# KangarooTwelve: fast hashing based on Keccak-*p*

Guido Bertoni[3] Joan Daemen[1,2] Michaël Peeters[1]
Gilles Van Assche[1] Ronny Van Keer[1] Benoît Viguier[2]

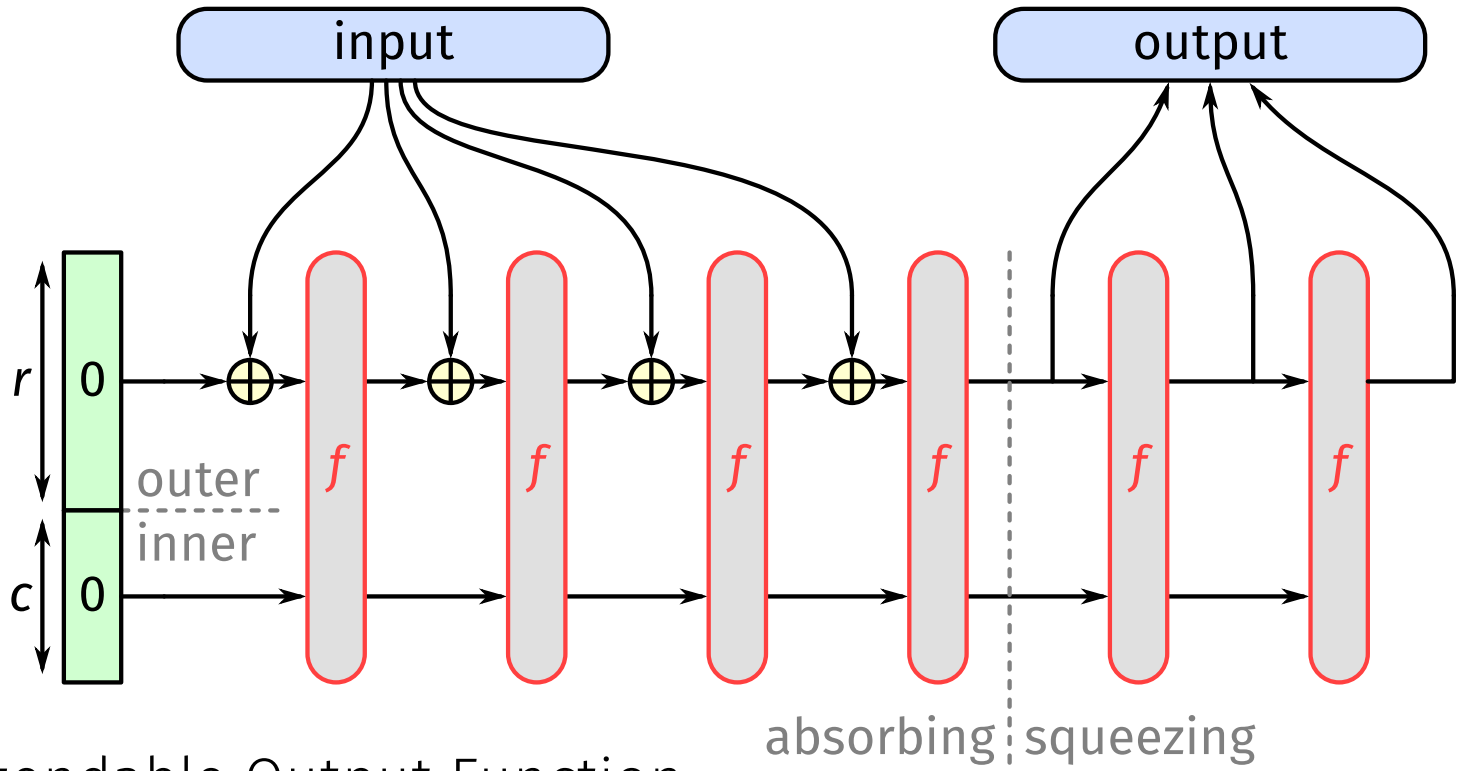
[1]STMicroelectronics
[2]Radboud University
[3]Security Pattern

The 16th International Conference on
Applied Cryptography and Network Security
Leuven, Belgium, July 2018

