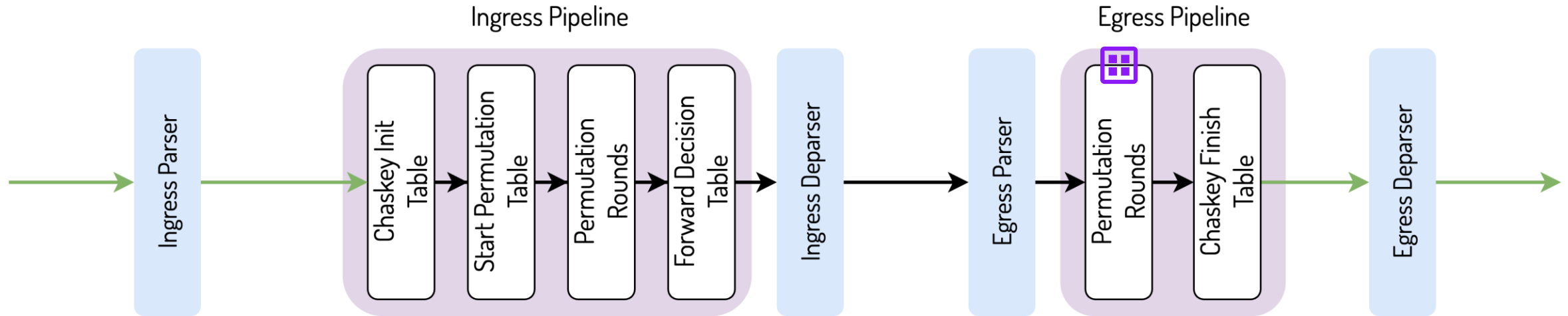
Forward Decision Table

Current Round	Operation
0	Set the output port to the destination Update the current round field to 4

