Keccak

Guido Bertoni[1] Joan Daemen[1]
Michaël Peeters[2] Gilles Van Assche[1]

[1]STMicroelectronics

[2]NXP Semiconductors

Eurocrypt 2013
Athens, Greece, May 28th, 2013

Symmetric crypto: what textbooks and intro's say

Symmetric cryptographic primitives:

- Block ciphers
- Stream ciphers
- Hash functions

And their modes-of-use

Picture by GlasgowAmateur

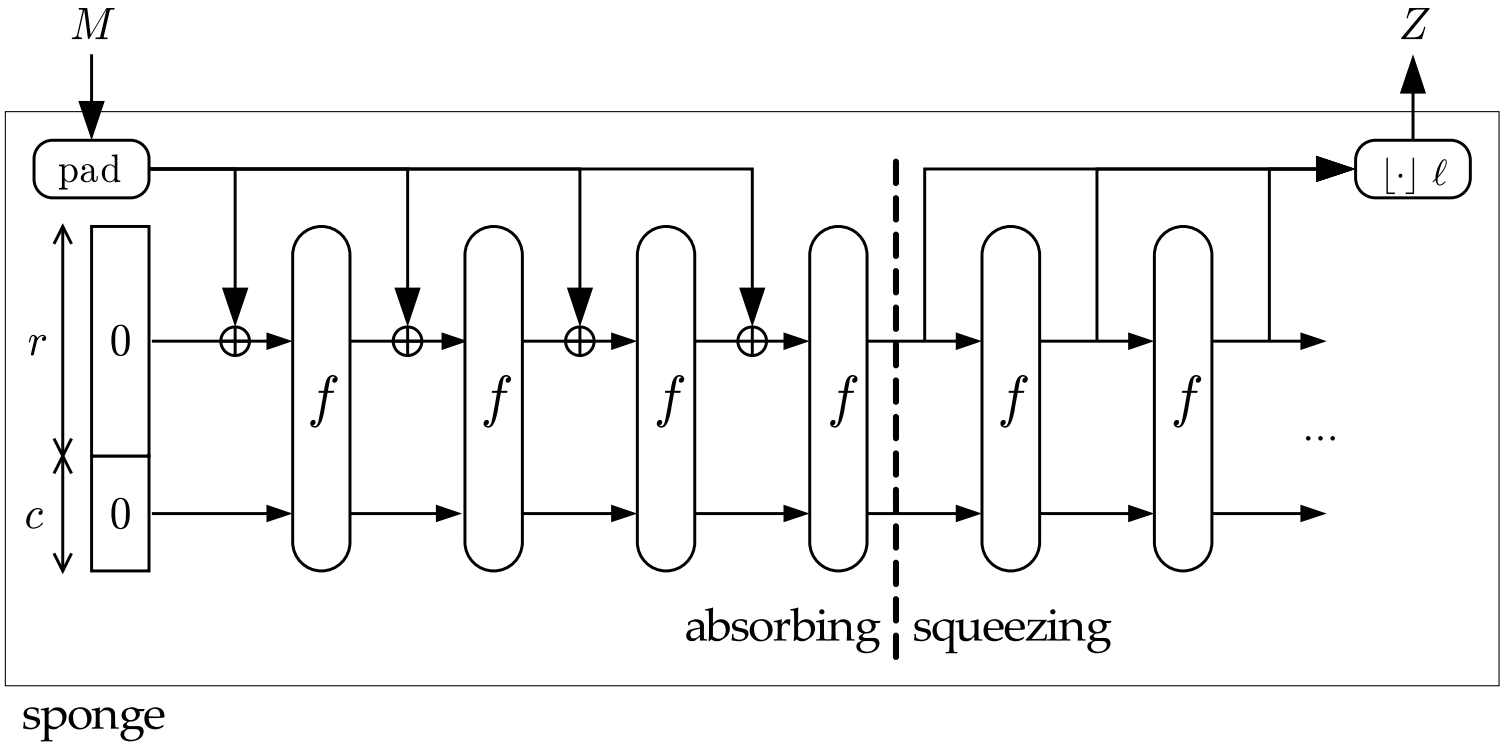
Outline

Outline

Our beginning: RadioGatún

- Initiative to design hash/stream function (late 2005)
  - rumours about NIST call for hash functions
  - forming of Keccak Team
  - starting point: fixing Panama [Daemen, Clapp, FSE 1998]
- RadioGatún [Keccak team, NIST 2nd hash workshop 2006]
  - more conservative than Panama
  - arbitrary output length primitive
  - **expressing security claim** for arbitrary output length primitive
- Sponge functions [Keccak team, Ecrypt hash, 2007]
  - *... closest thing to a random oracle with a finite state ...*
  - Sponge construction calling random permutation

The sponge construction



- More general than a hash function: arbitrary-length output
- Calls a b-bit permutation f, with $b = r + c$
  - r bits of *rate*
  - c bits of *capacity* (security parameter)

Generic security of the sponge construction

Theorem (Indifferentiability of the sponge construction)

The sponge construction calling a random permutation, $\mathcal{S}'[\mathcal{F}]$, is (t_D, t_S, N, ϵ)-indifferentiable from a random oracle, for any t_D, $t_S = O(N^2)$, $N < 2^c$ and for any ϵ with $\epsilon > f_P(N) \approx \frac{N}{2^{c+1}}$.

[Keccak team, Eurocrypt 2008]

Informally, a random sponge is like a random oracle when $N < 2^{c/2}$.

- Collision-, preimage-resistance, etc., up to security strength $c/2$
- The bound assumes f is a random permutation
  - It covers generic attacks
  - ...but not attacks that exploit specific properties of f

Design approach

Hermetic sponge strategy

- Instantiate a sponge function
- Claim a security level of $2^{c/2}$

Our mission

Design permutation f without exploitable properties

How to build a strong permutation

- Like a block cipher
  - Sequence of identical rounds
  - Round consists of sequence of simple step mappings
- ...but not quite
  - No key schedule
  - Round constants instead of round keys
  - Inverse permutation need not be efficient

Outline

Keccak

- Instantiation of a *sponge function*
- Using the permutation Keccak-f
  - 7 permutations: $b \in \{25, 50, 100, 200, 400, 800, 1600\}$
    ... from toy over lightweight to high-speed ...
- SHA-3 instance: $r = 1088$ and $c = 512$
  - permutation width: 1600
  - security strength 256: post-quantum sufficient
- Lightweight instance: $r = 40$ and $c = 160$
  - permutation width: 200
  - security strength 80: same as (initially expected from) SHA-1

See [The Keccak reference] for more details

Keccak

- Instantiation of a *sponge function*
- Using the permutation Keccak-f
  - 7 permutations: $b \in \{25, 50, 100, 200, 400, 800, 1600\}$
    ... from toy over lightweight to high-speed ...
- SHA-3 instance: $r = 1088$ and $c = 512$
  - permutation width: 1600
  - security strength 256: post-quantum sufficient
- Lightweight instance: $r = 40$ and $c = 160$
  - permutation width: 200
  - security strength 80: same as (initially expected from) SHA-1

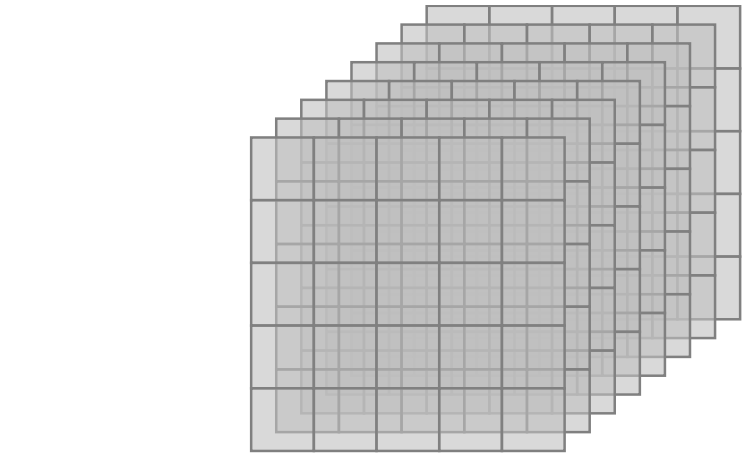
See [The Keccak reference] for more details

Keccak

- Instantiation of a *sponge function*
- Using the permutation Keccak-f
  - 7 permutations: $b \in \{25, 50, 100, 200, 400, 800, 1600\}$
    ... from toy over lightweight to high-speed ...
- SHA-3 instance: $r = 1088$ and $c = 512$
  - permutation width: 1600
  - security strength 256: post-quantum sufficient
- Lightweight instance: $r = 40$ and $c = 160$
  - permutation width: 200
  - security strength 80: same as (initially expected from) SHA-1

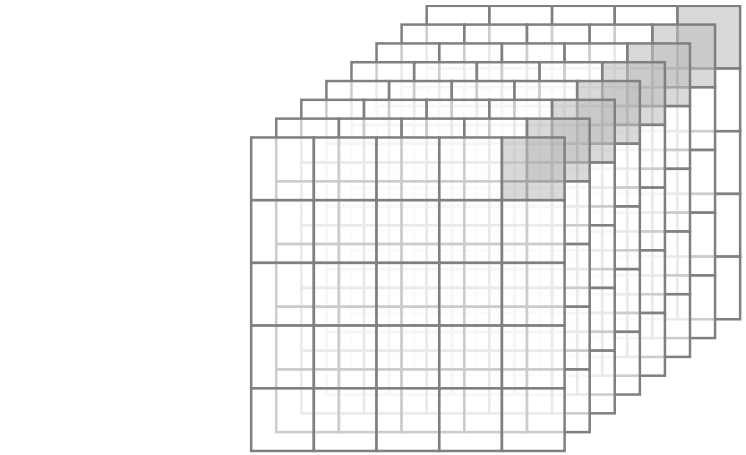
See [The Keccak reference] for more details

