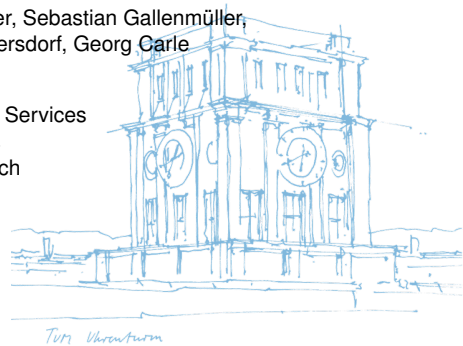
Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Cryptographic Hashing in P4 Data Planes

Dominik Scholz, Andreas Oeldemann, Fabien Geyer, Sebastian Gallenmüller,
Henning Stubbe, Thomas Wild, Andreas Herkersdorf, Georg Carle

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich



Motivation

Manifold P4 Applications and Programs

Imagine long list of P4 programs here
→ Few programs that require cryptographic functionality

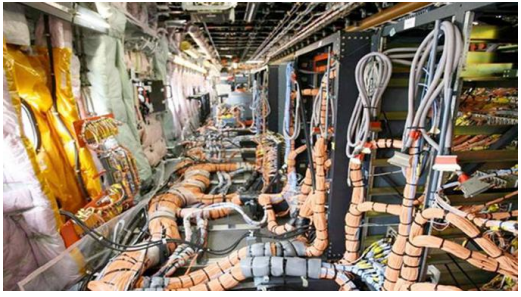


Image from https://bit.ly/2LHVmDZ

P4 is of high interest to industry, e.g. avionics

- Rapid prototyping
- Program verification
- ...

Requires guarantees: e.g. authentication of switches

Motivation

Cryptographic Properties and Functions

Cryptographic properties commonly found in network applications and protocols

- Confidentiality
- Authenticity (data/message integrity)
- Authentication (data origin authentication)

Cryptographic functions

- Encryption
- Hash functions

$\rightarrow$ in this work we focus on cryptographic hash functions

Motivation

Use cases for cryptographic hash functions

Data structures

- Hash maps
- Bloom-Filter

But: cryptographic functions not required
e.g. Bloom-Filter: linear-independent hashes suffice

Authenticity/Authentication

- Message Authentication Codes
- Client puzzles (TCP SYN cookies)

Cryptographic functions required

Outline

Problem Statement

Feasibility of cryptographic hashing in programmable data planes

- Hash with cryptographic properties
- Hash of complete packet content
- Ideally achieving 10 GbE line-rate
- Software and hardware P4 targets

Choice of Hash Function

Cryptography vs. Performance

Cryptographic (hash) functions are

- Slow $\leftrightarrow$ line-rate
- Complex $\leftrightarrow$ resource consumption on target

Pseudo-cryptographic SipHash

- Optimized for small inputs
- Optimized for performance in software

Benchmarks on software system

Hash algorithm	Cycles per B	Fixed cycles per packet	Cycles for 64 B
CRC32	0.32	0.00	10.79
Checksum	0.44	0.00	30.06
SipHash-2-4	1.06	56.40	121.10
BLAKE2b	3.14	35.85	232.77
HMAC-SHA256	5.57	959.69	1462.13

Available P4 Targets

. . . that can be extended with cryptographic hashing

- Software: t4p4s (P4ELTE), based on DPDK
- Network Processing Unit: Netronome Agilio SmartNIC (NFP-4000)
- FPGA: NetFPGA SUME (P4→NetFPGA)
- ASIC: none that we are aware of

P4 Hash Integration

t4p4s

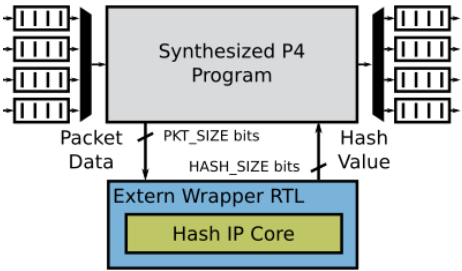
- Trivial: link library, add extern
- Added SipHash-2-4 and HMAC-SHA512 (openssl)

NFP-4000

- Crypto security accelerator (SHA1): not available on our card
- Integrated SipHash-2-4 as extern in variation of C

NetFPGA SUME

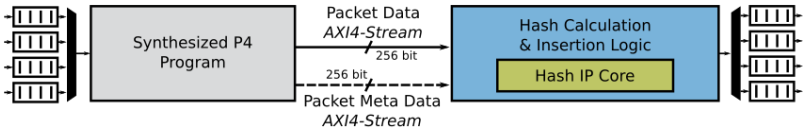
- Externs implemented in Verilog/VHDL
- Integrated SipHash-2-4 and SHA3-512
- Problem:
  - Data passed between P4 program and extern is a single data word
  - SDNet limit: 600 B input width
  - No timing closure due to resource congestion



