Portfolio of optimized cryptographic functions based on KECCAK

Guido BERTONI[1] Joan DAEMEN[1,2] Michaël PEETERS[1]
Gilles VAN ASSCHE[1] Ronny VAN KEER[1]

[1]STMicroelectronics
[2]Radboud University

FOSDEM, Brussels, February 4-5, 2017

Outline

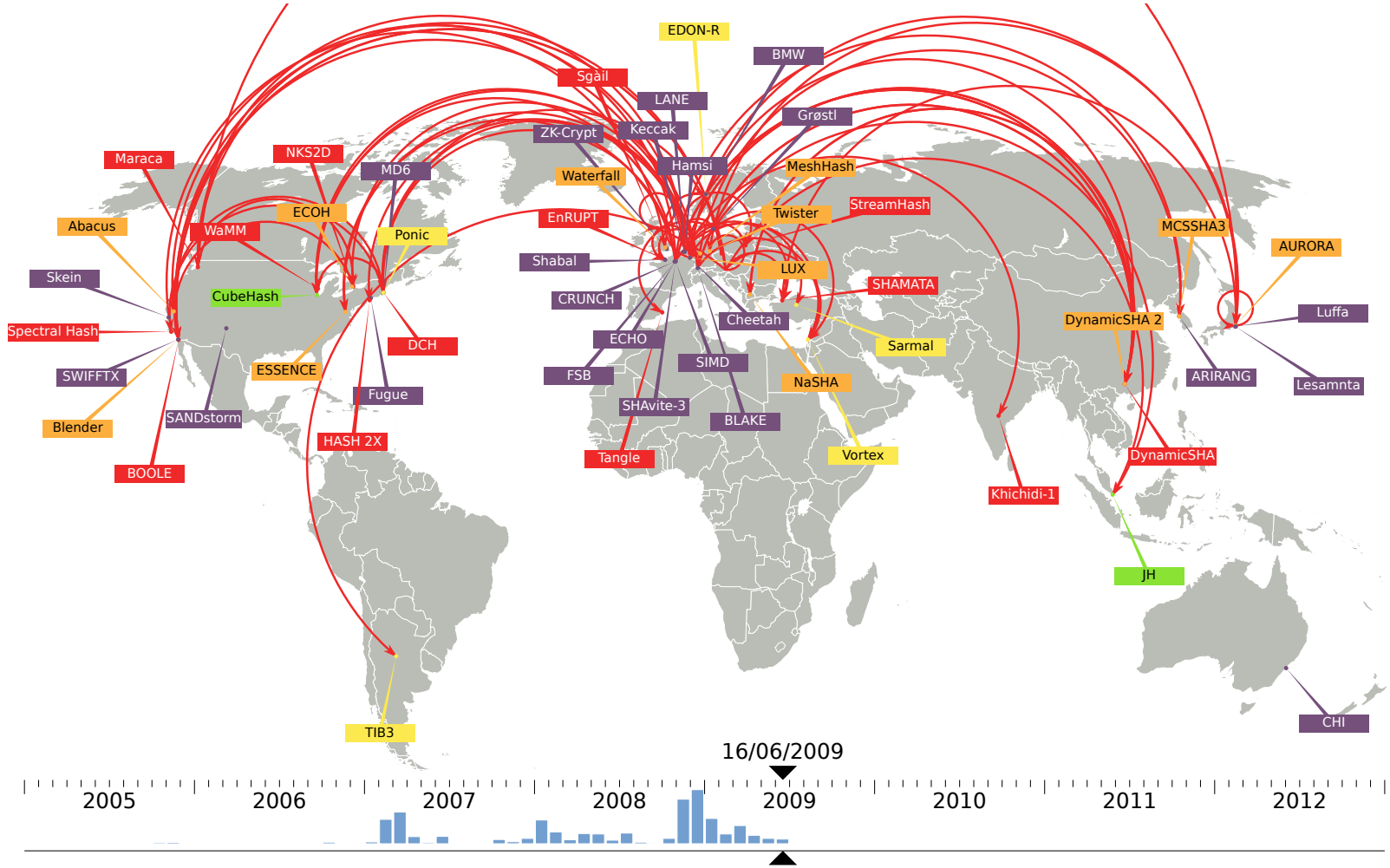
Outline

Crisis!

By Marcel Germain (flickr.com)

- 2004: SHA-0 broken (Joux et al.)
- 2004: MD5 broken (Wang et al.)
- 2005: practical attack on MD5 (Lenstra et al., and Klima)
- 2005: SHA-1 theoretically broken (Wang et al.)
- 2006: SHA-1 broken further (De Cannière and Rechberger)
- 2007: NIST calls for SHA-3

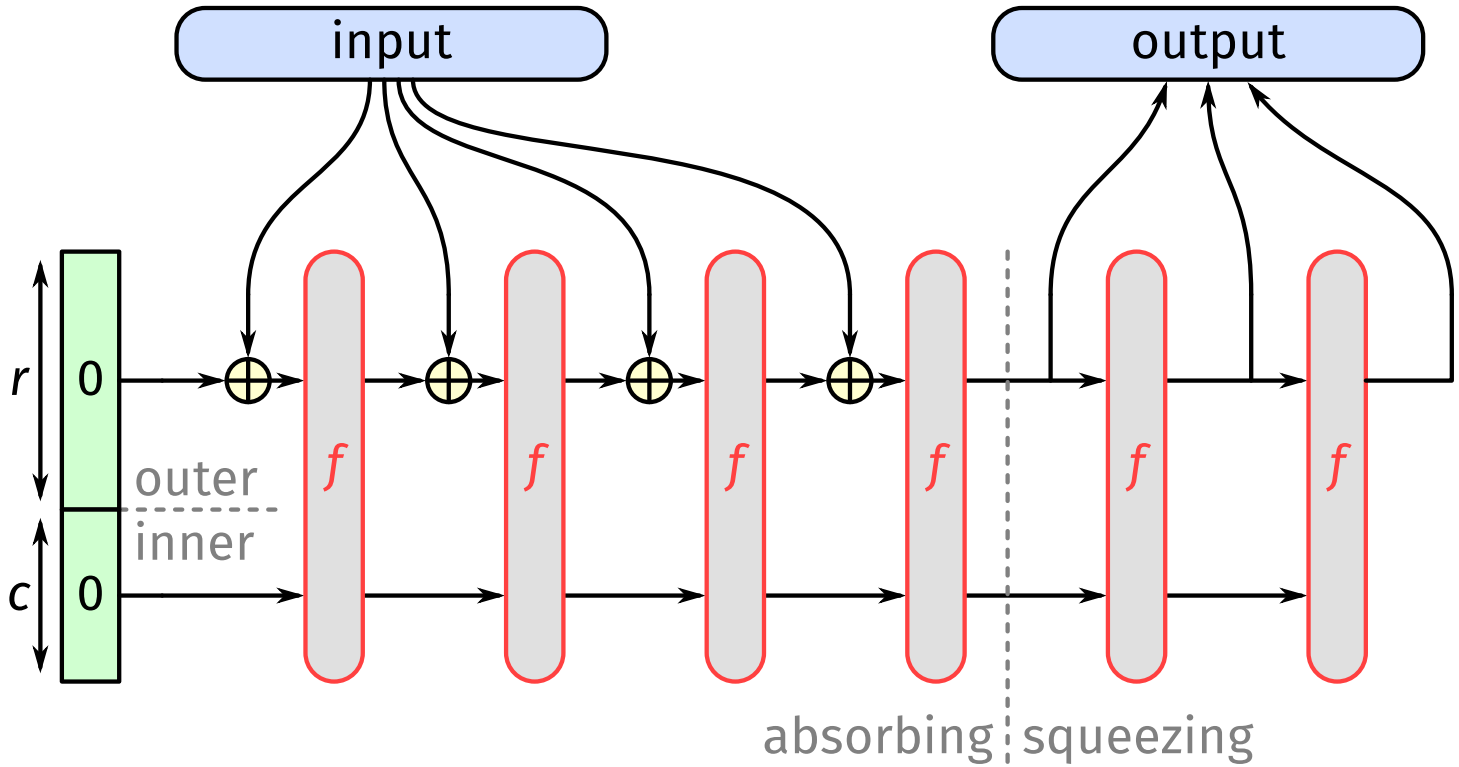
The SHA-3 competition (2008-2012)



[courtesy of Christophe De Cannière]

Our candidate: KECCAK

KECCAK is a *sponge function* ...



... that uses the KECCAK-*f* permutation

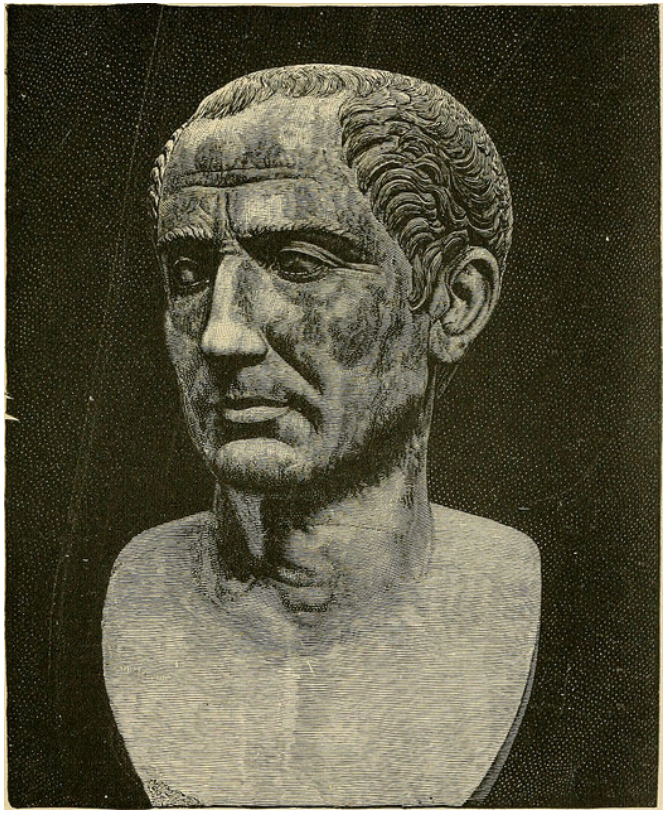
Standardization

By @Doug88888 (flickr.com)