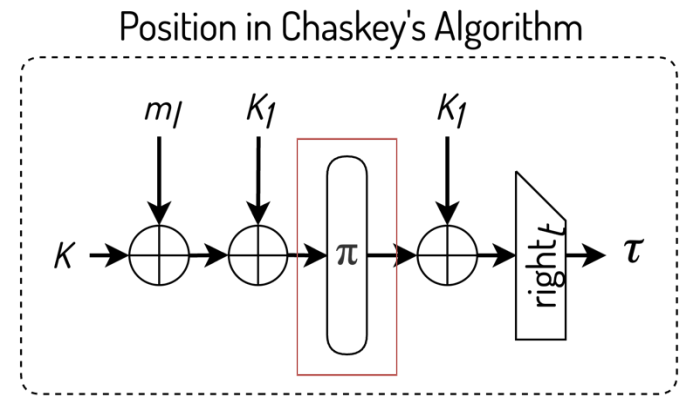


P4Chaskey Control Flow

Current Round: 4

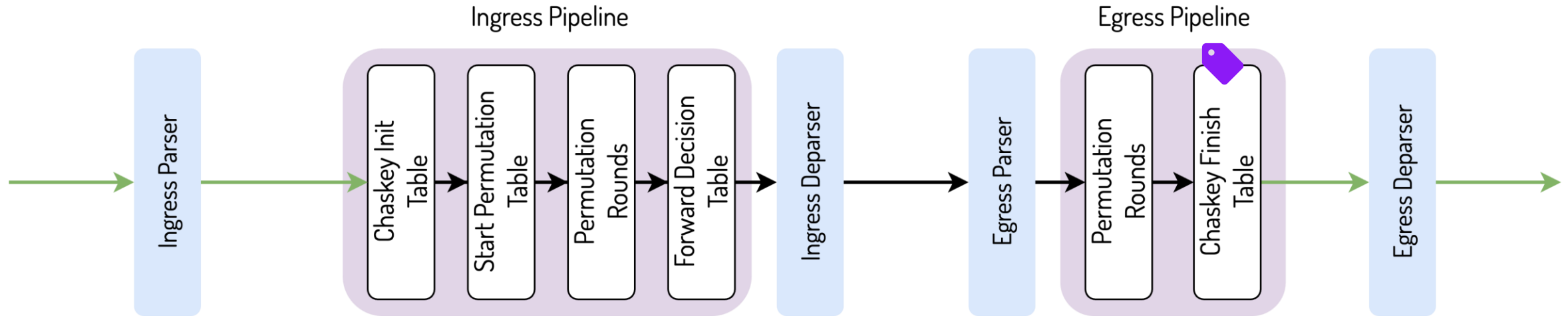


X 4

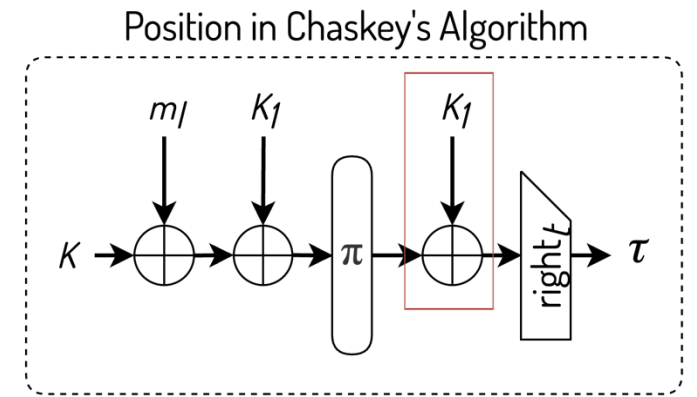


P4Chaskey Control Flow

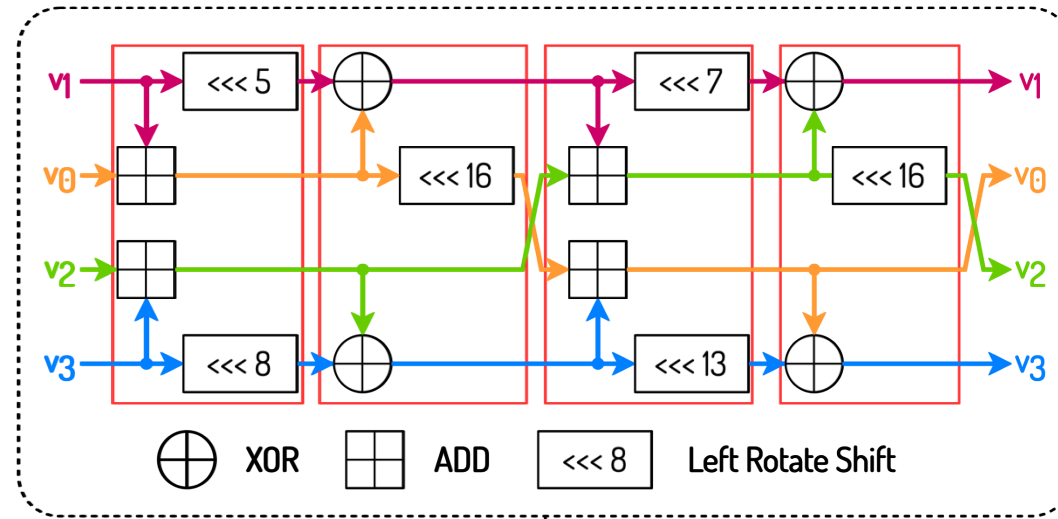
Current Round: 4



Chaskey Finish Table	
Current Round	Operation
4	Final XOR Operation ($v \oplus k_1$)