# Outline

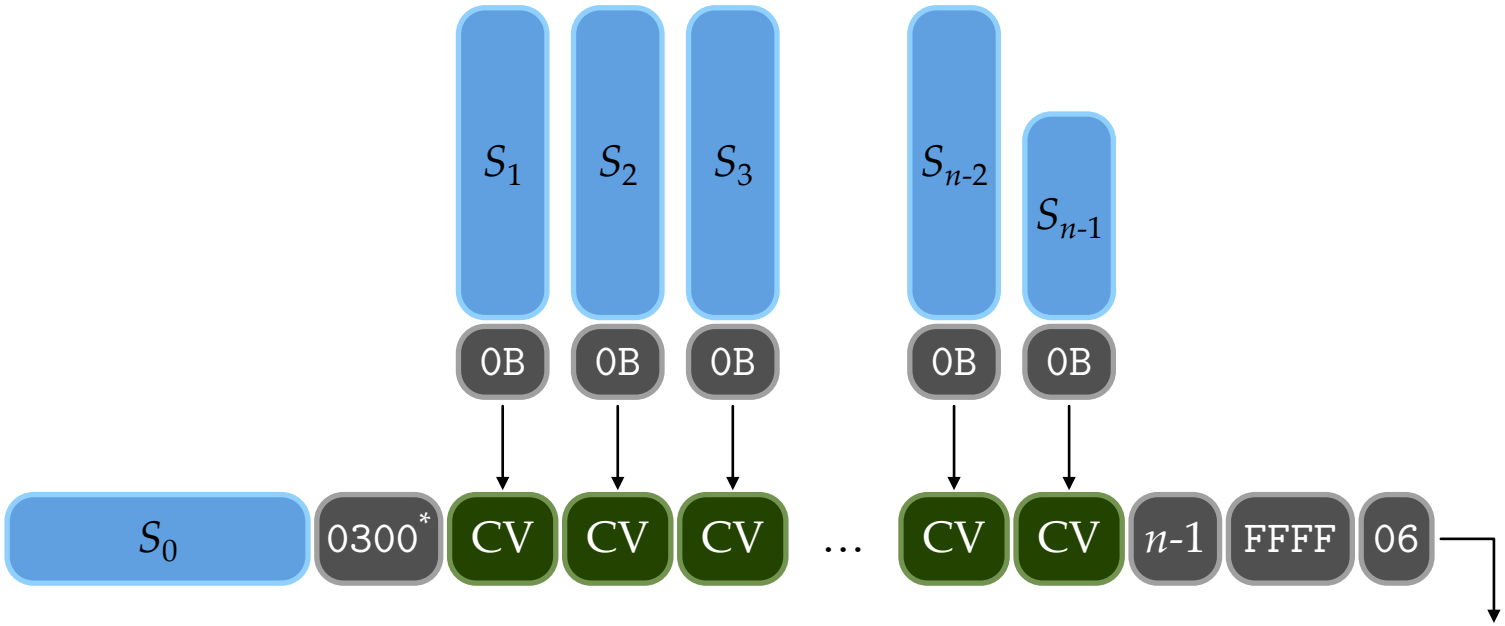
# Outline

# Let's start from SHAKE128



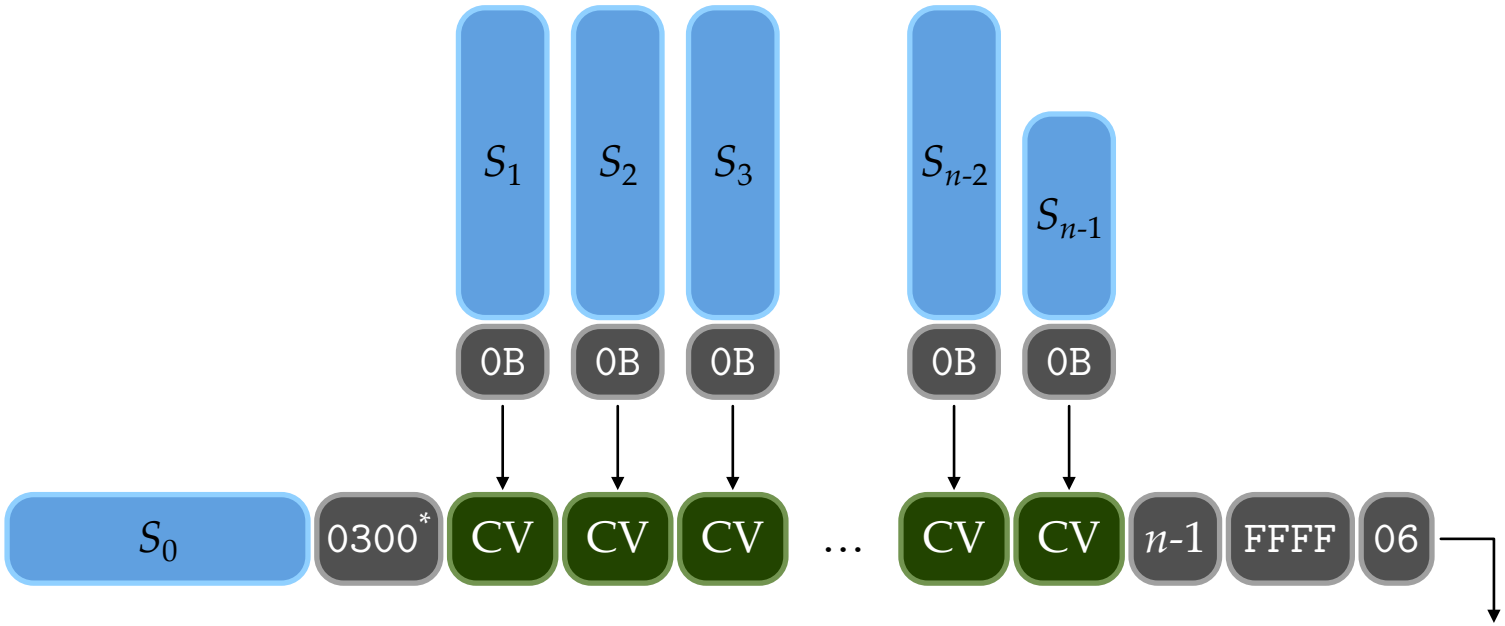

- eXtendable Output Function
- Sponge construction
- Uses Keccak-$p[1600, n_r = 24]$
- No parallelism at construction level