- 2012: KECCAK selected by NIST for SHA-3
- 2014: 3GPP adopts KECCAK in **TUAK**
- 2015: NIST's **FIPS 202**
  - Of course: SHA3-{224, 256, 384, 512}
  - But also: SHAKE{128, 256}
- 2016: NIST's **SP 800-185**
  - cSHAKE
  - KMAC
  - TupleHash
  - ParallelHash

More designs building on KECCAK

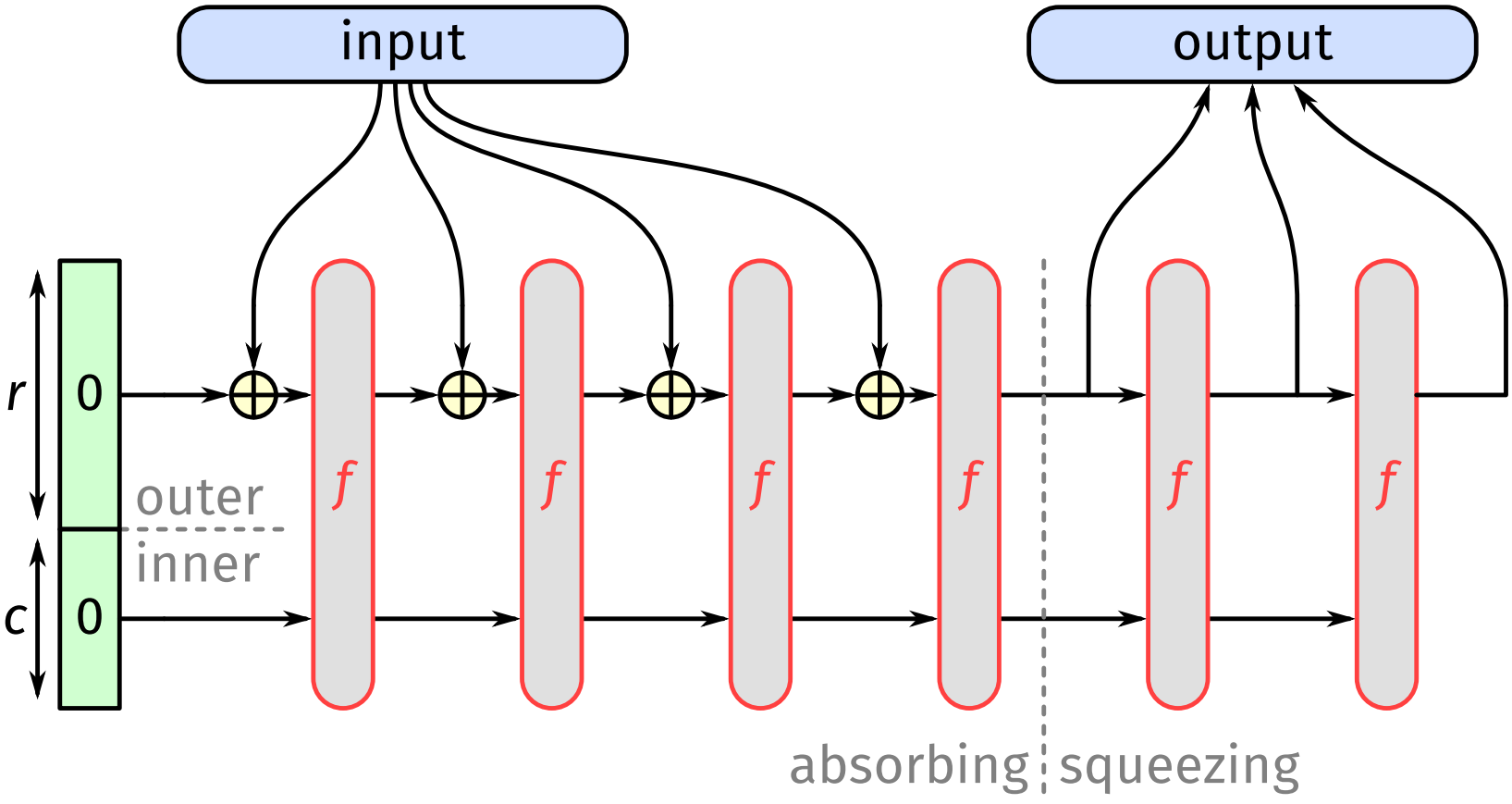
- 2014: KETJE and KEYAK
  - submitted to the CAESAR competition
- 2016: KANGAROOTWELVE
- 2017: KRAVATTE



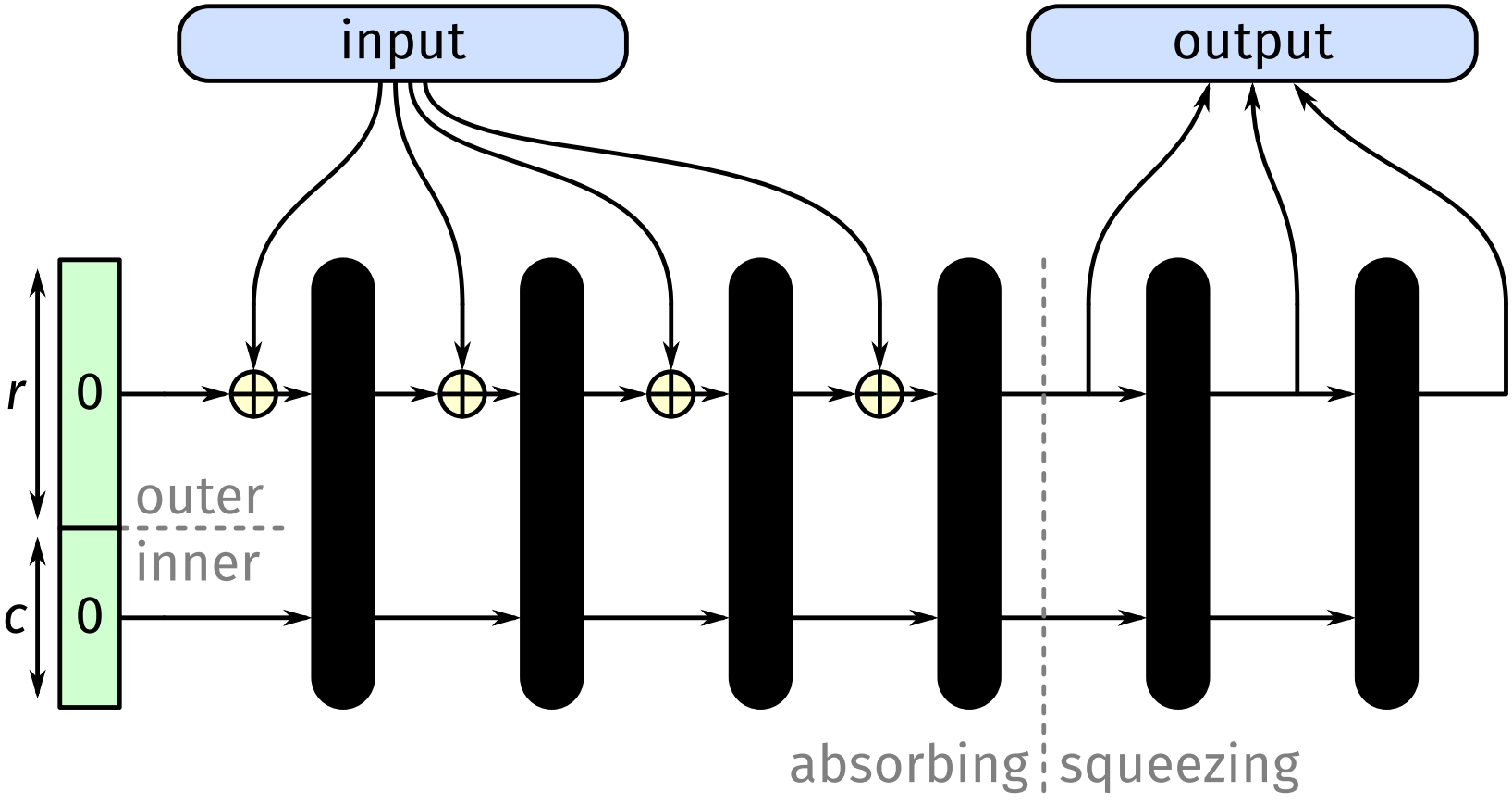
... again using the KECCAK-*f* permutation

Outline

Analyzing the sponge construction



Analyzing the sponge construction



Generic security of the sponge construction

Theorem 2. *A padded sponge construction calling a random permutation, $\mathcal{S}'[\mathcal{F}]$, is (t_D, t_S, N, ϵ)-indistinguishable from a random oracle, for any t_D, $t_S = O(N^2)$, $N < 2^c$ and and for any ϵ with $\epsilon > f_{\mathrm{P}}(N)$.*

If N is significantly smaller than 2^c, $f_P(N)$ can be approximated closely by:

$$f_P(N) \approx 1 - e^{-\frac{(1-2^{-r})N^2+(1+2^{-r})N}{2^{c+1}}} < \frac{(1-2^{-r})N^2+(1+2^{-r})N}{2^{c+1}}. \qquad (6)$$

[EuroCrypt 2008]

Theorem, explained

$$\Pr[\text{attack}] \leq \frac{N^2}{2^{c+1}} \text{ (or so)}$$

$\Rightarrow$ if $N \ll 2^{c/2}$, then the probability is negligible

Generic security of the sponge construction

Theorem 2. *A padded sponge construction calling a random permutation, $\mathcal{S}'[\mathcal{F}]$, is (t_D, t_S, N, ϵ)-indistinguishable from a random oracle, for any t_D, $t_S = O(N^2)$, $N < 2^c$ and and for any ϵ with $\epsilon > f_P(N)$.*