The state: an array of $5 \times 5 \times 2^\ell$ bits



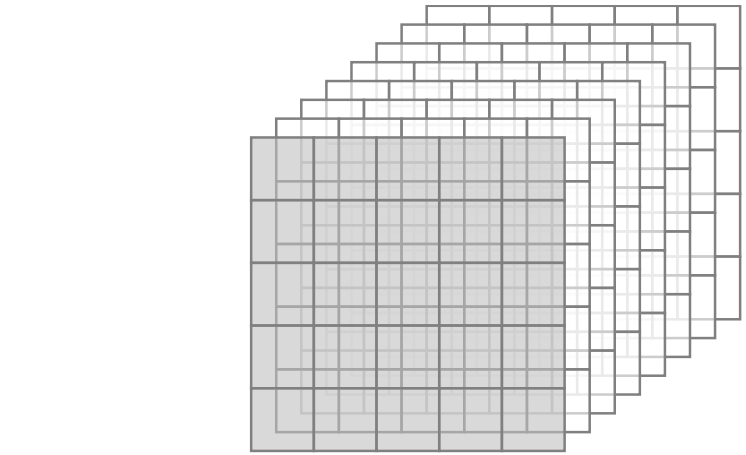
- 5×5 lanes, each containing 2^ℓ bits (1, 2, 4, 8, 16, 32 or 64)
- (5×5)-bit slices, 2^ℓ of them

The state: an array of $5 \times 5 \times 2^\ell$ bits



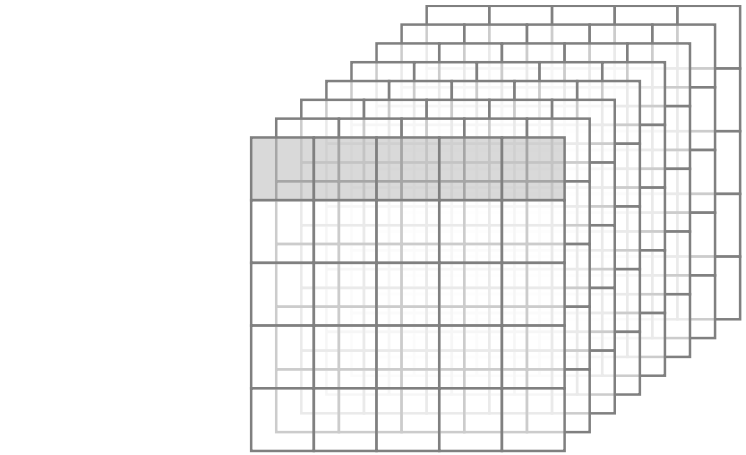
- 5×5 lanes, each containing 2^ℓ bits (1, 2, 4, 8, 16, 32 or 64)
- (5×5)-bit slices, 2^ℓ of them

The state: an array of $5 \times 5 \times 2^\ell$ bits



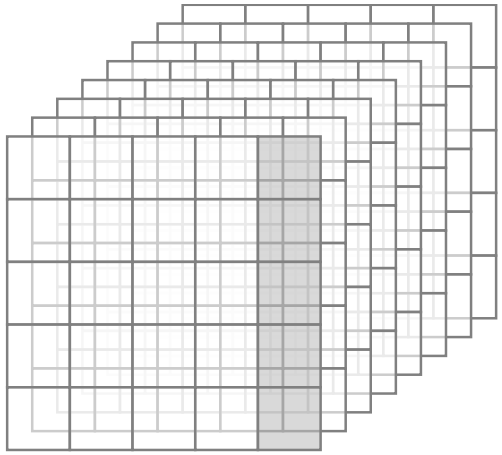
- 5×5 lanes, each containing 2^ℓ bits (1, 2, 4, 8, 16, 32 or 64)
- (5×5)-bit slices, 2^ℓ of them

The state: an array of $5 \times 5 \times 2^\ell$ bits



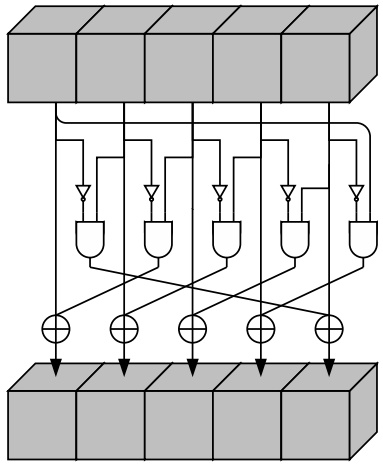
- 5×5 lanes, each containing 2^ℓ bits (1, 2, 4, 8, 16, 32 or 64)
- (5×5)-bit slices, 2^ℓ of them

The state: an array of $5 \times 5 \times 2^\ell$ bits



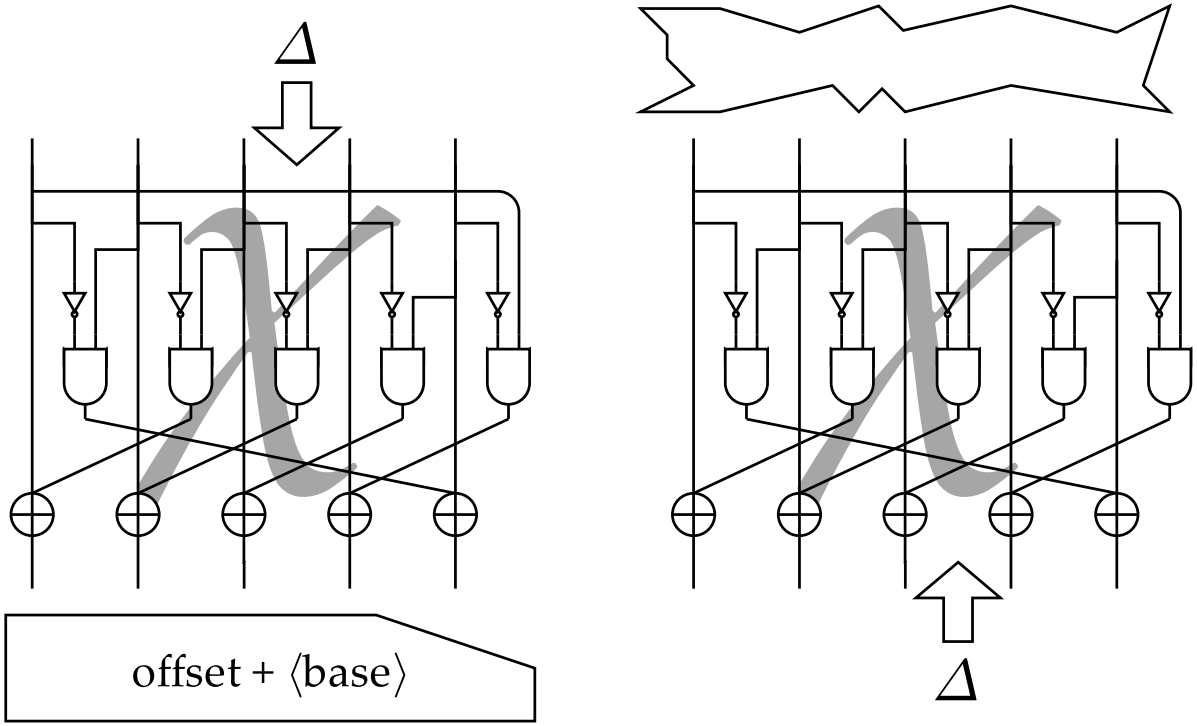
- 5×5 lanes, each containing 2^ℓ bits (1, 2, 4, 8, 16, 32 or 64)
- (5×5)-bit slices, 2^ℓ of them

χ, the nonlinear mapping in Keccak-f



- "Flip bit if neighbors exhibit 01 pattern"
- Operates independently and in parallel on 5-bit rows
- Cheap: small number of operations per bit
- Algebraic degree 2, inverse has degree 3
- LC/DC propagation properties easy to describe and analyze

Propagating differences through χ



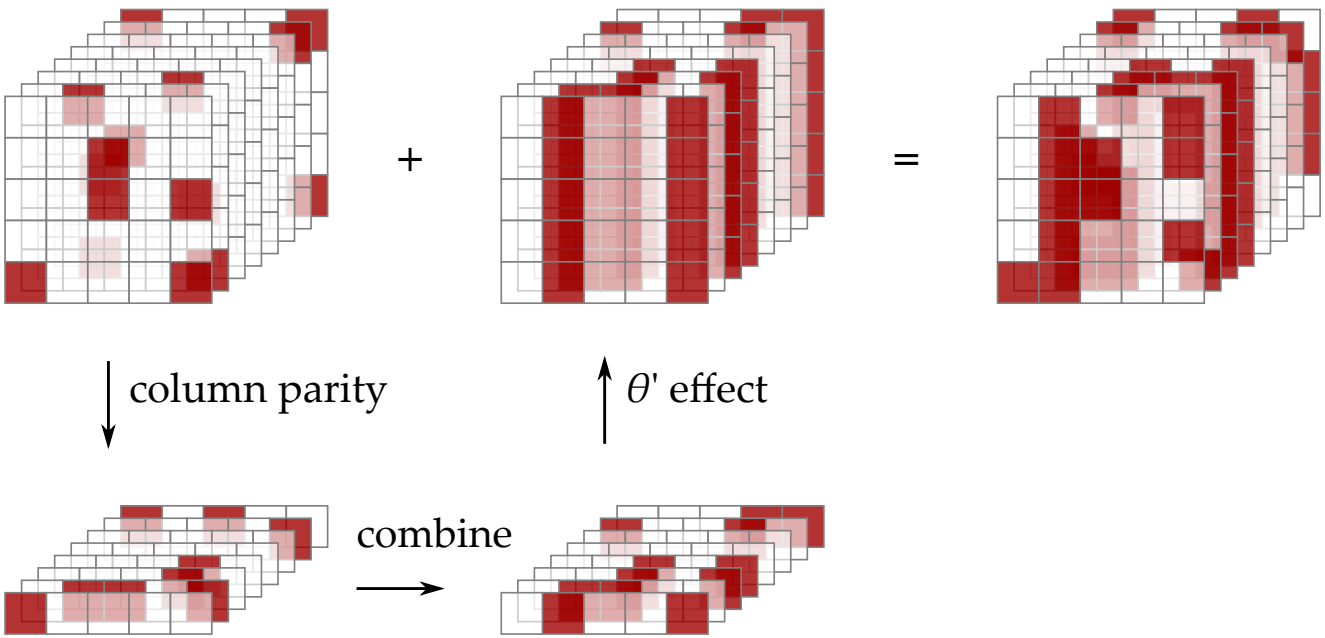
- The propagation weight...
  - ... is equal to $-\log_2(\text{fraction of pairs})$;
  - ... is determined by input difference only;
  - ... is the size of the affine base;
  - ... is the number of affine conditions.