# From SHAKE128 to KangarooTwelve



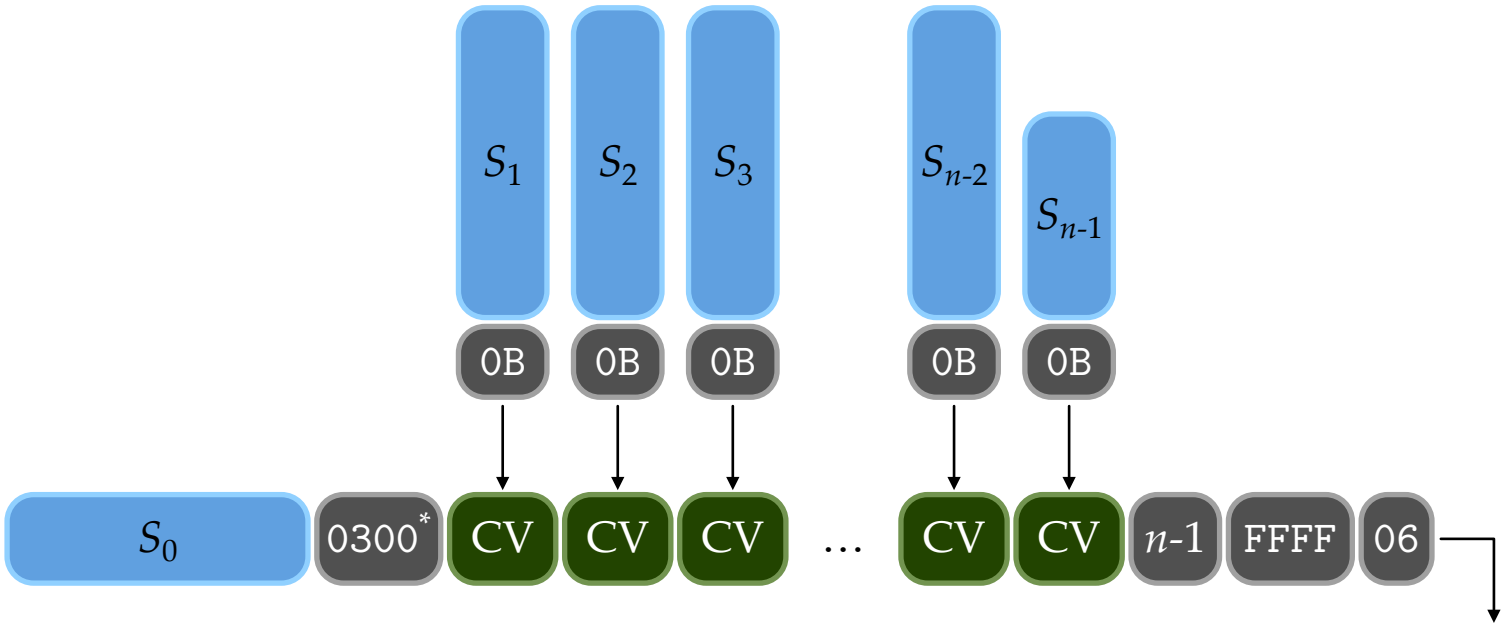

- eXtendable Output Function
- **Tree** on top of sponge construction
- Uses KeccaK-$p[1600, n_r = \mathbf{12}]$
- **Parallelism** grows automatically with input size