P4 Hash Integration

NetFPGA

Alternative P4 architecture model



Limitations

- All packets are always hashed
- Hash outcome not usable in P4
- Alternatives:
  - Hashing before P4 pipeline
  - Second P4 pipeline after hashing core
  - Traffic manager
- SHA3-512 core uses 125 MHz $\rightarrow$ clock domain crossing

Measurement Results

Setup

- Load Generator
  - CBR traffic of constant packet size
  - Precise latency measurements
- Device-under-Test (DuT)
  - Intel Xeon E5-2620 with Intel X540 NIC
  - Netronome NFP-4000 SmartNIC
  - NetFPGA SUME
- P4 program
  - L2 forwarder
  - Hashes complete packets

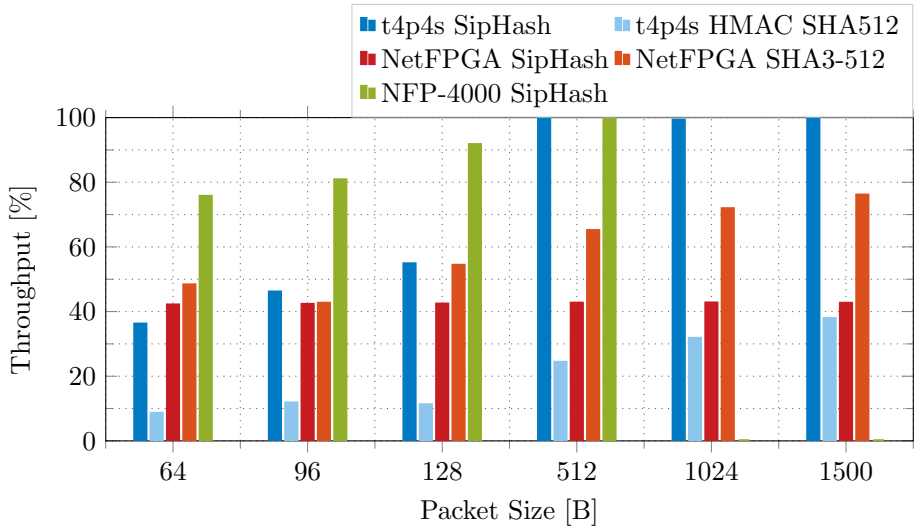
Disclaimer

- Open-source implementations
- Non-commercial IP cores
- Not optimized integration → proof-of-concept
- Take performance figures with grain of salt

→ conservative numbers

Measurement Results

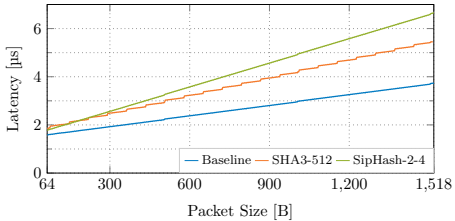
Achievable Throughput



Measurement Results

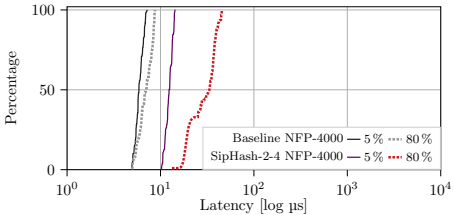
Latency

NetFPGA



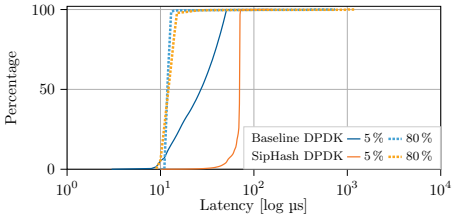
Stable latency: no long-tail

NFP-4000



t4p4s

Typical behavior of software system/DPDK: long-tail



Measurement Results

Resource Utilization

Does the program fit on the target?

- t4p4s: trivial
- NFP-4000: no restrictions enountered
- NetFPGA:

	LUTs		Registers		BRAM	
	Abs.	%	Abs.	%	Abs. [kB]	%
Baseline	64,533	14.90	109,783	12.67	16,362	30.92
SipHash-2-4	66,380	15.32	114,282	13.19	17,460	32.99
SHA3-512	73,449	16.95	118,689	13.70	17,460	32.99

Conclusion

The current use of hash functions in P4 programs

- Data structures might be vulnerable to attacks (hash collisions)
- Lack of programs/protocols requiring authentication and integrity

→ Cryptographic hash functions increase applicability of P4

Cryptographic hashing is target, algorithm and use-case dependent

- Line-rate possible on hardware targets
- Integration for instance by adjusting P4 architecture model
- Algorithms might be better on one target than another

→ no one-size-fits-all solution

→ P4 specification should recommend family of hash functions, including cryptographic ones