If N is significantly smaller than 2^c, $f_P(N)$ can be approximated closely by:

$$f_P(N) \approx 1 - e^{-\frac{(1-2^{-r})N^2+(1+2^{-r})N}{2^{c+1}}} < \frac{(1-2^{-r})N^2+(1+2^{-r})N}{2^{c+1}}. \qquad (6)$$

[EuroCrypt 2008]

Theorem, explained

$$\Pr[\text{attack}] \leq \frac{N^2}{2^{c+1}} \text{ (or so)}$$

$\Rightarrow$ if $N \ll 2^{c/2}$, then the probability is negligible

Two pillars of security in cryptography

- Generic security
  - Strong mathematical proofs
    $\Rightarrow$ scope of cryptanalysis reduced to primitive
- Security of the primitive
  - No proof!
    $\Rightarrow$ open design rationale
    $\Rightarrow$ **cryptanalysis!**
  - Confidence
    $\Leftarrow$ sustained cryptanalysis activity and no break
    $\Leftarrow$ proven properties

Two pillars of security in cryptography

- Generic security
  - Strong mathematical proofs
    $\Rightarrow$ scope of cryptanalysis reduced to primitive
- Security of the primitive
  - No proof!
    $\Rightarrow$ open design rationale
    $\Rightarrow$ **cryptanalysis!**
  - Confidence
    $\Leftarrow$ sustained cryptanalysis activity and no break
    $\Leftarrow$ proven properties

Two pillars of security in cryptography

- Generic security
  - Strong mathematical proofs
    $\Rightarrow$ scope of cryptanalysis reduced to primitive
- Security of the primitive
  - No proof!
    $\Rightarrow$ open design rationale
    $\Rightarrow$ **cryptanalysis!**
  - Confidence
    $\Leftarrow$ sustained cryptanalysis activity and no break
    $\Leftarrow$ proven properties

Two pillars of security in cryptography

- Generic security
  - Strong mathematical proofs
    $\Rightarrow$ scope of cryptanalysis reduced to primitive
- Security of the primitive
  - No proof!
    $\Rightarrow$ open design rationale
    $\Rightarrow$ **cryptanalysis!**
  - Confidence
    $\Leftarrow$ sustained cryptanalysis activity and no break
    $\Leftarrow$ proven properties

Two pillars of security in cryptography

- Generic security
  - Strong mathematical proofs
    $\Rightarrow$ scope of cryptanalysis reduced to primitive
- Security of the primitive
  - No proof!
    $\Rightarrow$ open design rationale
    $\Rightarrow$ **cryptanalysis!**
  - Confidence
    $\Leftarrow$ sustained cryptanalysis activity and no break
    $\Leftarrow$ proven properties

Two pillars of security in cryptography

- Generic security
  - Strong mathematical proofs
    $\Rightarrow$ scope of cryptanalysis reduced to primitive
- Security of the primitive
  - No proof!
    $\Rightarrow$ open design rationale
    $\Rightarrow$ third-party **cryptanalysis!**
  - Confidence
    $\Leftarrow$ sustained cryptanalysis activity and no break
    $\Leftarrow$ proven properties

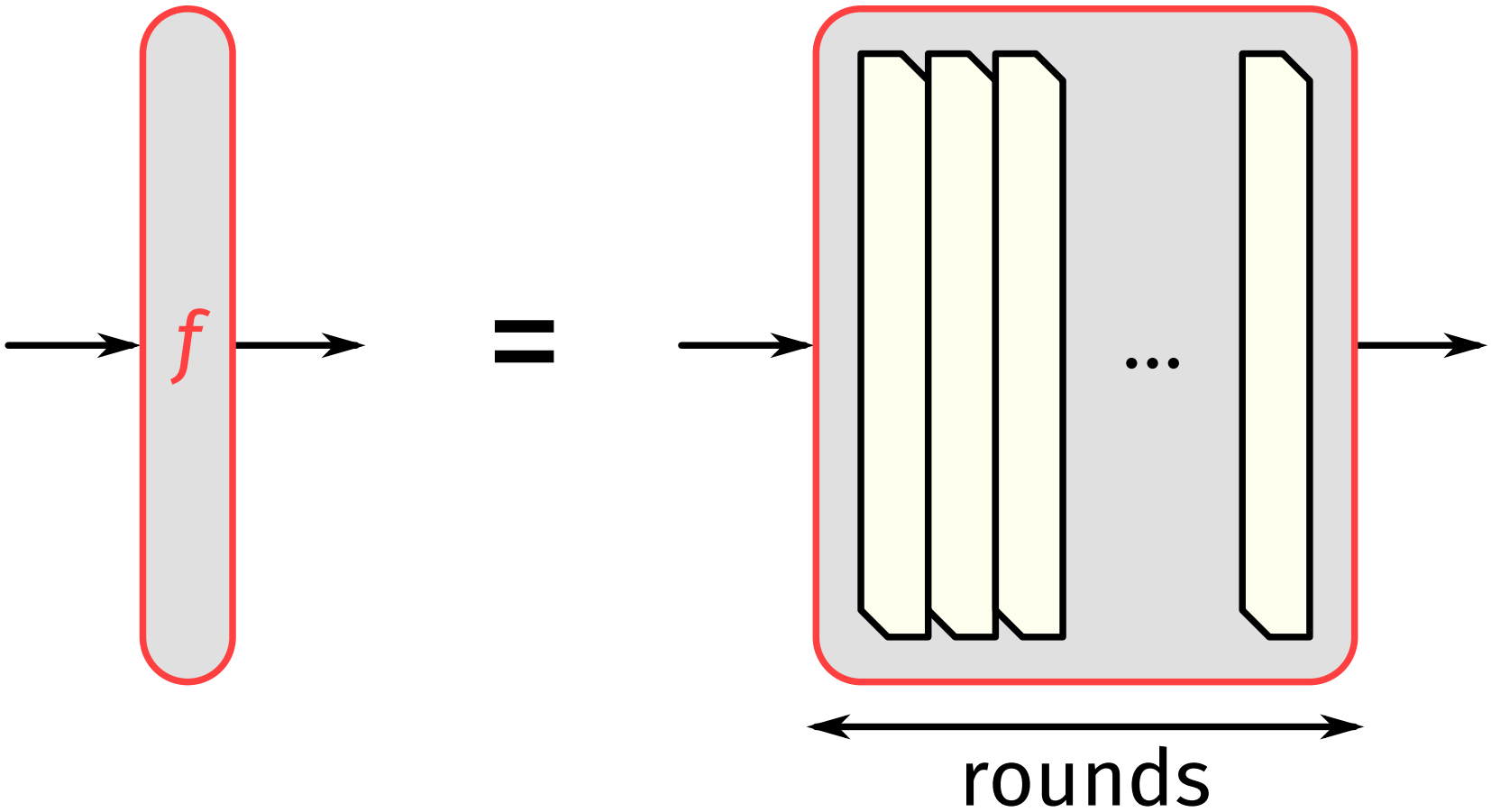
Two pillars of security in cryptography

- Generic security
  - Strong mathematical proofs
    $\Rightarrow$ scope of cryptanalysis reduced to primitive
- Security of the primitive
  - No proof!
    $\Rightarrow$ open design rationale
    $\Rightarrow$ lots of third-party **cryptanalysis!**
  - Confidence
    $\Leftarrow$ sustained cryptanalysis activity and no break
    $\Leftarrow$ proven properties

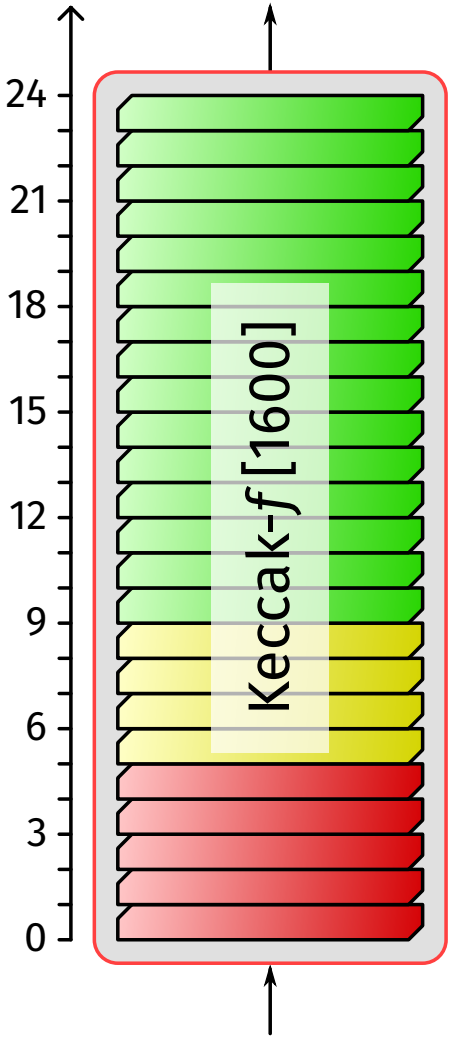
Two pillars of security in cryptography

- Generic security
  - Strong mathematical proofs
    $\Rightarrow$ scope of cryptanalysis reduced to primitive
- Security of the primitive
  - No proof!
    $\Rightarrow$ open design rationale
    $\Rightarrow$ lots of third-party **cryptanalysis!**
  - Confidence
    $\Leftarrow$ sustained cryptanalysis activity and no break
    $\Leftarrow$ proven properties