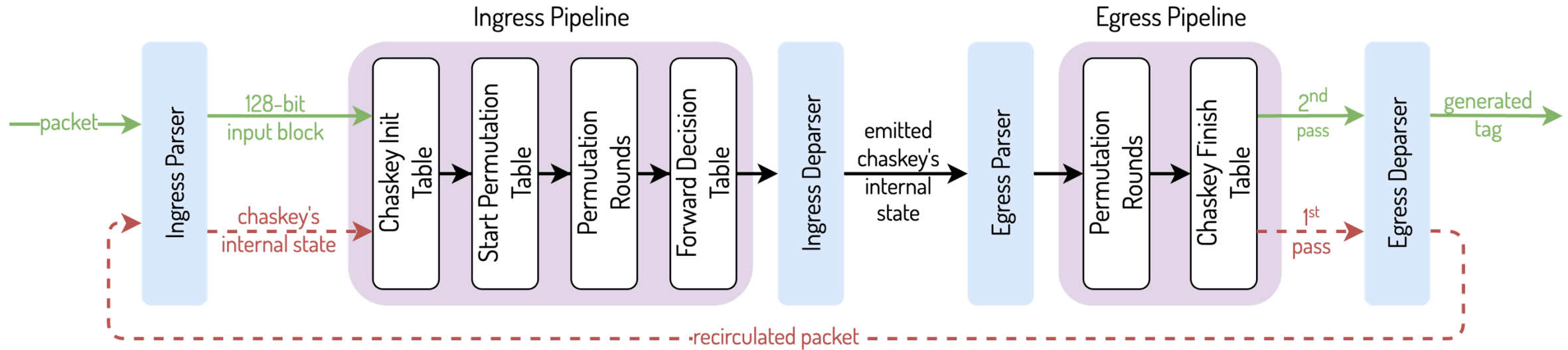


P4Chaskey Permutation Round



Group 1	Group 2	Group 3	Group 4	Stage 1	Stage 2	Stage 3	Stage 4
$v1 \lll 5$	$v1 \oplus = v0$	$v1 \lll 7$	$v1 \oplus = v2$	$a1 = v1 \lll 5$	$b1 = a1 \oplus a0$	$a1 = b1 \lll 7$	$v1 = a1 \oplus a2$
$v0 += v1$	$v0 \lll 16$	$v2 += v1$	$v2 \lll 16$	$a0 = v0 + v1$	$b0 = a0 \lll 16$	$a2 = b2 + b1$	$v2 = a2 \lll 16$
$v2 += v3$	$v3 \oplus = v2$	$v0 += v3$	$v3 \oplus = v0$	$a2 = v2 + v3$	$b3 = a3 \oplus a2$	$a0 = b0 + b3$	$v3 = a3 \oplus v0$
$v3 \lll 8$		$v3 \lll 13$		$a3 = v3 \lll 8$	$b2 = a2$	$a3 = b3 \lll 13$	$v0 = a0$

P4Chaskey Operation on Tofino 1





How did we evaluate P4Chaskey?

Evaluation Targets

P4Chaskey correctness was assessed by using the results obtained with Chaskey's C implementation^[1].

Our evaluation has two main targets:

- Demonstrating that our design and P4 implementation of Chaskey for PISA is the first that enables computing a MAC using a 128-bit key without packet recirculation.
- Measuring the resource usage of P4Chaskey on the target switch platforms.

[1] <https://mouha.be/wp-content/uploads/chaskey12.c>

Comparison Against State-of-the-art Solutions

Number of pipeline passes for a 128-bit input message

Target	Algorithm	Key Size (Bits)	Block/Message Size (Bits)	Pipeline Passes
Tofino 1	HalfSipHash ¹	64	32	5
	Chaskey	128	128	2
Tofino 2	SipHash ²	128	64	2
	AES - CMAC ³	128	128	2
	Chaskey	128	128	1 ✓

[1] Sophia Yoo, Xiaoqi Chen. 2021. Secure Keyed Hashing on Programmable Switches;

[2] Yutaro Yoshinaka, Mio Kochiyama, Yuki Koizumi, Junji Takemasa, Toru Hasegawa. 2024. A Lightweight Anonymity Protocol at Terabit Speeds on Programmable Switches;

[3] Lars-Christian Schulz, Robin Wehner, David Hausheer. 2023. Cryptographic Path Validation for SCION in P4.

Resource usage: Tofino 1 & Tofino 2

Resource Type	Tofino 1	Tofino 2
Number of Stages	10/12	18/20
Action Data Bus Bytes	9,5%	9,5%
VLIW Instructions	8,1%	8,4%
Exact Match Input Xbar	8,6%	10,2%
Gateway	13%	15,6%
Hash Distribution Units	66,7%	80%
Logical Table ID	35,4%	41,3%
PHV Allocation	≤40%	≥40%
SRAM	0,7%	0,4%
TCAM	0%	0%

→ ≤ Pipeline Length

→ Moderate control flow logic

→ Almost no memory used

Integrating P4Chaskey With Other P4 Programs

In our public repository on github, we have provided an example of using P4Chaskey as a control block into a generic P4 program.

- Small reflector program that verifies IPv6 addresses before forwarding.

We are currently integrating P4Chaskey into the desing of a SCION border router in P4 that includes some of the EPIC^[1] security extensions.

Conclusion

We presented P4Chaskey, the first data plane implementation of a MAC algorithm for PISA that:

- Uses 128-bit security.
- It executes in a single pipeline pass (Tofino 2) for inputs up to 128-bits.

Our evaluation has shown that:

- It requires fewer pipeline passes than state-of-the-art implementations.
- Its resource usage allows for other data plane functionality to be executed in parallel.