θ', a first attempt at mixing bits

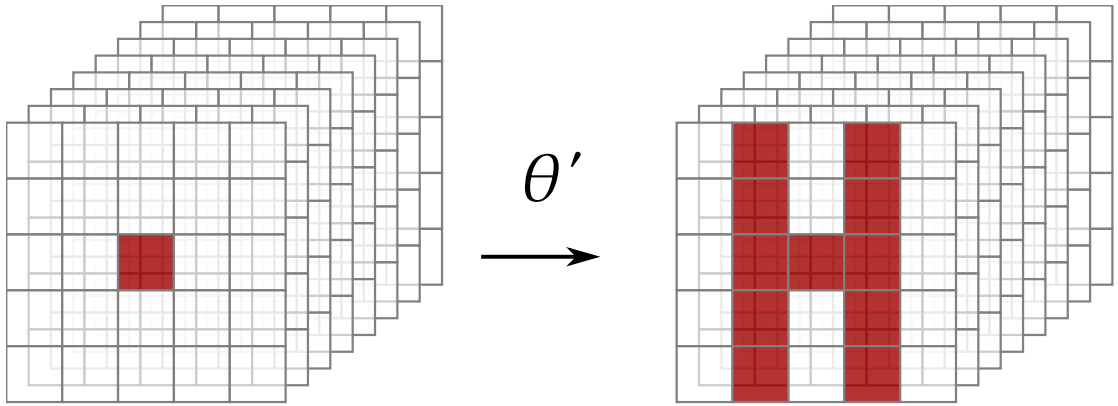
- Compute parity $c_{x,z}$ of each column
- Add to each cell parity of neighboring columns:

$$b_{x,y,z} = a_{x,y,z} \oplus c_{x-1,z} \oplus c_{x+1,z}$$

- Cheap: two XORs per bit

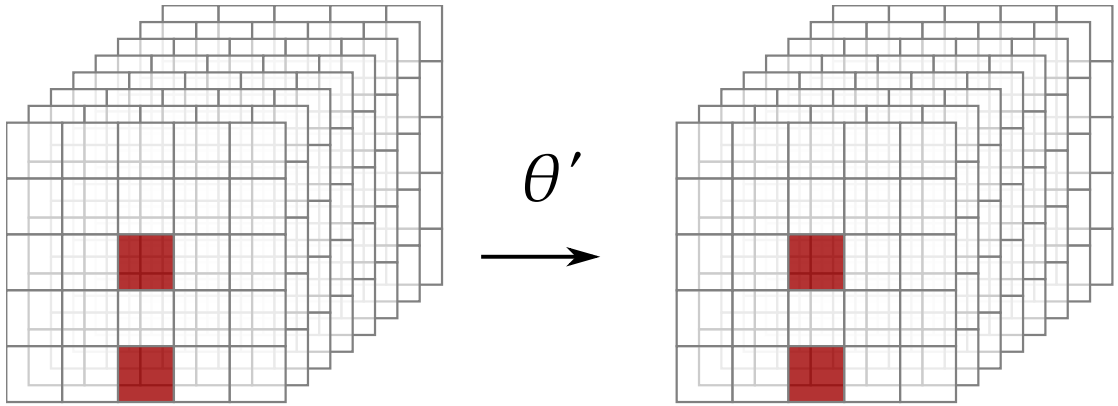


Diffusion of θ'



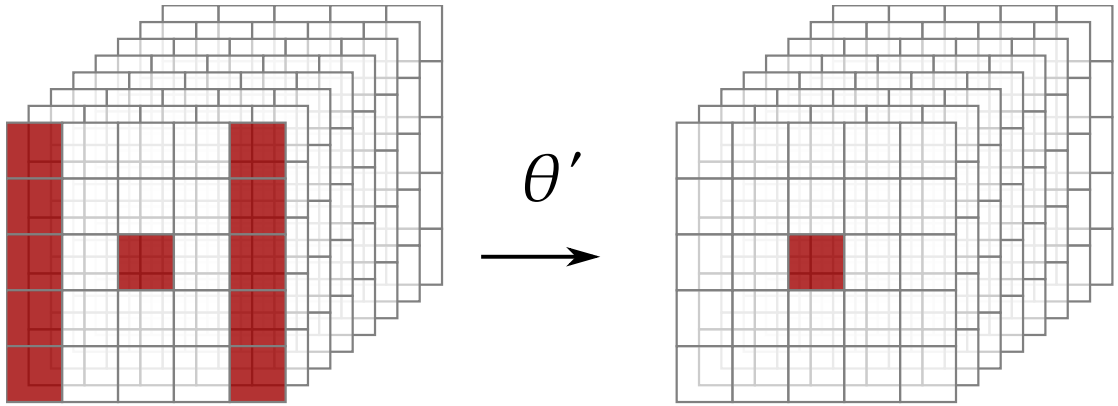
$$1 + \left(1 + y + y^2 + y^3 + y^4\right)\left(x + x^4\right) \left(\mathrm{mod}\ \left\langle 1 + x^5, 1 + y^5, 1 + z^w \right\rangle\right)$$

Diffusion of θ' (kernel)



$$1 + \left(1 + y + y^2 + y^3 + y^4\right)\left(x + x^4\right)$$
$$\left(\mathrm{mod}\ \left\langle 1 + x^5, 1 + y^5, 1 + z^w \right\rangle\right)$$

Diffusion of the inverse of θ'



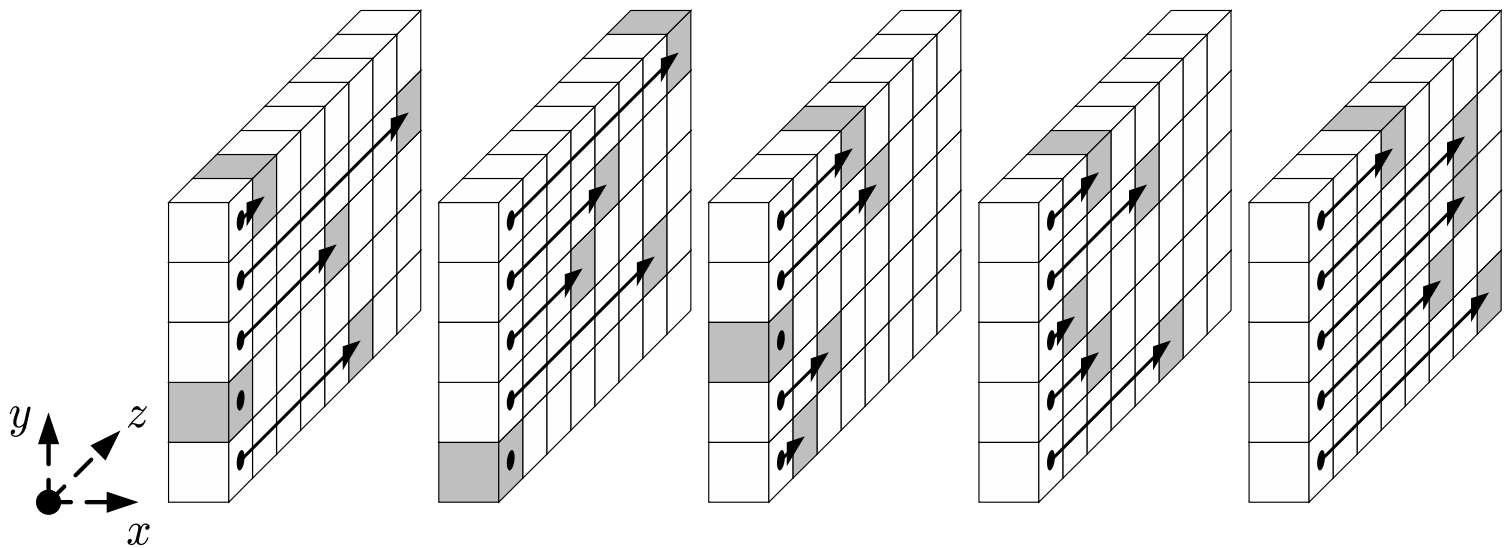
$$1 + \left(1 + y + y^2 + y^3 + y^4\right)\left(x^2 + x^3\right) \left(\mathrm{mod}\ \left\langle 1 + x^5, 1 + y^5, 1 + z^w \right\rangle\right)$$

ρ for inter-slice dispersion

- We need diffusion between the slices ...
- ρ: cyclic shifts of lanes with offsets

$$i(i+1)/2 \bmod 2^\ell, \text{ with } \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^{i-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

- Offsets cycle through all values below 2^ℓ

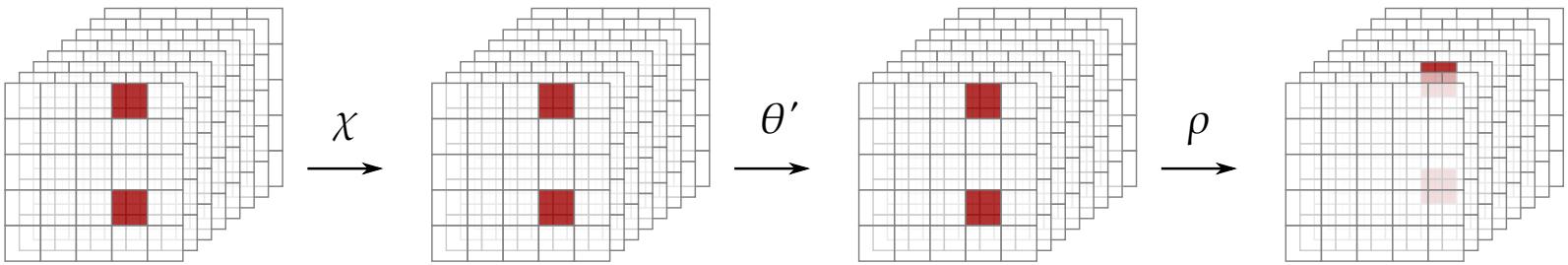


ι to break symmetry

- XOR of round-dependent constant to lane in origin
- Without ι, the round mapping would be symmetric
  - invariant to translation in the z-direction
  - susceptible to *rotational* cryptanalysis
- Without ι, all rounds would be the same
  - susceptibility to *slide* attacks
  - defective cycle structure
- Without ι, we get simple fixed points (000 and 111)