# From SHAKE128 to KangarooTwelve



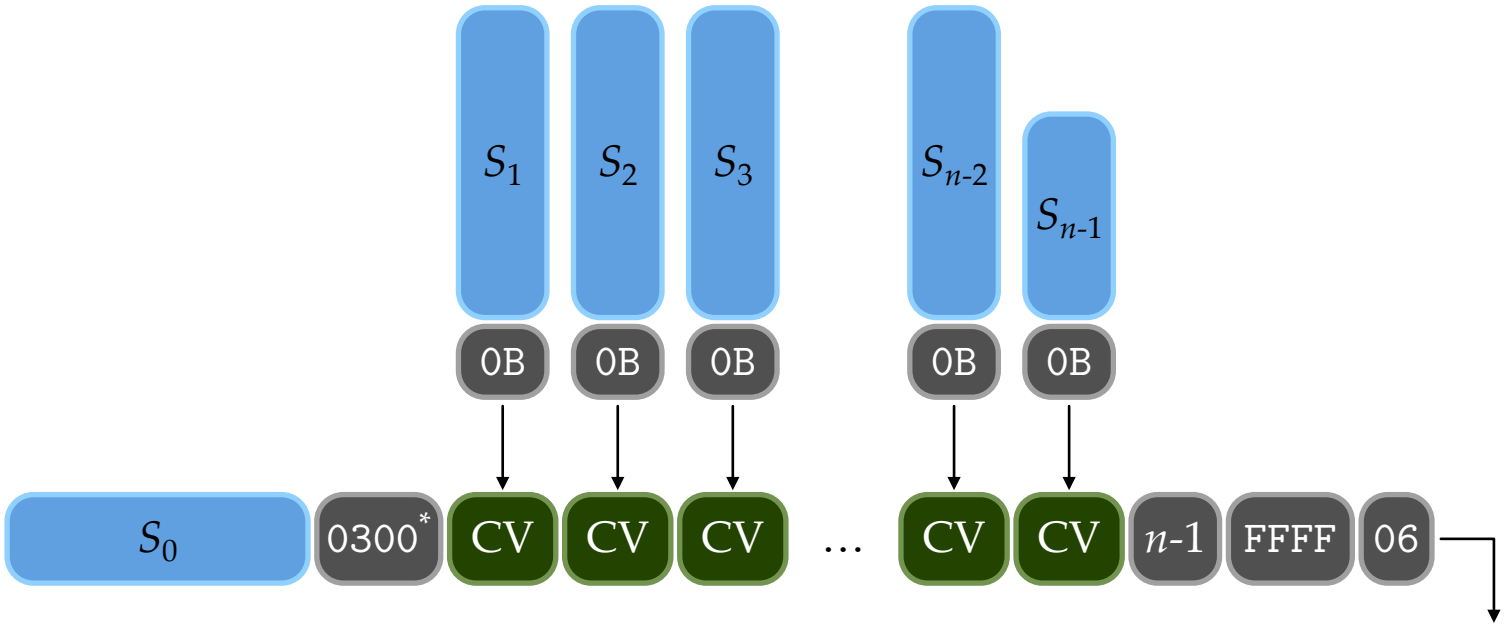

- eXtendable Output Function
- **Tree** on top of sponge const. +SAKURA coding
- Uses KECCAK-$p[1600, n_r = \mathbf{12}]$
- **Parallelism** grows automatically with input size