Inside the permutation



Status of KECCAK



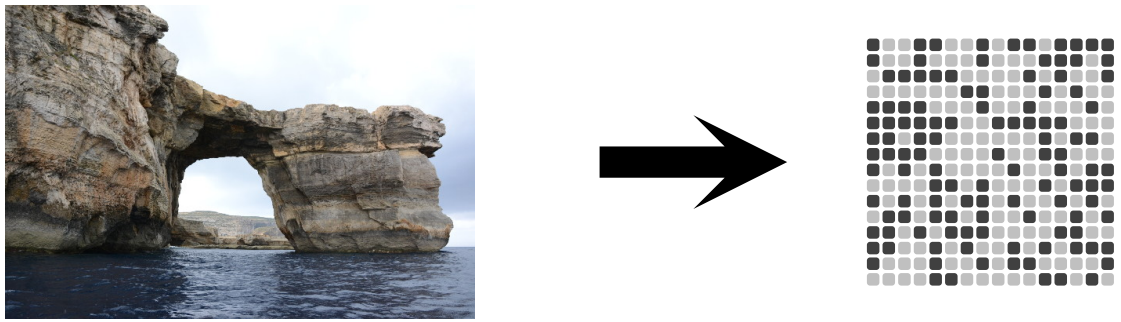
- Practical (collision) attacks up to 5 rounds
- Theoretical collision attacks up to 6 rounds
  [Qiao, Song, Liu, Guo 2016]
- Theoretical attack up to 9 rounds (2^{256} time...)
  [Dinur, Morawiecki, Pieprzyk, Srebrny, Straus 2014]

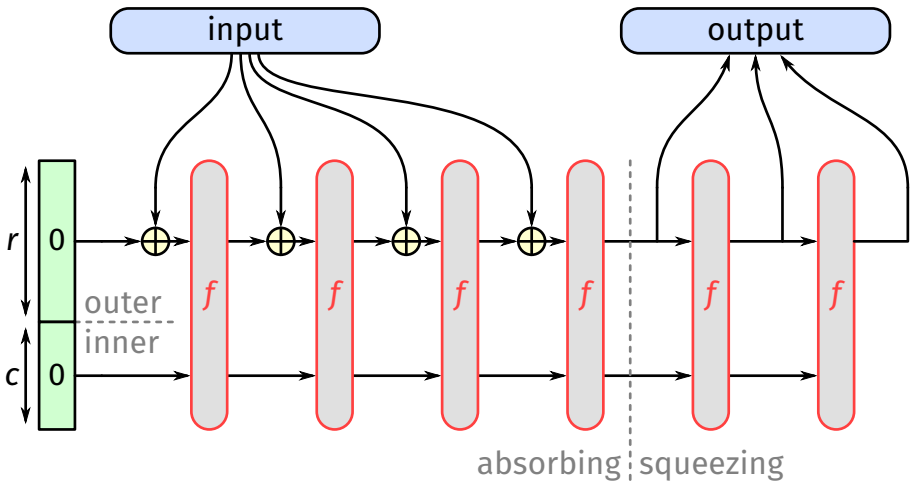
Round function unchanged since 2008

Outline

Cryptographic hash functions

$$h : \{0,1\}^* \rightarrow \{0,1\}^n$$



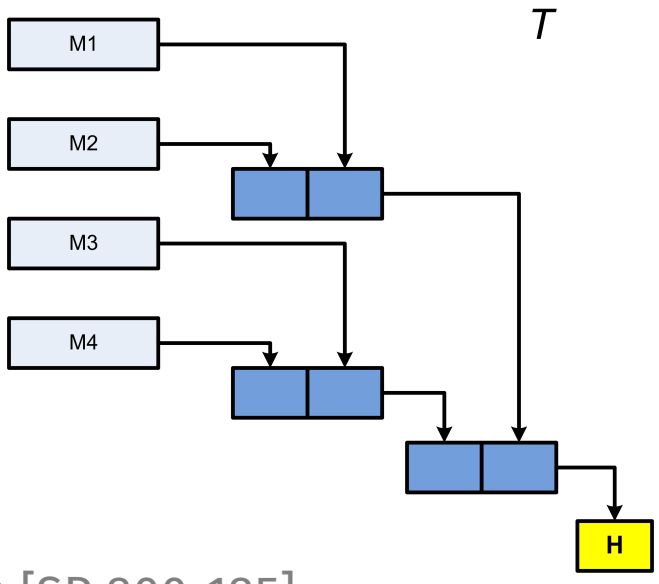


Impact of parallelism

KECCAK-$f[1600] \times 1$	1070 cycles
KECCAK-$f[1600] \times 2$	1360 cycles
KECCAK-$f[1600] \times 4$	1410 cycles

CPU: Intel® Core™ i5-6500 (Skylake) with AVX2 256-bit SIMD

Tree hashing



Example: **ParallelHash** [SP 800-185]

function	instruction set	cycles/byte
KECCAK$[c = 256] \times 1$	x86_64	6.29
KECCAK$[c = 256] \times 2$	AVX2	4.32
KECCAK$[c = 256] \times 4$	AVX2	2.31

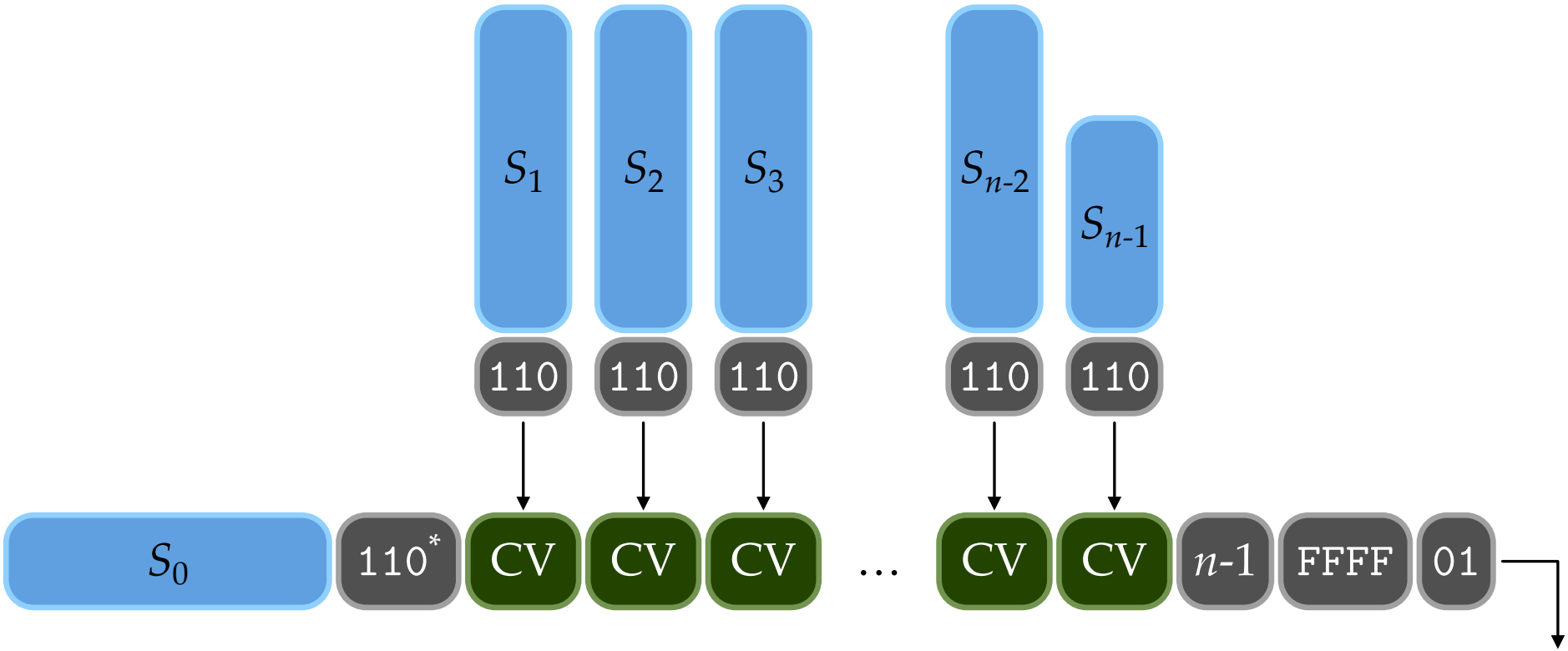
CPU: Intel® Core™ i5-6500 (Skylake) with AVX2 256-bit SIMD

KangarooTwelve: a variant of Keccak

- Safety margin: from *rock-solid* to *comfortable*
  - Same round function, 12 instead of 24
    $\Rightarrow$ cryptanalysis since 2008 still valid
- "Embarassingly" parallel mode
  - Proven generic security

[IACR ePrint 2016/770]

KangarooTwelve's mode



Final node growing with kangaroo hopping and SAKURA coding

[ACNS 2014]

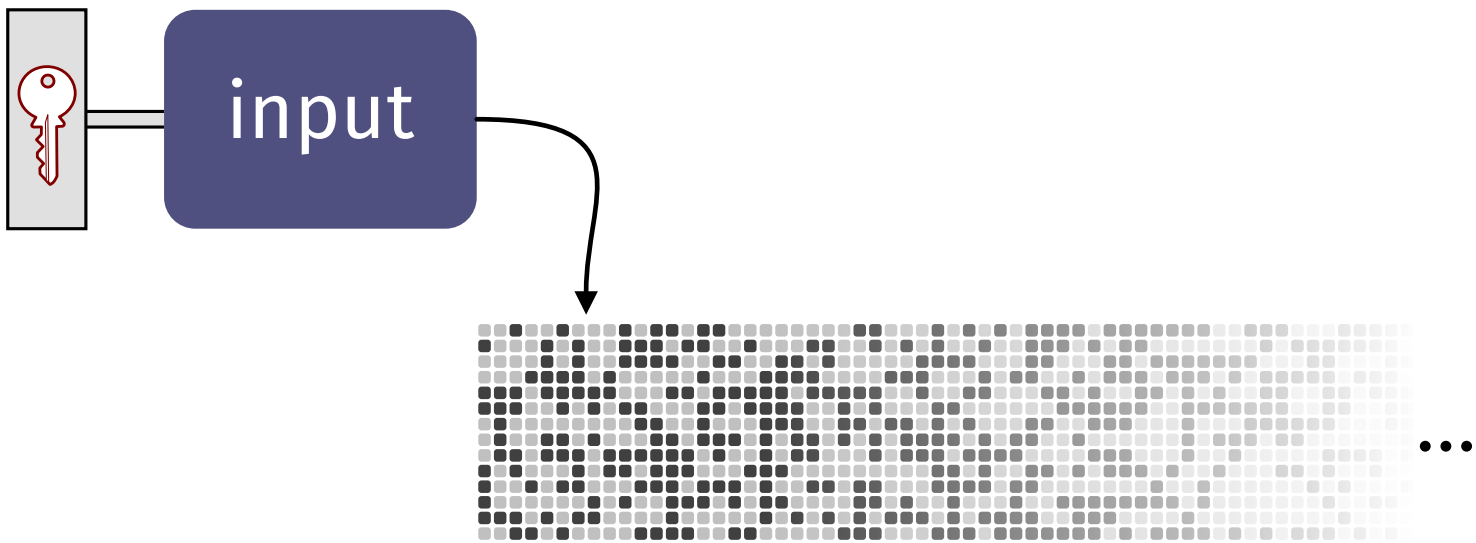
KangarooTwelve performance

	Short input	Long input
Intel® Core™ i5-4570 (Haswell)	4.15 c/b	1.44 c/b
Intel® Core™ i5-6500 (Skylake)	3.72 c/b	1.22 c/b
Intel® Xeon Phi™ 7250 (Knights Landing)*	(4.56 c/b)	0.74 c/b