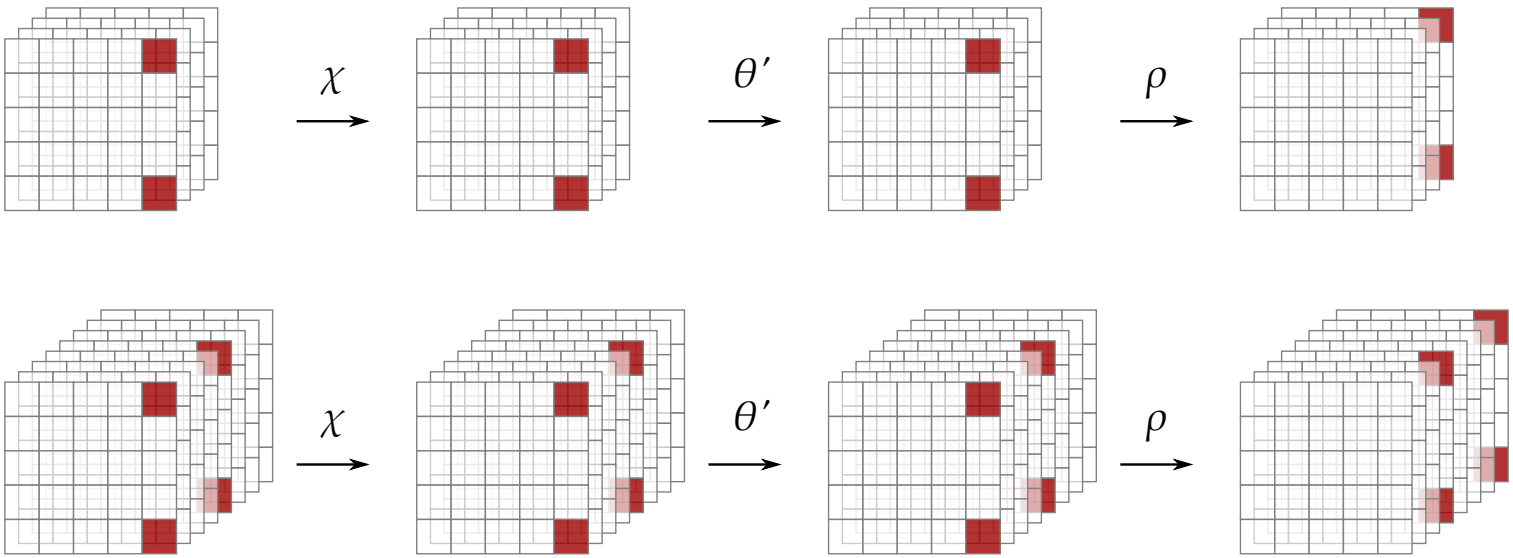
A first attempt at Keccak-f

- Round function: $\mathsf{R} = \iota \circ \rho \circ \theta' \circ \chi$
- Problem: low-weight periodic trails by chaining:



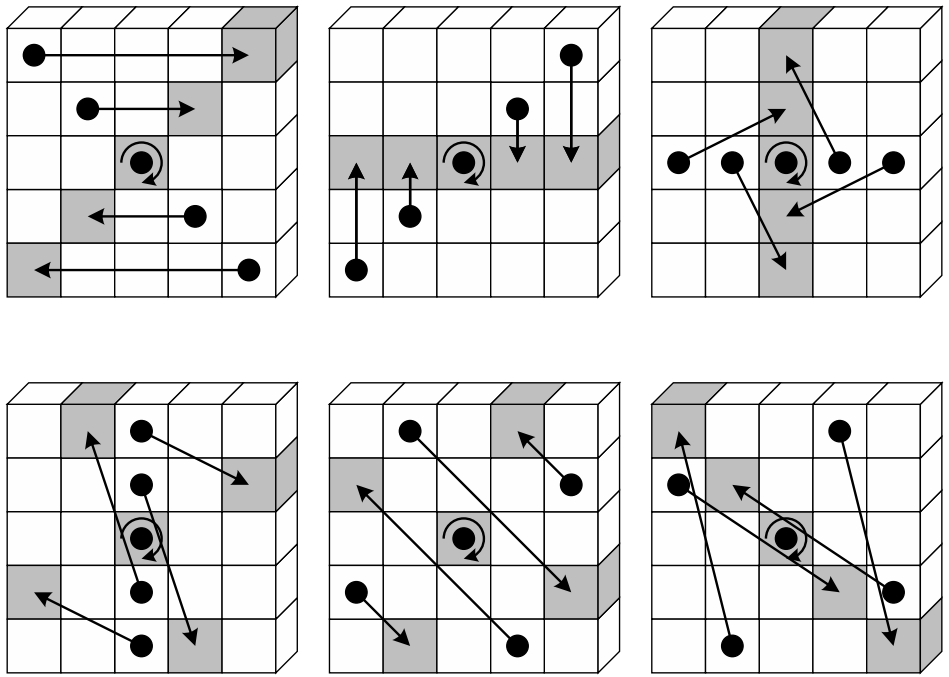
  - χ: propagates unchanged with weight 4
  - θ': propagates unchanged, because all column parities are 0
  - ρ: in general moves active bits to different slices …
    …but not always

The Matryoshka property



- Patterns in Q' are z-periodic versions of patterns in Q
- Weight of trail Q' is twice that of trail Q (or 2^n times in general)

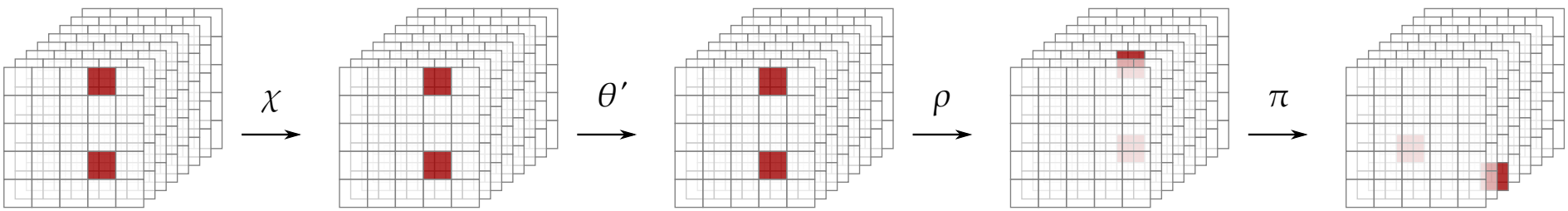
π for disturbing horizontal/vertical alignment



$$a_{x,y} \leftarrow a_{x',y'} \text{ with } \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}$$

A second attempt at Keccak-f

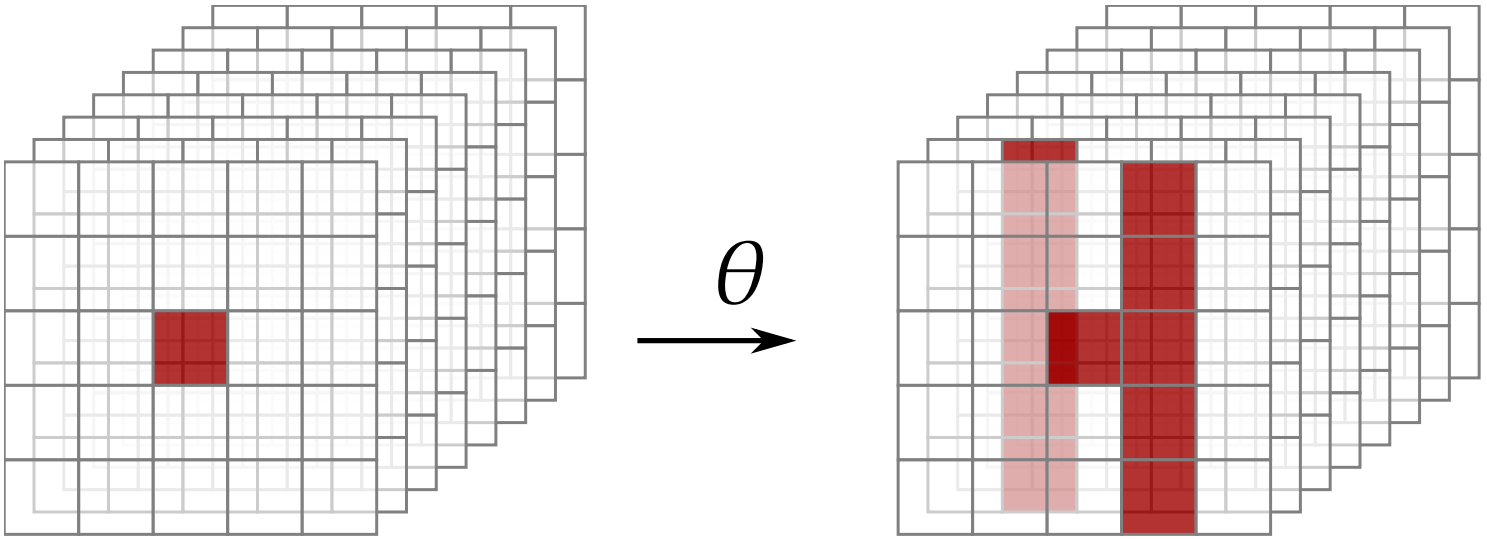
- Round function: $\mathsf{R} = \iota \circ \pi \circ \rho \circ \theta' \circ \chi$
- Solves problem encountered before:



- π moves bits in same column to different columns!