# From SHAKE128 to KangarooTwelve




- eXtendable Output Function
- **Tree** on top of sponge const. +SAKURA coding +kangaroo hopping
- Uses KECCAK-$p[1600, n_r = \mathbf{12}]$
- **Parallelism** grows automatically with input size

# From SHAKE128 to KangarooTwelve




- eXtendable Output Function
- **Tree** on top of sponge const. +SAKURA coding +kangaroo hopping
- Uses KECCAK-$p[1600, n_r = \mathbf{12}]$
- **Parallelism** grows automatically with input size (per 8KiB)

# Outline

# KangarooTwelve targets 128-bit security

- Flat sponge claim: 128-bit security strength
  - Collision resistance
  - (Second-) preimage resistance
    - **Multi-target** preimage resistance
    - Chosen-target forced-prefix preimage resistance
  - Correlation-freeness
  - Resistance against length-extension attacks
  - ...
- What about 256-bit security?
  - Philosophically much higher
  - But practically the same: well above the attacker's budget
  - MarsupilamiFourteen

# KangarooTwelve targets 128-bit security

- Flat sponge claim: 128-bit security strength
  - Collision resistance
  - (Second-) preimage resistance
    - **Multi-target** preimage resistance
    - Chosen-target forced-prefix preimage resistance
  - Correlation-freeness
  - Resistance against length-extension attacks
  - ...
- What about 256-bit security?
  - Philosophically much higher
  - But practically the same: well above the attacker's budget
  - MarsupilamiFourteen

# KangarooTwelve targets 128-bit security

- Flat sponge claim: 128-bit security strength
  - Collision resistance
  - (Second-) preimage resistance
    - **Multi-target** preimage resistance
    - Chosen-target forced-prefix preimage resistance
  - Correlation-freeness
  - Resistance against length-extension attacks
  - ...
- What about 256-bit security?
  - Philosophically much higher
  - But practically the same: well above the attacker's budget
  - MarsupilamiFourteen

# KangarooTwelve targets 128-bit security

- Flat sponge claim: 128-bit security strength
  - Collision resistance
  - (Second-) preimage resistance
    - **Multi-target** preimage resistance
    - Chosen-target forced-prefix preimage resistance
  - Correlation-freeness
  - Resistance against length-extension attacks
  - ...
- What about 256-bit security?
  - Philosophically much higher
  - But practically the same: well above the attacker's budget
  - MarsupilamiFourteen

# First pillar of security in symmetric cryptography

- Generic security
  - Strong mathematical proofs
    - ⇒ mode introduces no weaknesses
    - ⇒ scope of cryptanalysis focused on primitive
- In our case:
  [EuroCrypt 2008] – On the Indifferentiability of the Sponge Construction
  [IJIS 2014] – Sufficient conditions for sound tree and sequential hashing modes
  [ACNS 2014] – SAKURA: A Flexible Coding for Tree Hashing

# First pillar of security in symmetric cryptography

- Generic security
  - Strong mathematical proofs
    $\Rightarrow$ mode introduces no weaknesses
    $\Rightarrow$ scope of cryptanalysis focused on primitive
- In our case:
  [EuroCrypt 2008] – On the Indifferentiability of the Sponge Construction
  [IJIS 2014] – Sufficient conditions for sound tree and sequential hashing modes
  [ACNS 2014] – SAKURA: A Flexible Coding for Tree Hashing

# First pillar of security in symmetric cryptography

- Generic security
  - Strong mathematical proofs
    $\Rightarrow$ mode introduces no weaknesses
    $\Rightarrow$ scope of cryptanalysis focused on primitive
- In our case:
  [EuroCrypt 2008] – On the Indifferentiability of the Sponge Construction
  [IJIS 2014] – Sufficient conditions for sound tree and sequential hashing modes
  [ACNS 2014] – SAKURA: A Flexible Coding for Tree Hashing

# Second pillar of security in symmetric cryptography

- Security of the primitive
  - No proof!
    - $\Rightarrow$ publicly documented design rationale
    - $\Rightarrow$ **cryptanalysis!**
- In our case:
  - Ten years of cryptanalysis on (reduced-round) KECCAK-$f$[1600]
    - $\Leftarrow$ tune the number of rounds
    - $\Leftarrow$ no tweak!