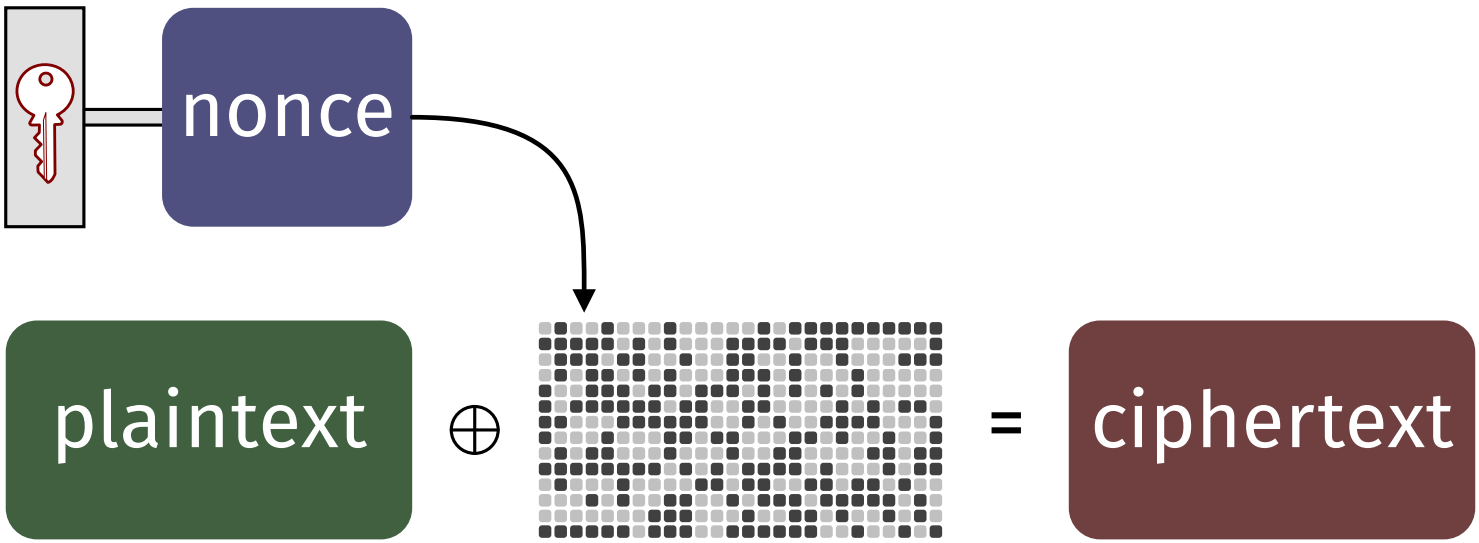
* Thanks to Romain Dolbeau

Outline

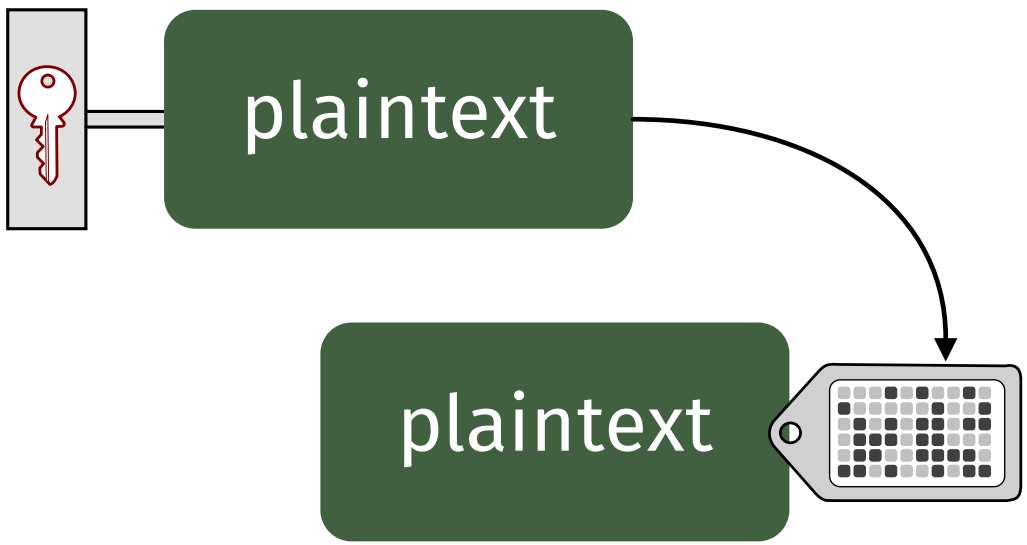
Pseudo-random function (PRF)



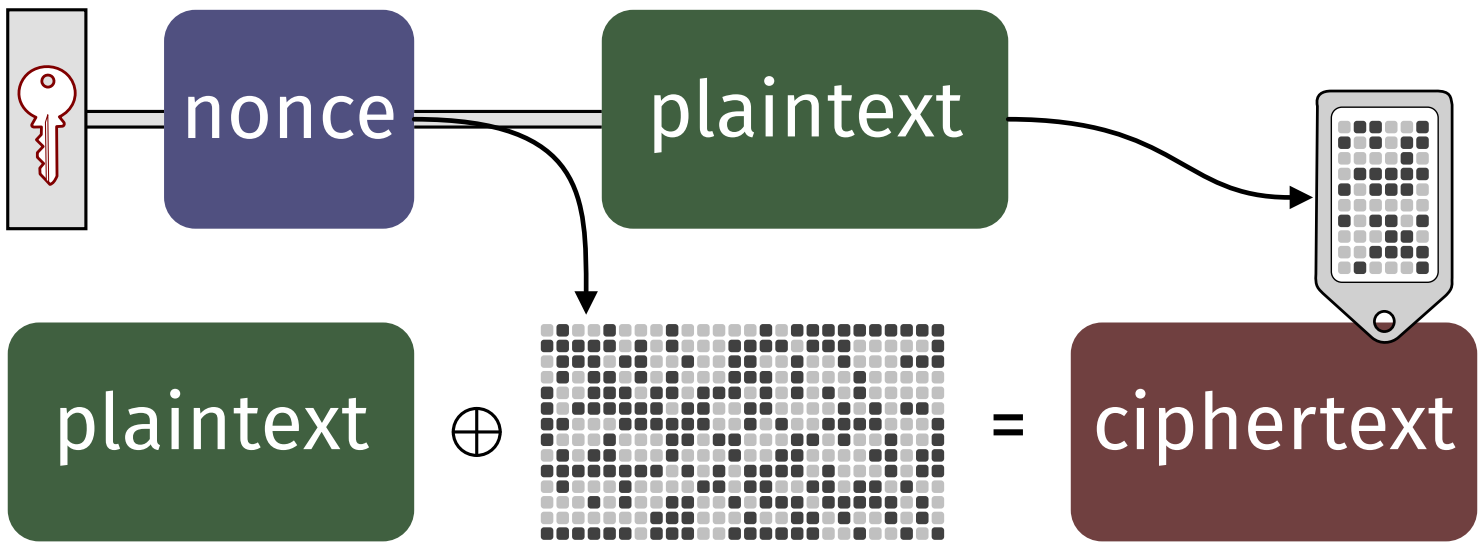
Stream cipher



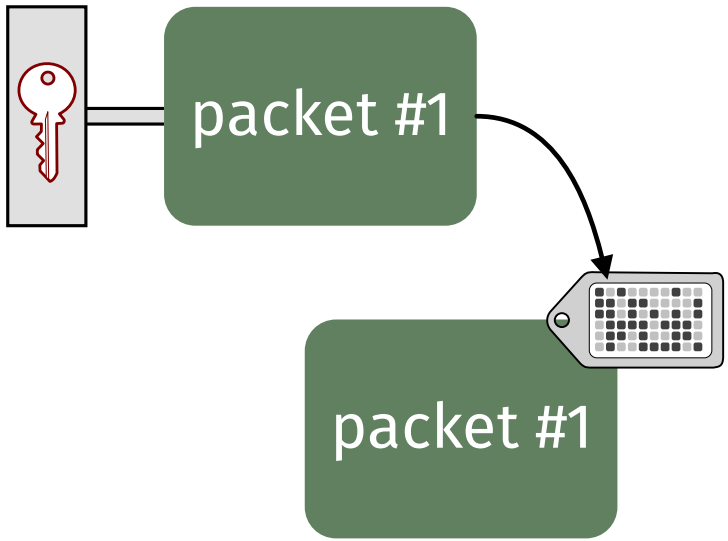
Message authentication code (MAC)