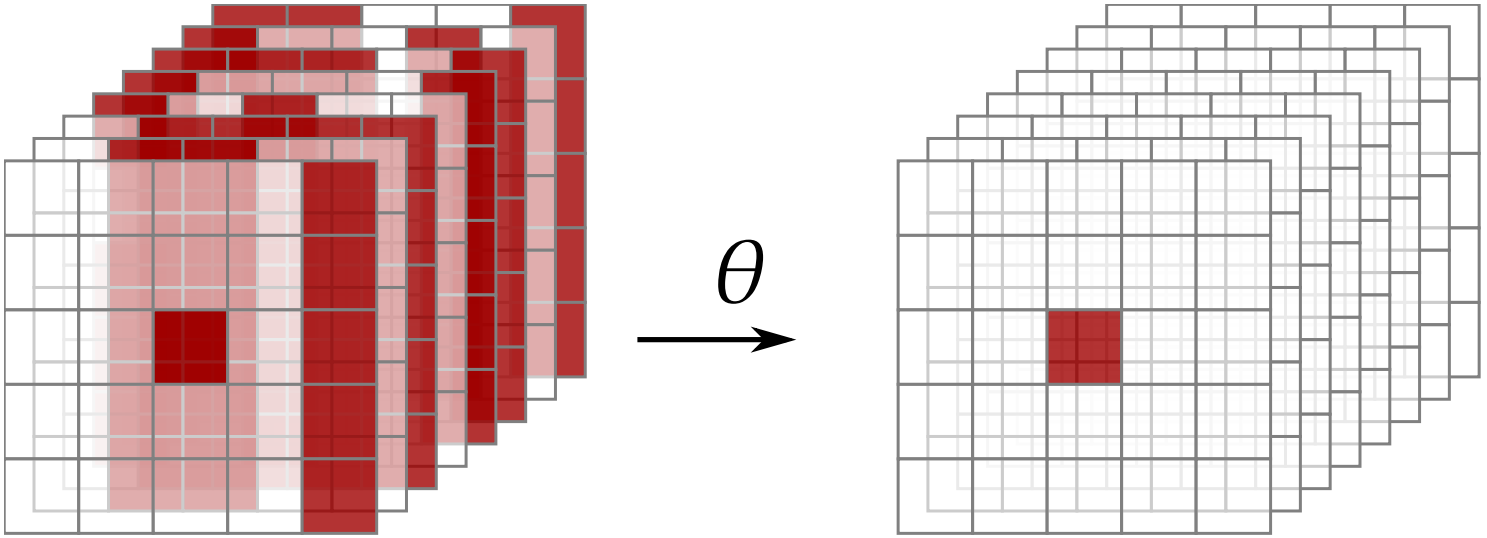
Almost there, still a final tweak ...

Tweaking θ' to θ



$$1 + \left(1 + y + y^2 + y^3 + y^4\right)\left(x + x^4 z\right) \left(\mathrm{mod}\ \left\langle 1 + x^5, 1 + y^5, 1 + z^w \right\rangle\right)$$

Inverse of θ



$$1 + \left(1 + y + y^2 + y^3 + y^4\right)\mathbf{Q},$$

$$\text{with } \mathbf{Q} = 1 + \left(1 + x + x^4 z\right)^{-1} \bmod \left\langle 1 + x^5, 1 + z^w \right\rangle$$

- **Q** is dense, so:
  - Diffusion from single-bit output to input very high
  - Increases resistance against LC/DC and algebraic attacks

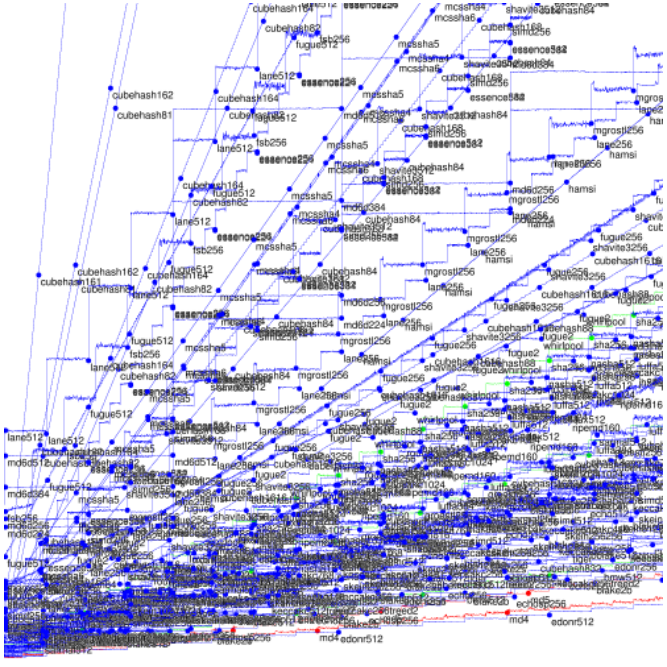
Keccak-f summary

- Round function:

$$\mathsf{R} = \iota \circ \chi \circ \pi \circ \rho \circ \theta$$

- Number of rounds: $12 + 2\ell$
  - Keccak-$f[25]$ has 12 rounds
  - Keccak-$f[1600]$ has 24 rounds
- Efficiency [Keccak implementation overview]
  - high level of parallellism
  - flexibility: bit-interleaving
  - software: fast on wide range of CPU
  - dedicated hardware: very fast
  - suited for protection against side-channel attack
    [Debande, Le and Keccak team, HASP 2012 + ePrint 2013/067]

Performance in software



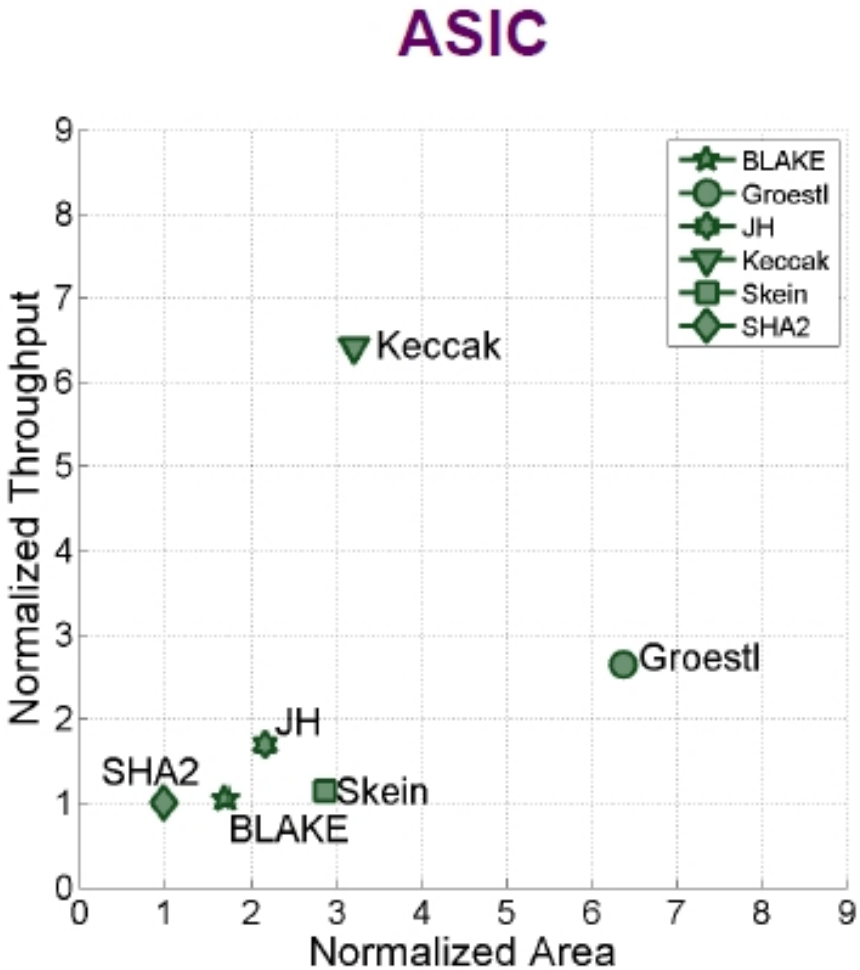
- Faster than SHA-2 on all modern PCs
- KeccakTree faster than MD5 on some platforms

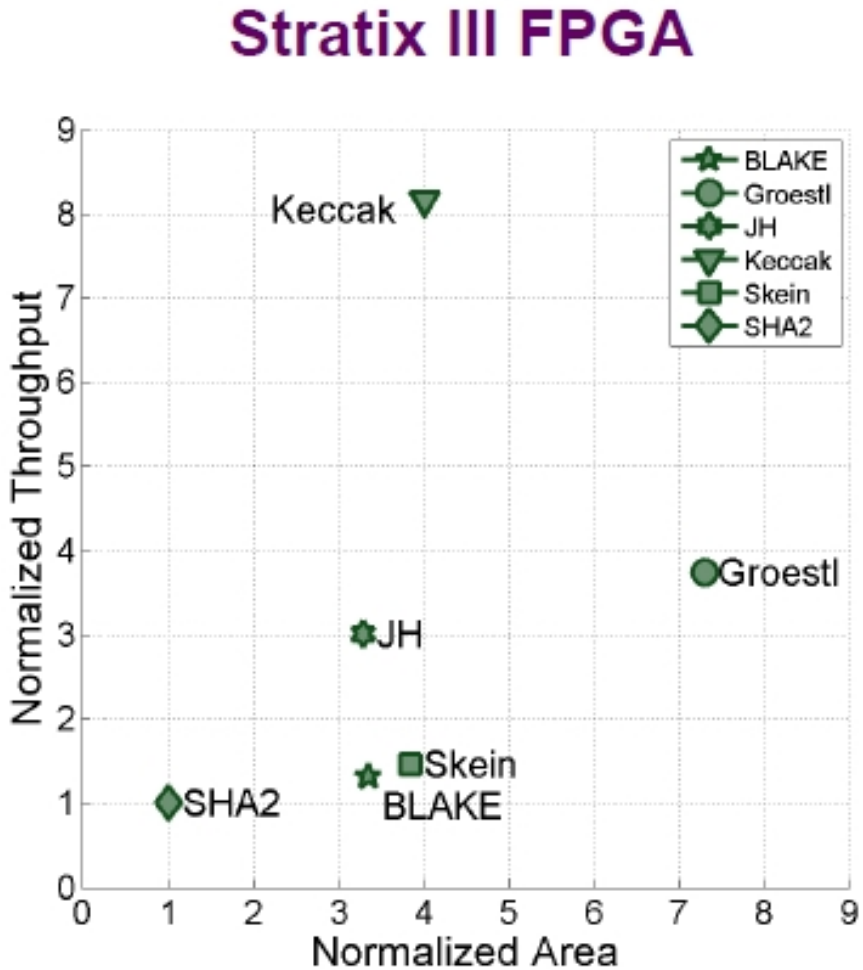
C/b	Algo	Strength
4.79	keccakc256treed2	128
4.98	md5 **broken!**	64
5.89	keccakc512treed2	256
6.09	sha1 **broken!**	80
8.25	keccakc256	128
10.02	keccakc512	256
13.73	sha512	256
21.66	sha256	128