# Second pillar of security in symmetric cryptography

- Security of the primitive
  - No proof!
    $\Rightarrow$ publicly documented design rationale
    $\Rightarrow$ **cryptanalysis!**
- In our case:
  - Ten years of cryptanalysis on (reduced-round) KECCAK-$f$[1600]
    $\Leftarrow$ tune the number of rounds
    $\Leftarrow$ no tweak!

# Second pillar of security in symmetric cryptography

- Security of the primitive
  - No proof!
    $\Rightarrow$ publicly documented design rationale
    $\Rightarrow$ **cryptanalysis!**
- In our case:
  - Ten years of cryptanalysis on (reduced-round) KECCAK-$f$[1600]
    $\Leftarrow$ tune the number of rounds
    $\Leftarrow$ no tweak!

# Second pillar of security in symmetric cryptography

- Security of the primitive
  - No proof!
    $\Rightarrow$ publicly documented design rationale
    $\Rightarrow$ third-party **cryptanalysis!**
- In our case:
  - Ten years of cryptanalysis on (reduced-round) KECCAK-$f$[1600]
    $\Leftarrow$ tune the number of rounds
    $\Leftarrow$ no tweak!

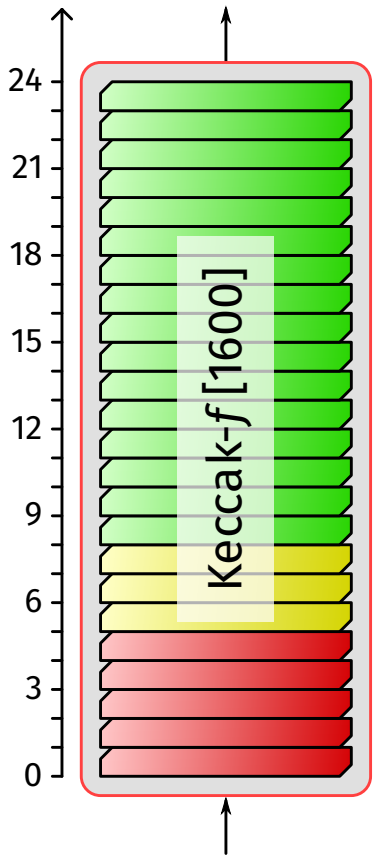
# Second pillar of security in symmetric cryptography

- Security of the primitive
  - No proof!
    $\Rightarrow$ publicly documented design rationale
    $\Rightarrow$ lots of third-party **cryptanalysis!**
- In our case:
  - Ten years of cryptanalysis on (reduced-round) KECCAK-$f$[1600]
    $\Leftarrow$ tune the number of rounds
    $\Leftarrow$ no tweak!

# Second pillar of security in symmetric cryptography

- Security of the primitive
  - No proof!
    $\Rightarrow$ publicly documented design rationale
    $\Rightarrow$ lots of third-party **cryptanalysis!**
- In our case:
  - Ten years of cryptanalysis on (reduced-round) KECCAK-$f$[1600]
    $\Leftarrow$ tune the number of rounds
    $\Leftarrow$ no tweak!

# Status of KECCAK & KANGAROOTWELVE cryptanalysis




- Collision attacks up to 5 rounds
  - Also up to 6 rounds, but for non-standard parameters ($c = 160$)

  [Song, Liao, Guo, CRYPTO 2017]
- Distinguishers
  - 7 rounds (practical time)
    [Huang et al., EUROCRYPT 2017]
  - 8 rounds ($2^{128}$ time, academic)
    [Dinur et al., EUROCRYPT 2015]
- Lots of third-party cryptanalysis available at:
  `https://keccak.team/third_party.html`

# Outline

# Low-end vs high-end

- How to optimize for both low-end and high-end platforms?
- Avoid 32-bit/64-bit mismatches