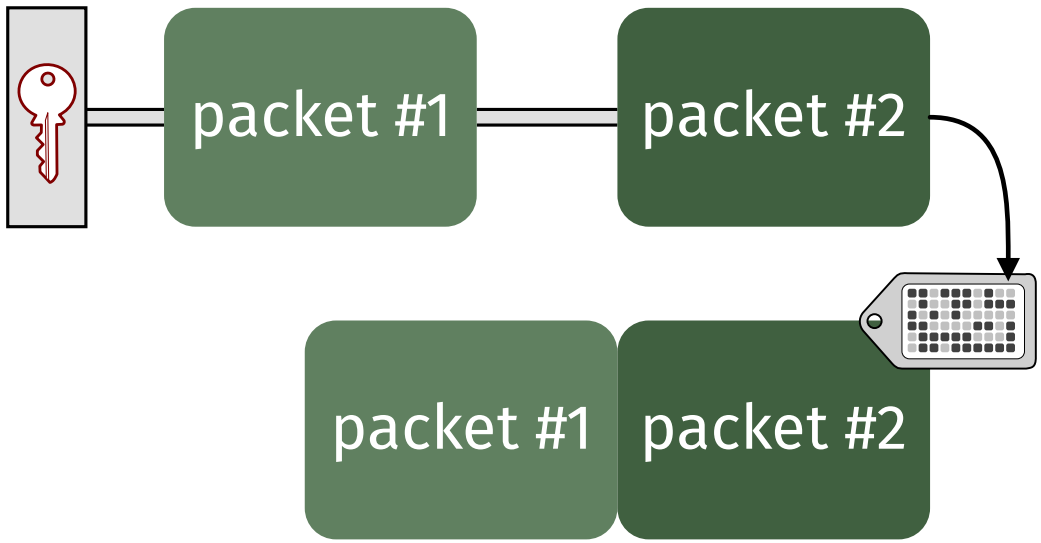
Authenticated encryption



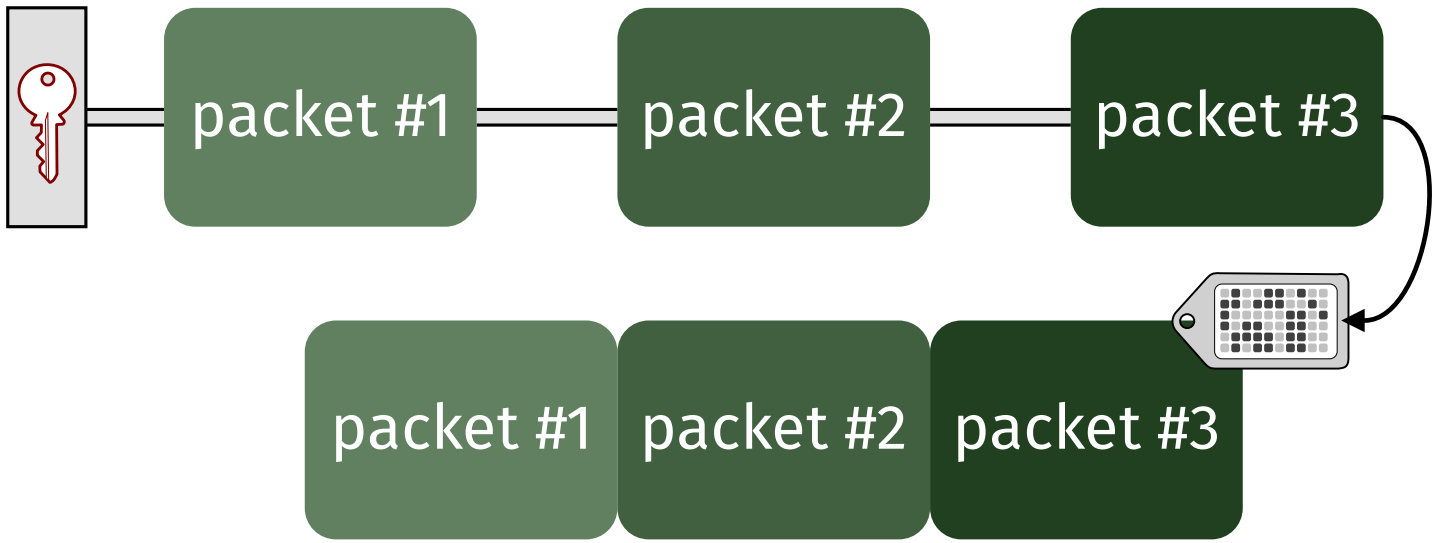
Incrementality



Incrementality



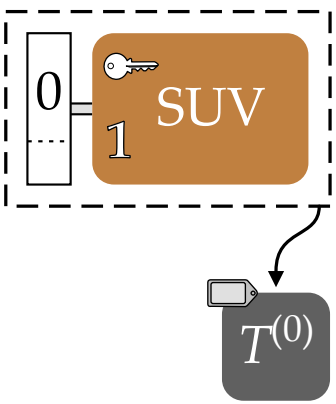
Incrementality



KEYAK in a nutshell

An authenticated-encryption scheme submitted to CAESAR

$\rightarrow$ using KECCAK-$p[1600, n_r = 12]$ or KECCAK-$p[800, n_r = 12]$ $\leftarrow$

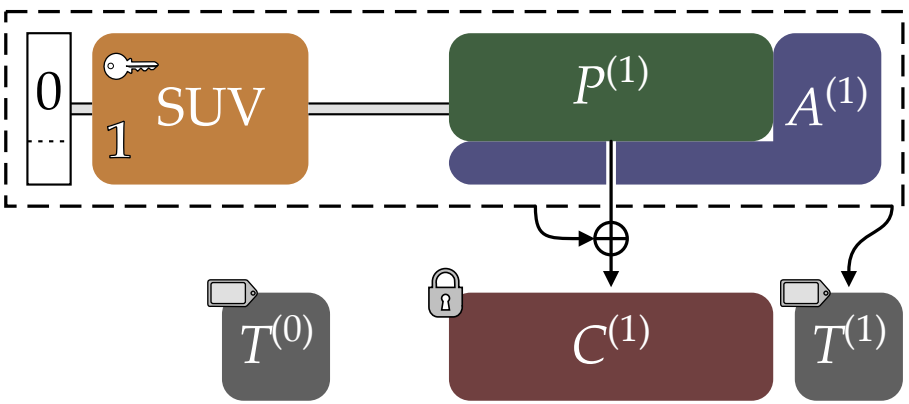


- SUV = Secret and Unique Value
- Works in *sessions*

KEYAK in a nutshell

An authenticated-encryption scheme submitted to CAESAR

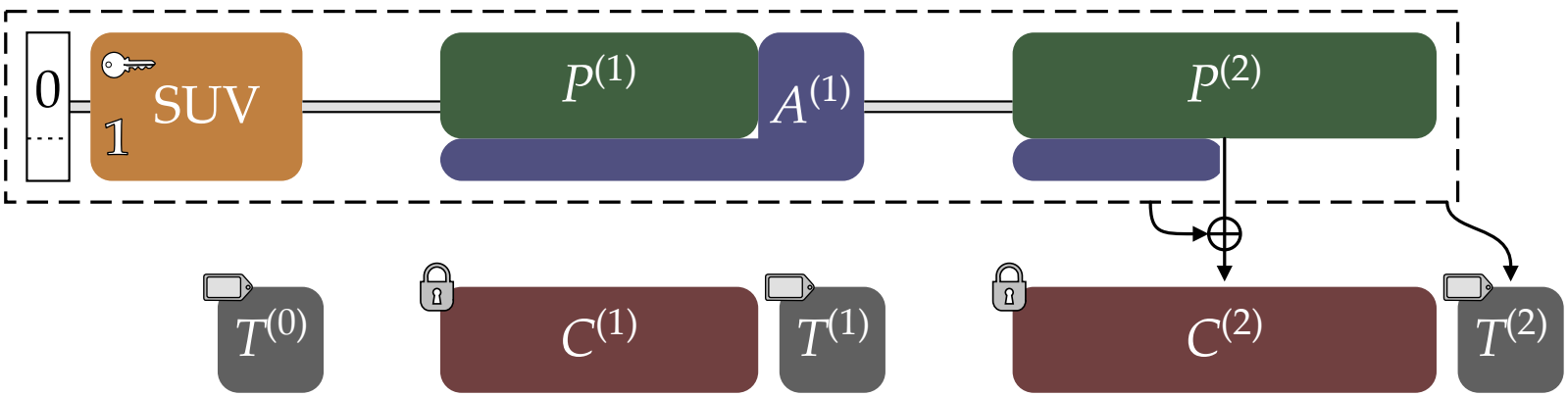
$\rightarrow$ using KECCAK-$p[1600, n_r = 12]$ or KECCAK-$p[800, n_r = 12]$ $\leftarrow$



- SUV = Secret and Unique Value
- Works in *sessions*

KEYAK in a nutshell

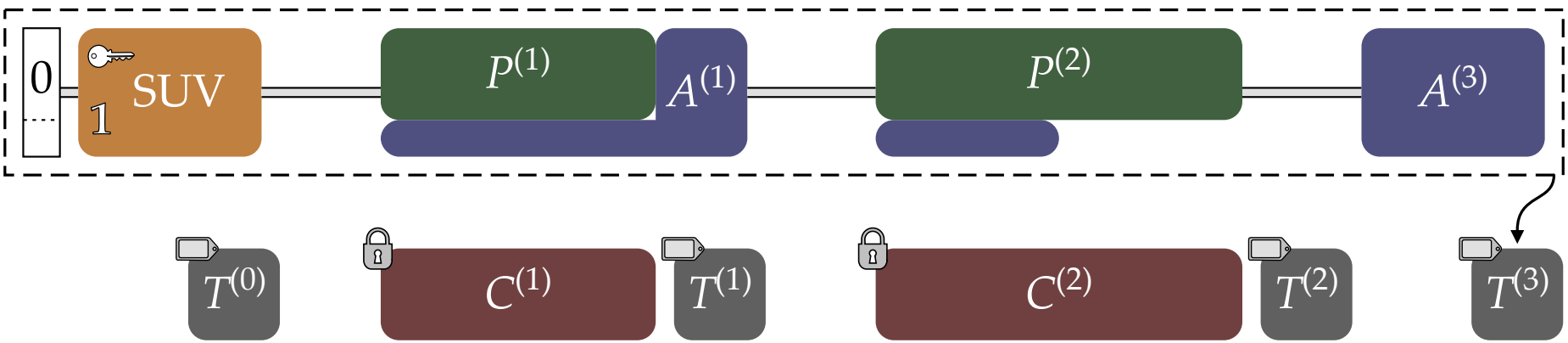
An authenticated-encryption scheme submitted to CAESAR
→ using KECCAK-$p[1600, n_r = 12]$ or KECCAK-$p[800, n_r = 12]$ ←