[eBASH, hydra6 (AMD Bulldozer), http://bench.cr.yp.to/]

Efficient and flexible in hardware

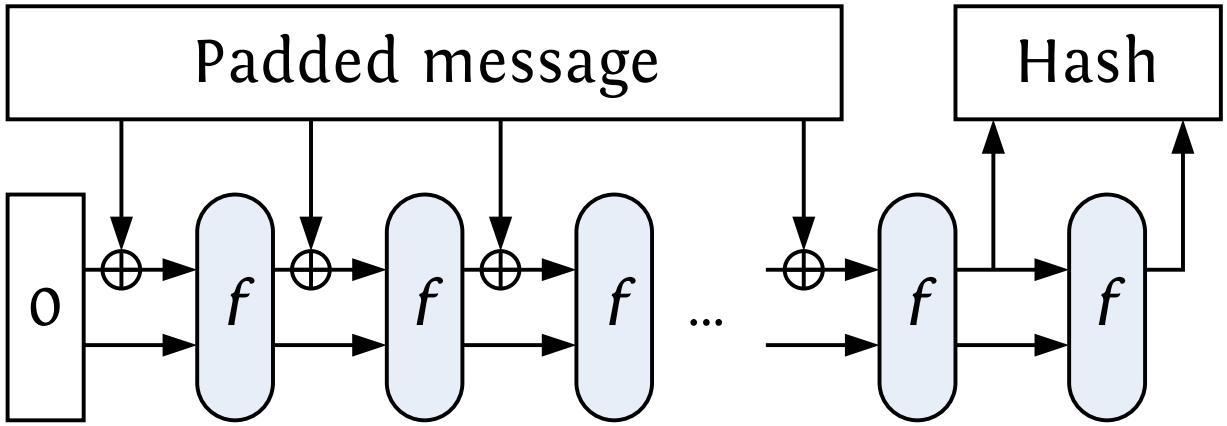
From Kris Gaj's presentation at SHA-3, Washington 2012:





Outline

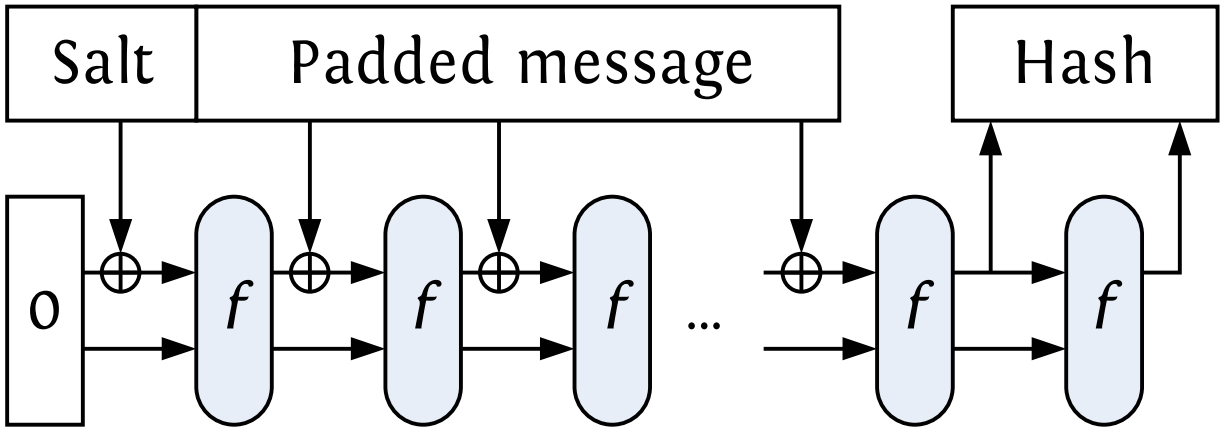
Regular hashing



- Electronic signatures
- Data integrity (*shaXsum* ...)
- Data identifier (*Git, online anti-virus, peer-2-peer* ...)

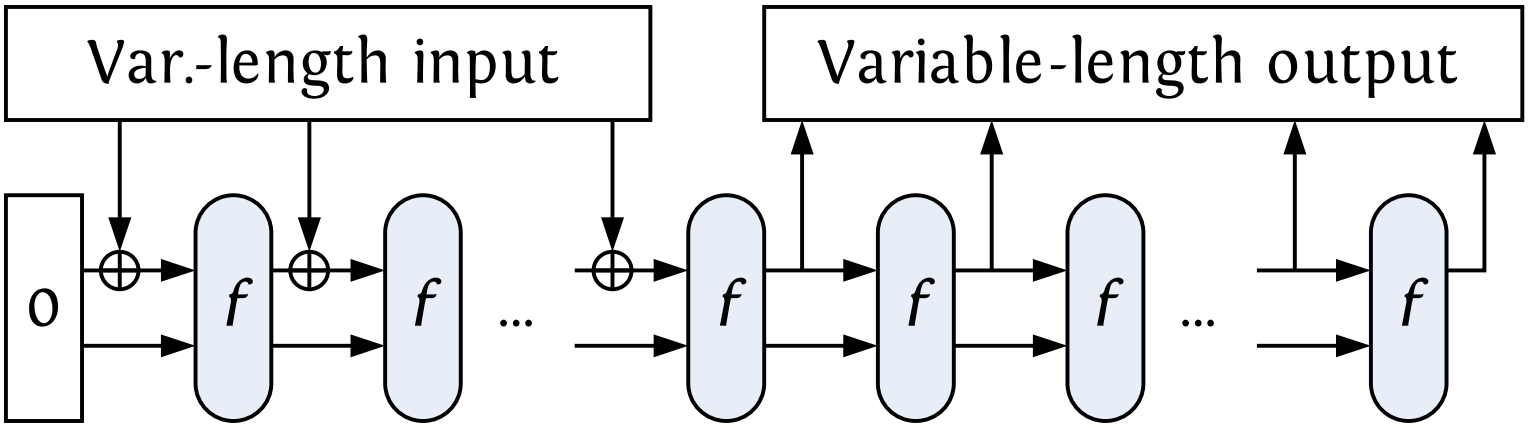
See [Cryptographic sponge functions] for more details

Salted hashing



- Randomized hashing (RSASSA-PSS)
- Password storage and verification (*Kerberos*, /etc/shadow)

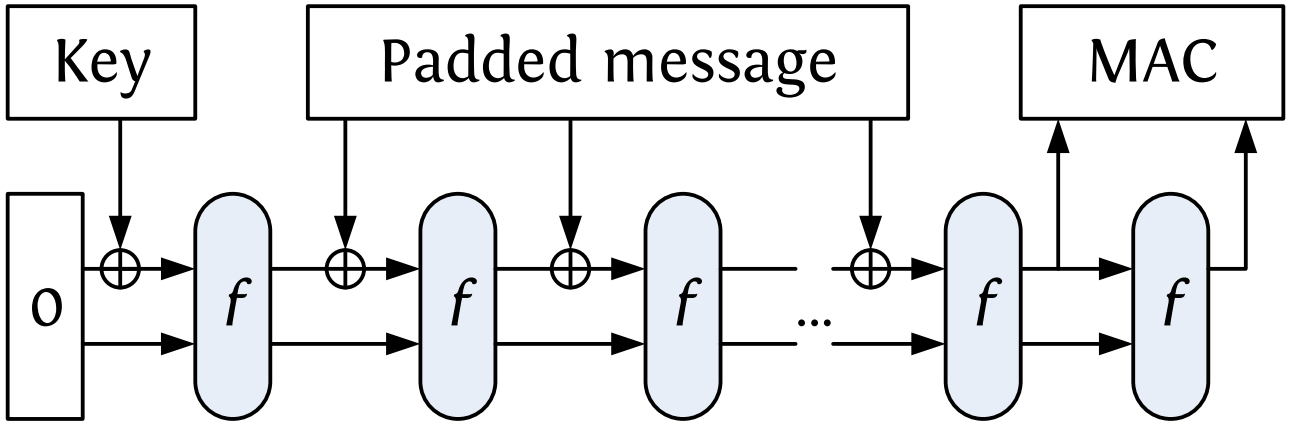
Mask generation function



output length often dictated by application ...
... rather than by security strength level

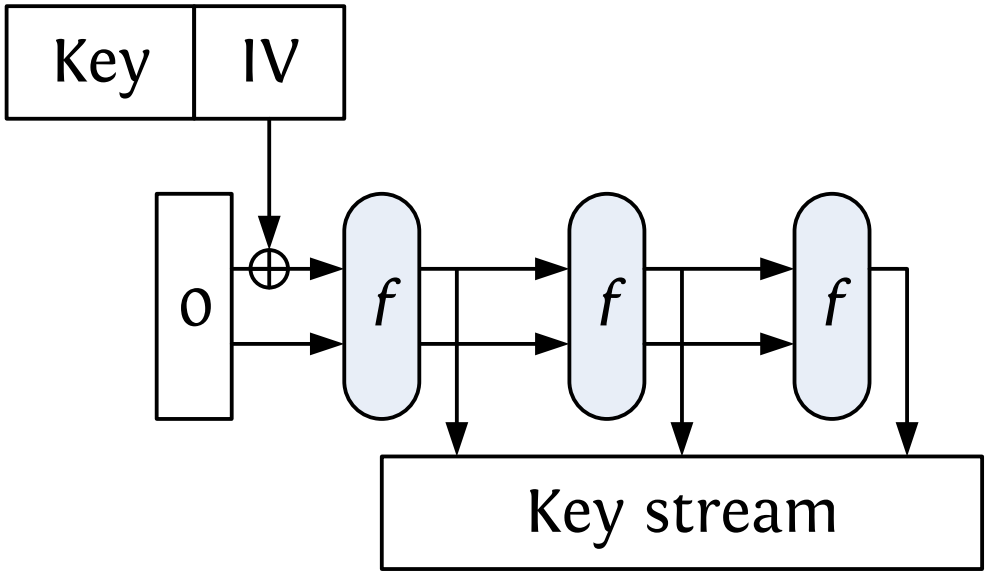
- Key derivation function in SSL, TLS
- Full-domain hashing in public key cryptography
  - electronic signatures RSASSA-PSS [PKCS#1]
  - encryption RSAES-OAEP [PKCS#1]
  - key encapsulation methods (KEM)