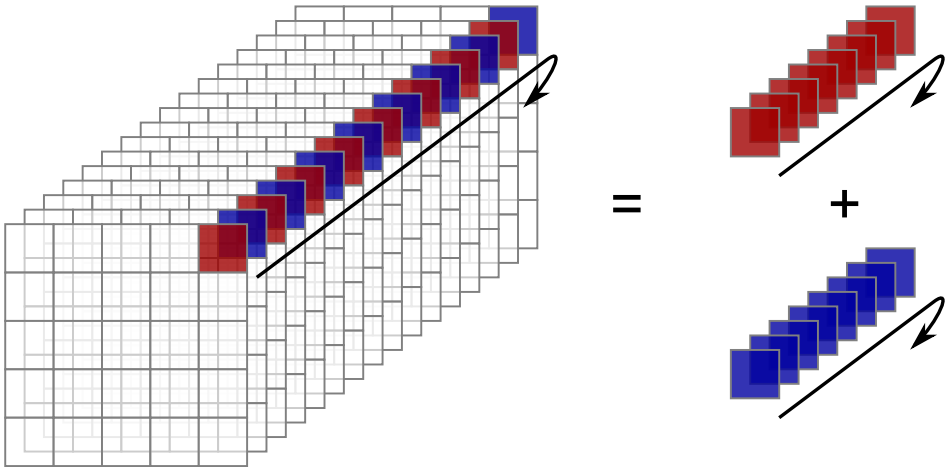
| | 32-bit | 64-bit |
|---|---|---|
| SHA-256 | √ | ± |
| SHA-512 | ± | √ |

# Low-end vs high-end

- How to optimize for both low-end and high-end platforms?
- Avoid 32-bit/64-bit mismatches

| | 32-bit | 64-bit |
|---|---|---|
| SHA-256 | $\surd$ | $\pm$ |
| SHA-512 | $\pm$ | $\surd$ |

# Bit interleaving



| | 32-bit | 64-bit |
|---|---|---|
| KECCAK-$f$[ 800] | $\surd$ | $\pm$ |
| KECCAK-$f$[1600] | $\surd$ | $\surd$ |

$\Rightarrow$ let's stick to KECCAK-$f$[1600]

# Exploit parallelism

At the high end:

- SIMD with growing widths
  - 128, 256 and now 512 bits
- Multiple cores

$\Rightarrow$ let's exploit this parallelism

To remain efficient at the low end:

- One-level tree
- Kangaroo hopping

# Exploit parallelism

At the high end:

- SIMD with growing widths
  - 128, 256 and now 512 bits
- Multiple cores

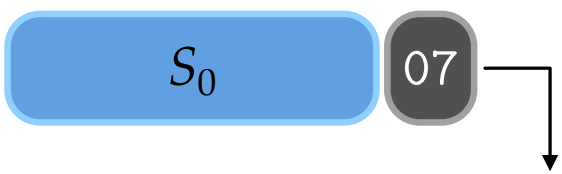
$\Rightarrow$ let's exploit this parallelism

To remain efficient at the low end:

- One-level tree
- Kangaroo hopping

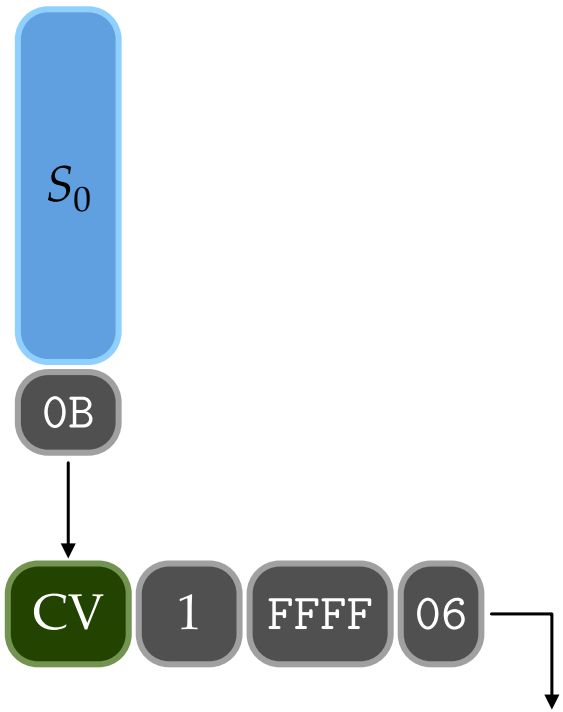
# Short messages ($\leq$ 8KiB)