- SUV = Secret and Unique Value
- Works in *sessions*

KEYAK in a nutshell

An authenticated-encryption scheme submitted to CAESAR
→ using KECCAK-$p[1600, n_r = 12]$ or KECCAK-$p[800, n_r = 12]$ ←



- SUV = Secret and Unique Value
- Works in *sessions*

KETJE in a nutshell

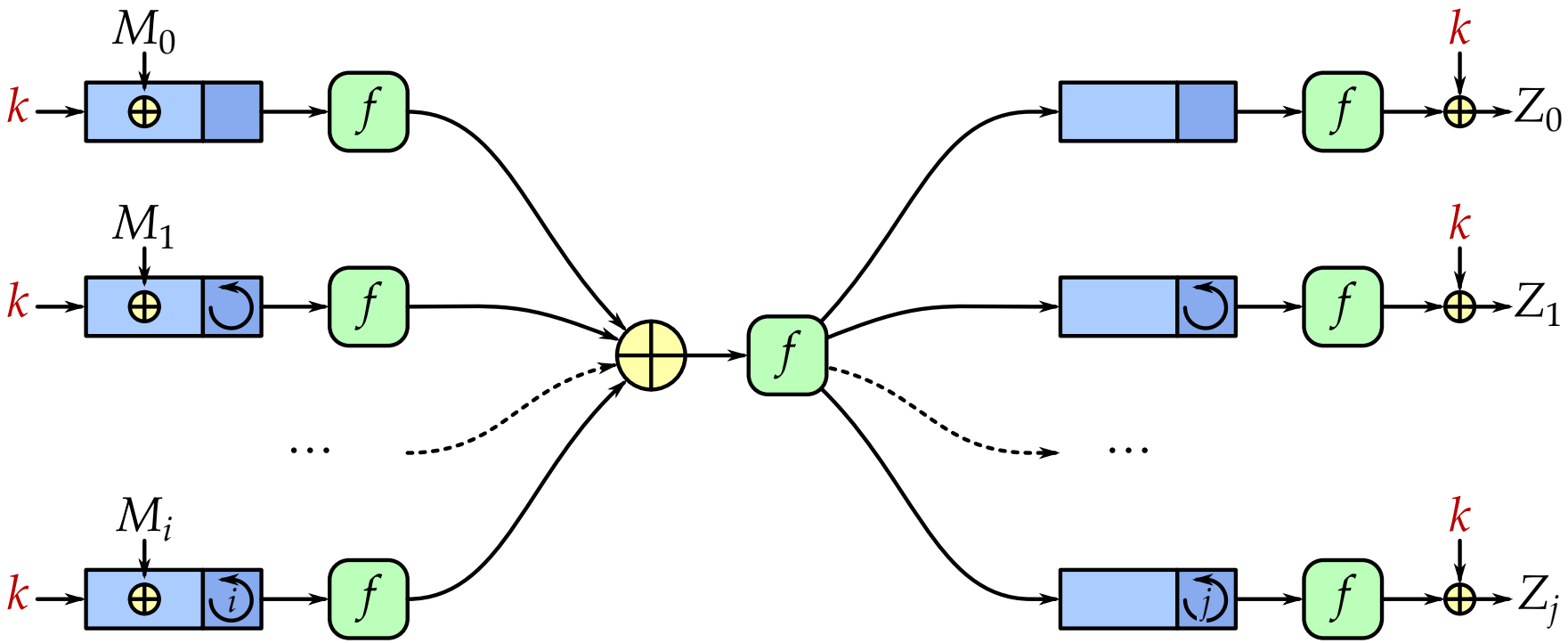
An authenticated-encryption scheme submitted to CAESAR

Similar to KEYAK, but ...

- mainly targeted at lightweight applications (e.g., IoT)
- also using smaller permutations (400 or 200 bits)

KRAVATTE with the new Farfalle construction

An incremental and parallel pseudo-random function …



… using the KECCAK-f permutation

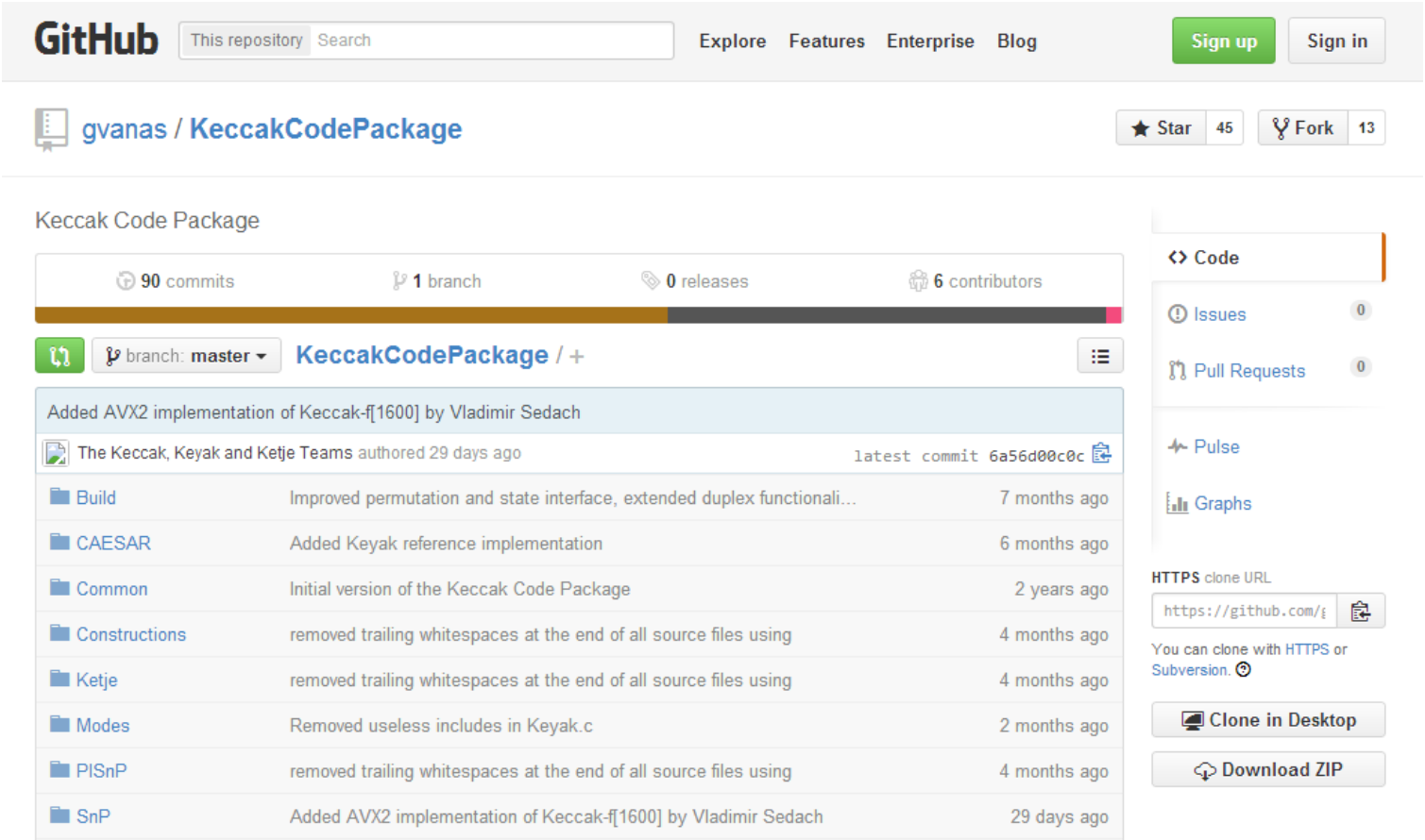
[IACR ePrint 2016/1188 — soon to be updated!]

KRAVATTE for many purposes

KRAVATTE-PRF	Authentication
KRAVATTE-SAE	Session authenticated encryption
KRAVATTE-SIV	Synthetic-IV authenticated encryption
KRAVATTE-WBC	Wide block cipher, authenticated encryption with minimal expansion

Outline

The KECCAK code package



https://github.com/gvanas/KeccakCodePackage

Using the KCP

- Making a library
  - `make generic64/libkeccak.a`
  - `make generic32/libkeccak.a`
  - `make Nehalem/libkeccak.a`
  - `make ARMv7A/libkeccak.a`
  - `make compact/libkeccak.a`
- Extracting the source files
  - `make generic64/libkeccak.a.pack`
- Running the unit tests
  - `make generic64/KeccakTests`
  - `KeccakTests --SnP --FIPS202 --Keyak --speed`