Message authentication codes



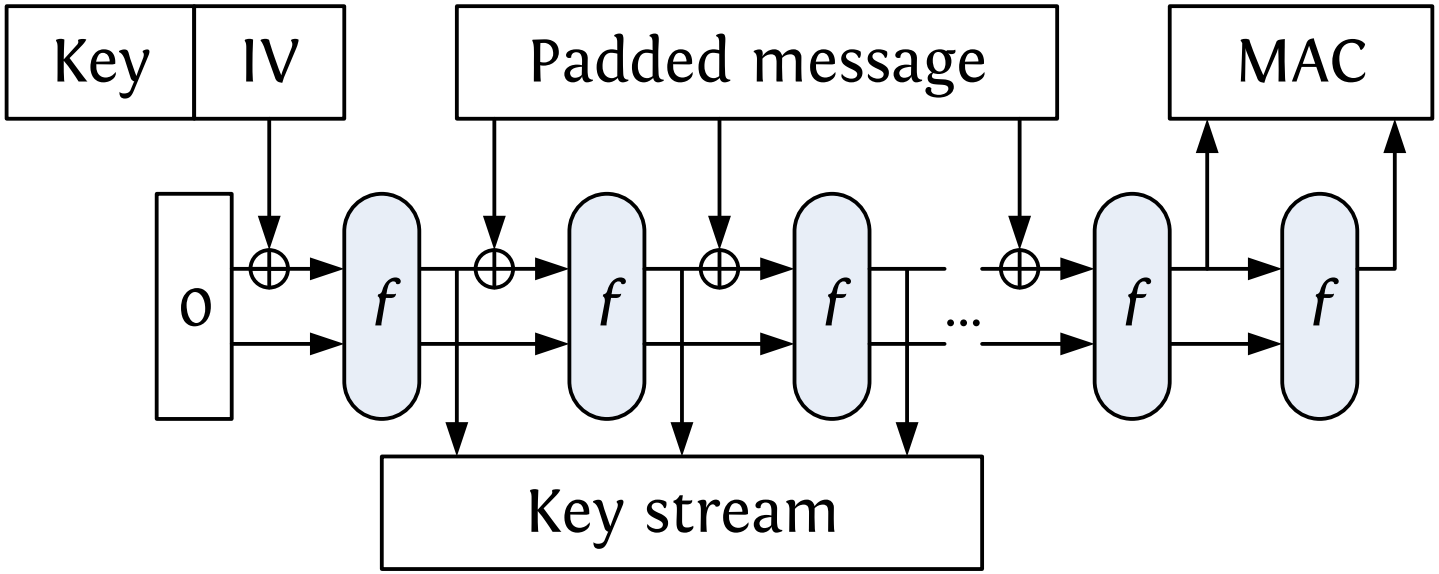
- As a message authentication code
- Simpler than HMAC [FIPS 198]
  - Required for SHA-1, SHA-2 due to length extension property
  - HMAC is **no longer needed** for sponge!

Stream encryption



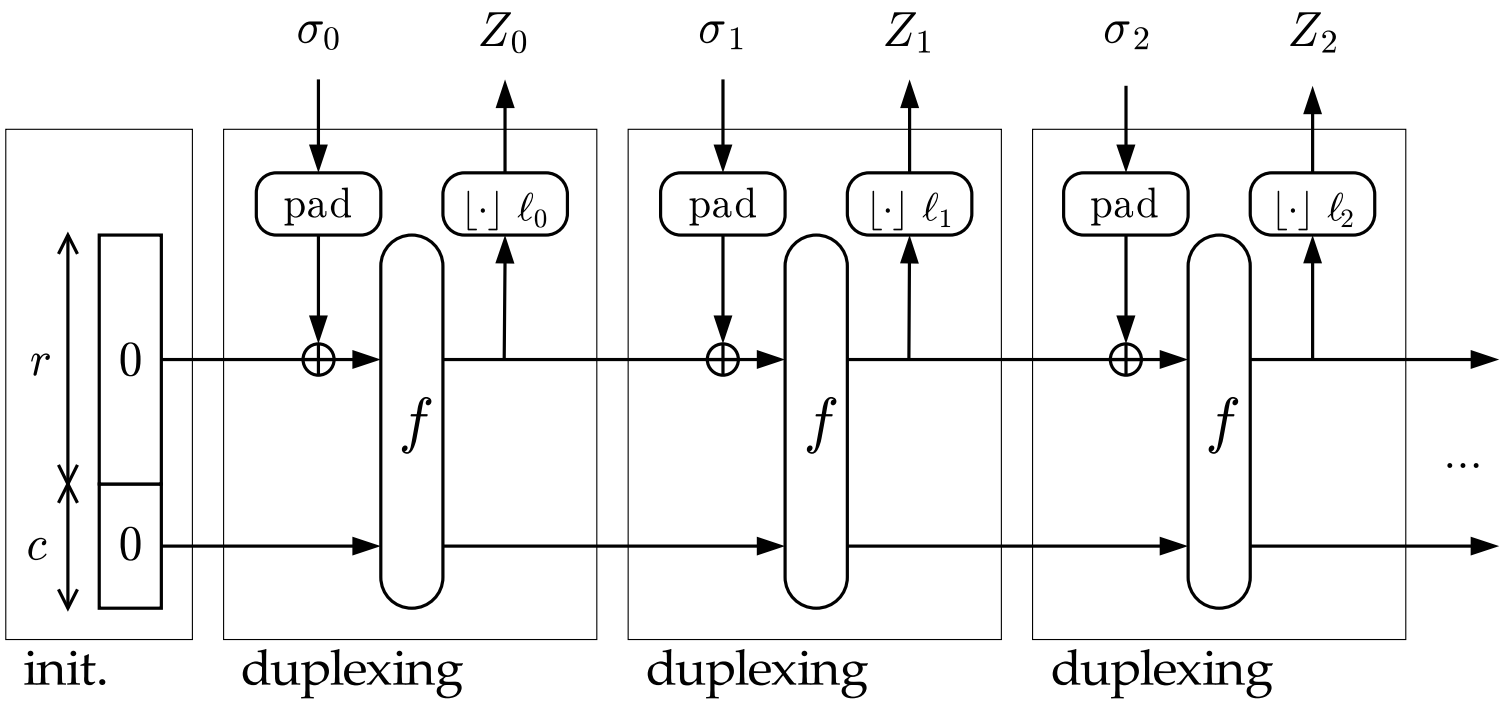
- As a stream cipher
  - Long output stream per IV: similar to OFB mode
  - Short output stream per IV: similar to counter mode

Single pass authenticated encryption



- Authentication and encryption in a single pass!
- Secure messaging (*SSL/TLS, SSH, IPSEC* ...)

The duplex construction



- Generic security equivalent to Sponge [Keccak team, SAC 2011]
- Applications include:
  - Authenticated encryption: spongeWrap
  - Reseedable pseudorandom sequence generator

Outline

Output length oriented approach

Output length	Collision resistance	Pre-image resistance	Required capacity	Relative perf.	SHA-3 instance
$n = 224$	$s \leq 112$	$s \leq 224$	$c = 448$	$\times 1.125$	**SHA3n224**
$n = 256$	$s \leq 128$	$s \leq 256$	$c = 512$	$\times 1.063$	**SHA3n256**
$n = 384$	$s \leq 192$	$s \leq 384$	$c = 768$	$\div 1.231$	**SHA3n384**
$n = 512$	$s \leq 256$	$s \leq 512$	$c = 1024$	$\div 1.778$	**SHA3n512**
n	$s \leq n/2$	$s \leq n$	$c = 2n$	$\times \frac{1600-c}{1024}$	

s: security strength level [NIST SP 800-57]

- These instances address the SHA-3 requirements, but:
  - multiple security strengths each
  - levels outside of [NIST SP 800-57] range
- Performance penalty!

Security strength oriented approach

Security strength	Collision resistance	Pre-image resistance	Required capacity	Relative perf.	SHA-3 instance
$s = 112$	$n \geq 224$	$n \geq 112$	$c = 224$	$\times 1.343$	**SHA3c224**
$s = 128$	$n \geq 256$	$n \geq 128$	$c = 256$	$\times 1.312$	**SHA3c256**
$s = 192$	$n \geq 384$	$n \geq 192$	$c = 384$	$\times 1.188$	**SHA3c384**
$s = 256$	$n \geq 512$	$n \geq 256$	$c = 512$	$\times 1.063$	**SHA3c512**
s	$n \geq 2s$	$n \geq s$	$c = 2s$	$\times \frac{1600-c}{1024}$	**SHA3[c=2s]**

s: security strength level [NIST SP 800-57]

- These SHA-3 instances
  - are consistent with philosophy of [NIST SP 800-57]
  - provide a one-to-one mapping to security strength levels
- Higher efficiency

NIST SHA-3 standardization plans

- A new FIPS number (not 180-*n*)
- Two capacities: 256 and 512
- 6 instances with domain separation between them
- Tree-hashing ready: SAKURA coding

Sponge instances	SHA-2 drop-in replacements
$\text{KECCAK}[c = 256](M\|\|11\|\|11)$	$\lfloor \text{KECCAK}[c = 256](M\|\|11\|\|001) \rfloor_{224}$ $\lfloor \text{KECCAK}[c = 256](M\|\|11\|\|101) \rfloor_{256}$
$\text{KECCAK}[c = 512](M\|\|11\|\|11)$	$\lfloor \text{KECCAK}[c = 512](M\|\|11\|\|001) \rfloor_{384}$ $\lfloor \text{KECCAK}[c = 512](M\|\|11\|\|101) \rfloor_{512}$

Sakura and tree hashing

- Sound tree hashing is relatively easy to achieve
  - Sufficient conditions for indifferentiability from RO
    [Keccak team, ePrint 2009/210 — **updated April 2013**]
- Defining tree hash modes addressing all future use cases is hard
  - A chosen number of leaves for a chosen amount of parallelism?
  - Or a binary tree with the option of saving intermediate hash results?
- Defining future-proof tree hash coding is easy

Sakura, a flexible coding for tree hashing

- Automatically satisfying the sufficient conditions of [ePrint 2009/210]
- For any underlying hash function (not just Keccak)
- For any tree topology