Kangaroo hopping




one call to $F$

No kangaroo hopping




two calls to $F$

# How fast is KangarooTwelve?

- Twice as fast as SHAKE128 on short inputs $\leq$ 8KiB
- Much faster with parallelism on long inputs $\gg$ 8KiB

| | Short input | Long input |
|---|---|---|
| Intel® Core™ i5-4570 (Haswell) | 3.68 c/b | 1.44 c/b |
| Intel® Core™ i5-6500 (Skylake) | 2.89 c/b | 1.22 c/b |
| Intel® Core™ i7-7800X (SkylakeX) | 2.06 c/b | 0.55 c/b |

Single core only.

# How fast is KangarooTwelve? (AVX2)

| | Skylake (AVX2) cycles/byte |
|---|---|
| KangarooTwelve | 1.22 |
| KangarooTwelve ($\leq$ 8KiB) | 2.89 |
| ParallelHash128 | 2.31 |
| SHAKE128 | 5.56 |
| SHA-256 | 6.91 |
| SHA-512 | 4.64 |
| Blake2bp | 1.34 |
| Blake2sp | 1.29 |
| Blake2b | 3.04 |
| Blake2s | 4.85 |

# How fast is KangarooTwelve? (AVX-512)

Skylake (AVX2) vs SkylakeX (AVX-512)

| | cycles/byte | |
|---|---|---|
| KangarooTwelve | 1.22 | 0.55 |
| KangarooTwelve ($\leq$ 8KiB) | 2.89 | 2.07 |
| ParallelHash128 | 2.31 | 0.96 |
| SHAKE128 | 5.56 | 4.12 |
| SHA-256 | 6.91 | 6.65 |
| SHA-512 | 4.64 | 4.44 |
| Blake2bp | 1.34 | 1.39 |
| Blake2sp | 1.29 | 1.22 |
| Blake2b | 3.04 | 2.98 |
| Blake2s | 4.85 | 4.26 |

Not all optimized for AVX-512 yet

# How fast is KangarooTwelve? (AVX-512)

Skylake (AVX2) vs SkylakeX (AVX-512)

| | cycles/byte | |
|---|---|---|
| KangarooTwelve | **1.22** | **0.55** |
| KangarooTwelve ($\leq$ 8KiB) | 2.89 | 2.07 |
| ParallelHash128 | **2.31** | **0.96** |
| SHAKE128 | 5.56 | 4.12 |
| SHA-256 | 6.91 | 6.65 |
| SHA-512 | 4.64 | 4.44 |
| Blake2bp | 1.34 | 1.39 |
| Blake2sp | 1.29 | 1.22 |
| Blake2b | 3.04 | 2.98 |
| Blake2s | 4.85 | 4.26 |

Not all optimized for AVX-512 yet

# How fast is KangarooTwelve? (AVX-512)

Skylake (AVX2) vs SkylakeX (AVX-512)

| | cycles/byte | |
|---|---|---|
| KangarooTwelve | 1.22 | 0.55 |
| KangarooTwelve ($\leq$ 8KiB) | **2.89** | **2.07** |
| ParallelHash128 | 2.31 | 0.96 |
| SHAKE128 | **5.56** | **4.12** |
| SHA-256 | 6.91 | 6.65 |
| SHA-512 | 4.64 | 4.44 |
| Blake2bp | 1.34 | 1.39 |
| Blake2sp | 1.29 | 1.22 |
| Blake2b | 3.04 | 2.98 |
| Blake2s | 4.85 | 4.26 |

Not all optimized for AVX-512 yet

# Any questions?

## Thanks for your attention!

- More information
  `https://keccak.team/kangarootwelve.html`
- Some implementations
  `https://github.com/gvanas/KeccakCodePackage` (C, Python)
  `https://github.com/kerukuro/digestpp` (C++)
  `https://github.com/mimoo/GoKangarooTwelve` (Go)
  `https://rubygems.org/gems/digest-kangarootwelve` (Ruby)
  `https://github.com/damaki/libkeccak` (Ada)
- Benoît's RFC draft
  `https://datatracker.ietf.org/doc/draft-viguier-kangarootwelve/`