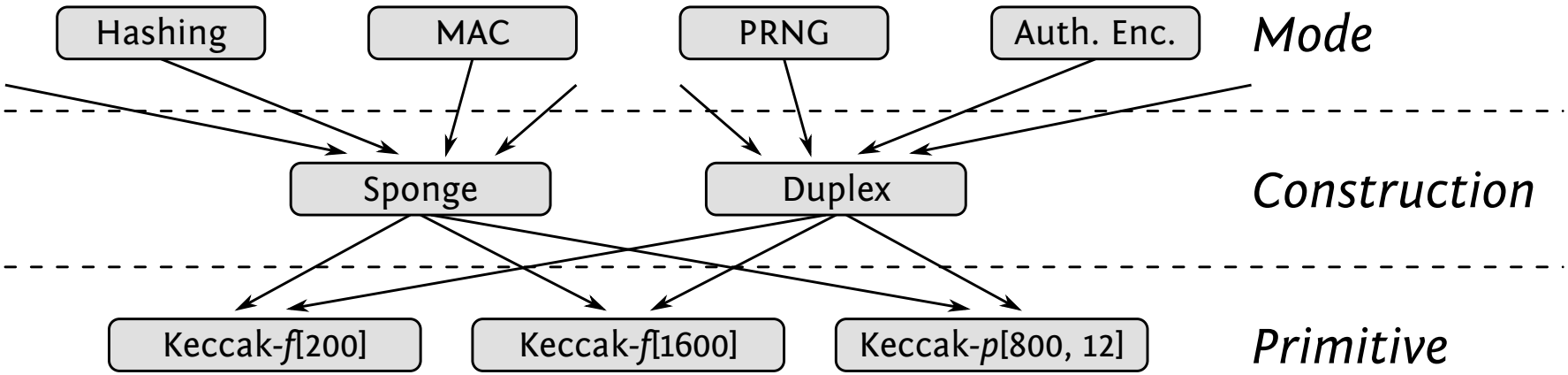
Using the KCP

- Making a library
  - `make generic64/libkeccak.a`
  - `make generic32/libkeccak.a`
  - `make Nehalem/libkeccak.a`
  - `make ARMv7A/libkeccak.a`
  - `make compact/libkeccak.a`
- Extracting the source files
  - `make generic64/libkeccak.a.pack`
- Running the unit tests
  - `make generic64/KeccakTests`
  - `KeccakTests --SnP --FIPS202 --Keyak --speed`

Using the KCP

- Making a library
  - `make generic64/libkeccak.a`
  - `make generic32/libkeccak.a`
  - `make Nehalem/libkeccak.a`
  - `make ARMv7A/libkeccak.a`
  - `make compact/libkeccak.a`
- Extracting the source files
  - `make generic64/libkeccak.a.pack`
- Running the unit tests
  - `make generic64/KeccakTests`
  - `KeccakTests --SnP --FIPS202 --Keyak --speed`

Inside the KCP: a layered approach



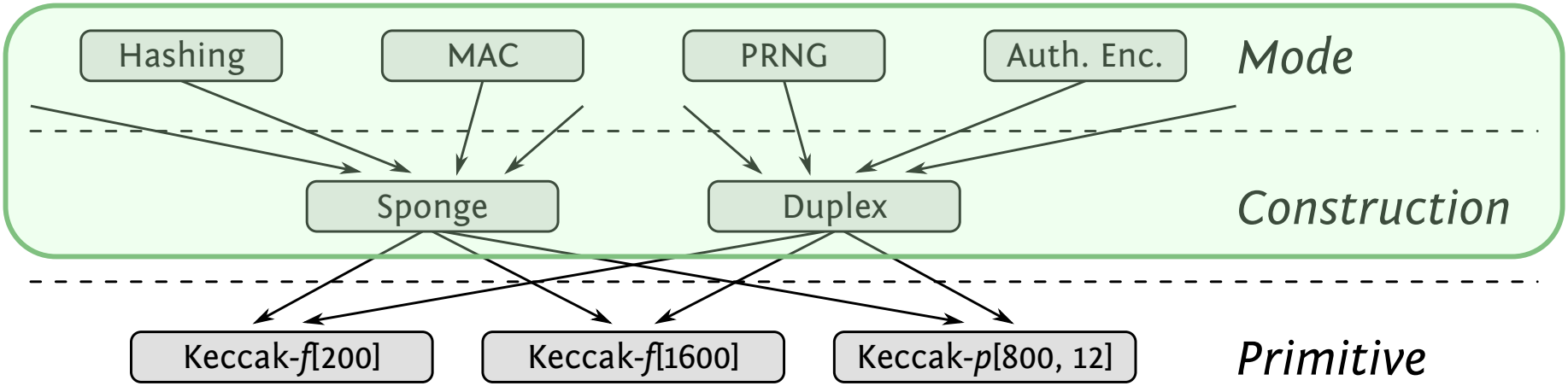
Generic

- focus on **user**
  - easy to use
  - e.g., message queue
- one implementation
  - pointers and arithmetic

Specific

- focus on **developer**
  - limited scope to optimize
  - unit tests
- tailored implementations
  - permutation
  - bulk data processing

Inside the KCP: a layered approach



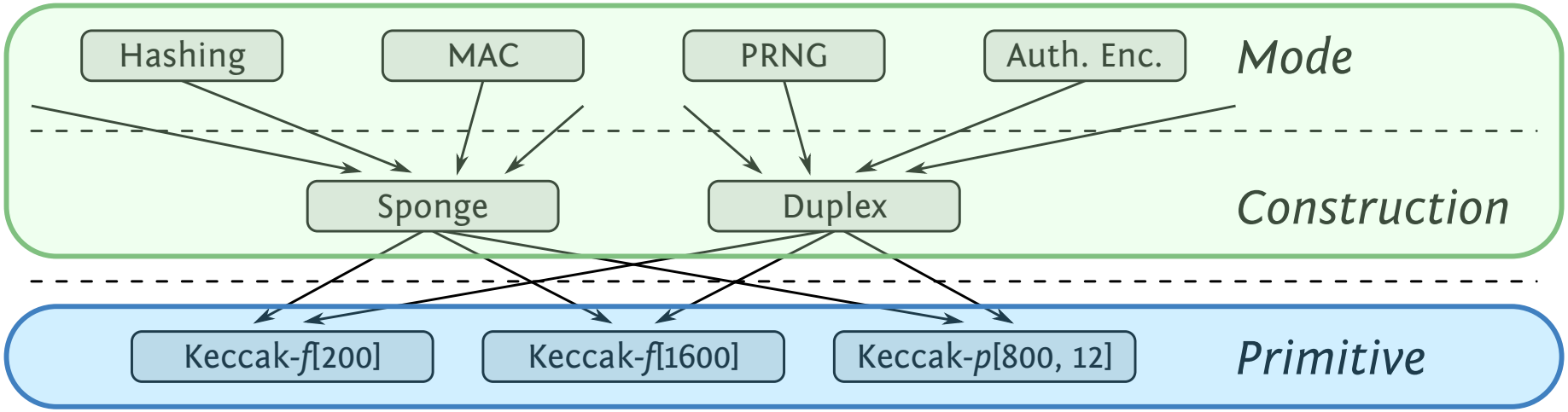
Generic

- focus on **user**
  - easy to use
  - e.g., message queue
- one implementation
  - pointers and arithmetic

Specific

- focus on **developer**
  - limited scope to optimize
  - unit tests
- tailored implementations
  - permutation
  - bulk data processing

Inside the KCP: a layered approach



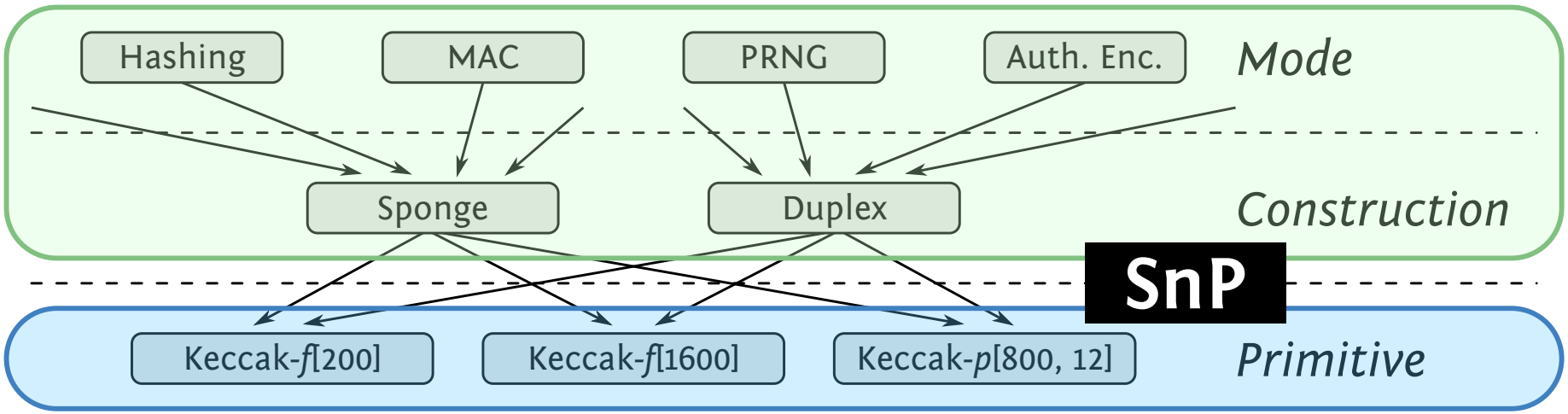
Generic

- focus on **user**
  - easy to use
  - e.g., message queue
- one implementation
  - pointers and arithmetic

Specific

- focus on **developer**
  - limited scope to optimize
  - unit tests
- tailored implementations
  - permutation
  - bulk data processing

Inside the KCP: a layered approach



Generic

- focus on **user**
  - easy to use
  - e.g., message queue
- one implementation
  - pointers and arithmetic

Specific

- focus on **developer**
  - limited scope to optimize
  - unit tests
- tailored implementations
  - permutation
  - bulk data processing

Outline

Hash and extendable-output functions

Standard, rock-solid: [FIPS 202, SP 800-185]

- SHA3-{224, 256, 384, 512}
- SHAKE{128, 256} or **cSHAKE**{128, 256}
- TupleHash
- ParallelHash

Mature:

- KangarooTwelve [IACR ePrint 2016/770]

Pseudo-random number generation

Rock-solid:

- KeccakPRG [SAC 2011, implemented in KCP]
  - reseedable at any time
  - forward secrecy

Authentication

Standard, rock-solid:

- KMAC [SP 800-185]
- HMAC with SHA-3 is suboptimal!

Mature:

- KEYAK [CAESAR competition]

Cutting edge:

- KRAVATTE-PRF [IACR ePrint 2016/1188]

Authenticated encryption

Mature:

- KETJE
- KEYAK [CAESAR competition]

Cutting edge:

- KRAVATTE-SAE
- KRAVATTE-SIV
- KRAVATTE-WBC [IACR ePrint 2016/1188]

Thanks for your attention!

Any questions?

`http://keccak.noekeon.org/`

@KeccakTeam