  $\Rightarrow$ no conflicts adding future tree structures

See [Keccak team, ePrint 2013/231] for more details

SAKURA and tree hashing

- Sound tree hashing is relatively easy to achieve
  - Sufficient conditions for indifferentiability from RO
    [Keccak team, ePrint 2009/210 – **updated April 2013**]
- Defining tree hash modes addressing all future use cases is hard
  - A chosen number of leaves for a chosen amount of parallelism?
  - Or a binary tree with the option of saving intermediate hash results?
- Defining future-proof tree hash coding is easy

SAKURA, a flexible coding for tree hashing

- Automatically satisfying the sufficient conditions of [ePrint 2009/210]
- For any underlying hash function (not just KECCAK)
- For any tree topology

  $\Rightarrow$ no conflicts adding future tree structures

See [Keccak team, ePrint 2013/231] for more details

Sakura and tree hashing

- Sound tree hashing is relatively easy to achieve
  - Sufficient conditions for indifferentiability from RO
    [Keccak team, ePrint 2009/210 – **updated April 2013**]
- Defining tree hash modes addressing all future use cases is hard
  - A chosen number of leaves for a chosen amount of parallelism?
  - Or a binary tree with the option of saving intermediate hash results?
- Defining future-proof tree hash coding is easy

Sakura, a flexible coding for tree hashing

- Automatically satisfying the sufficient conditions of [ePrint 2009/210]
- For any underlying hash function (not just Keccak)
- For any tree topology

  $\Rightarrow$ no conflicts adding future tree structures

See [Keccak team, ePrint 2013/231] for more details

Outline

Design decisions behind Keccak-f

- Ability to control propagation of differences or linear masks
  - Differential/linear trail analysis
  - Lower bounds for trail weights
  - Alignment and trail clustering
  - $\Rightarrow$ This shaped θ, π and ρ
- Algebraic properties
  - Distribution of # terms of certain degrees
  - Ability of solving certain problems (CICO) algebraically
  - Zero-sum distinguishers (third party)
  - $\Rightarrow$ This determined the number of rounds
- Analysis of symmetry properties
  $\Rightarrow$ This shaped ι

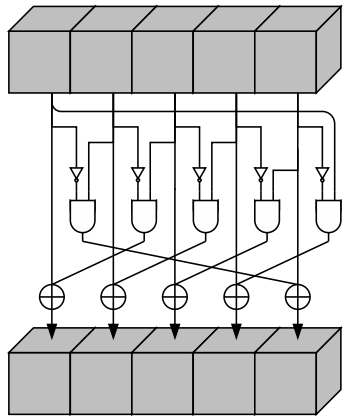
Design decisions behind KECCAK-f

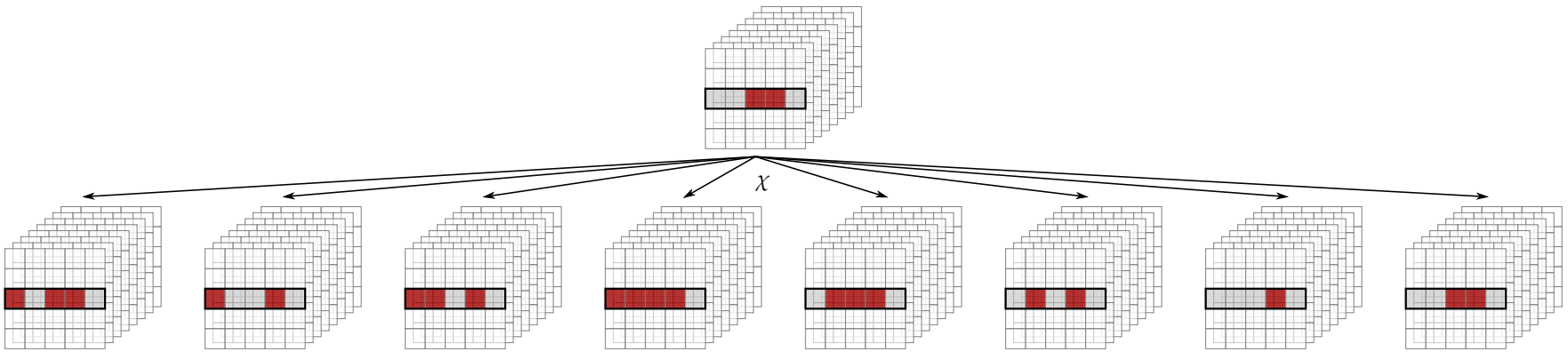
- Ability to control propagation of differences or linear masks
  - **Differential/linear trail analysis**
  - **Lower bounds for trail weights**
  - **Alignment and trail clustering**
  - $\Rightarrow$ This shaped θ, π and ρ
- Algebraic properties
  - Distribution of # terms of certain degrees
  - Ability of solving certain problems (CICO) algebraically
  - Zero-sum distinguishers (third party)
  - $\Rightarrow$ This determined the number of rounds
- Analysis of symmetry properties

  $\Rightarrow$ This shaped ι

Non-linear mapping χ

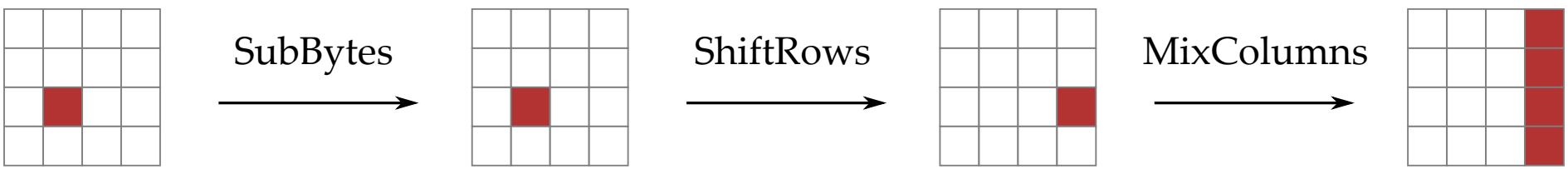
- Transforms each **row** independently
- E.g., a difference going through χ
  - Output: affine space





Difference propagation in RIJNDAEL: strong alignment

- Propagation of differentials:
  - One-to-one through MixColumns, ShiftRows and AddRoundKey
  - One-to-multiple through SubBytes
- Propagation of truncated differentials (active/passive bytes)
  - One-to-one through SubBytes, ShiftRows and AddRoundKey
  - One-to-multiple through MixColumns
    - Sometimes one-to-one: 1 byte $\rightarrow$ 4 bytes

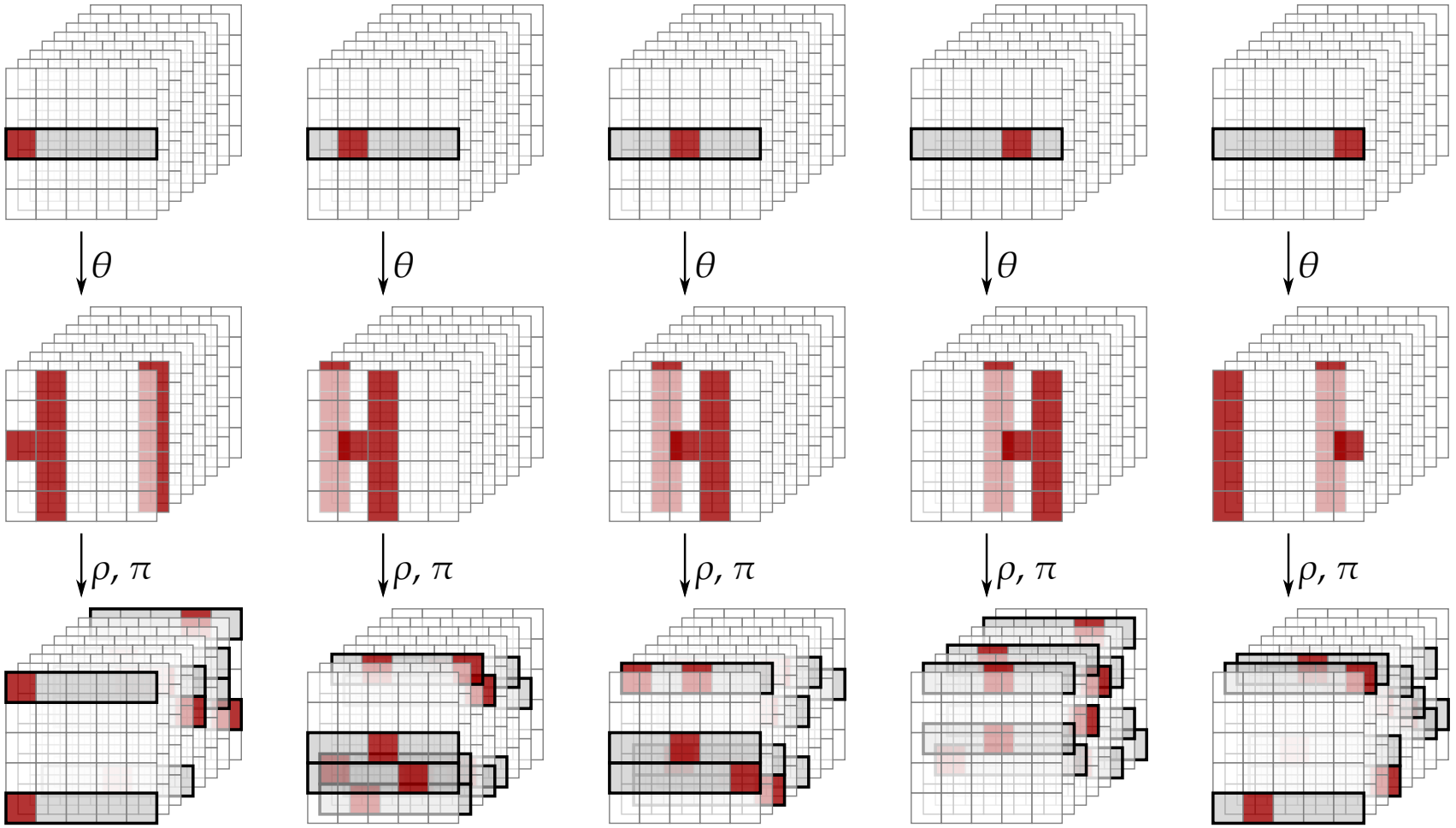


See also [Daemen and Rijmen, Understanding two-round AES differentials, SCN '06]

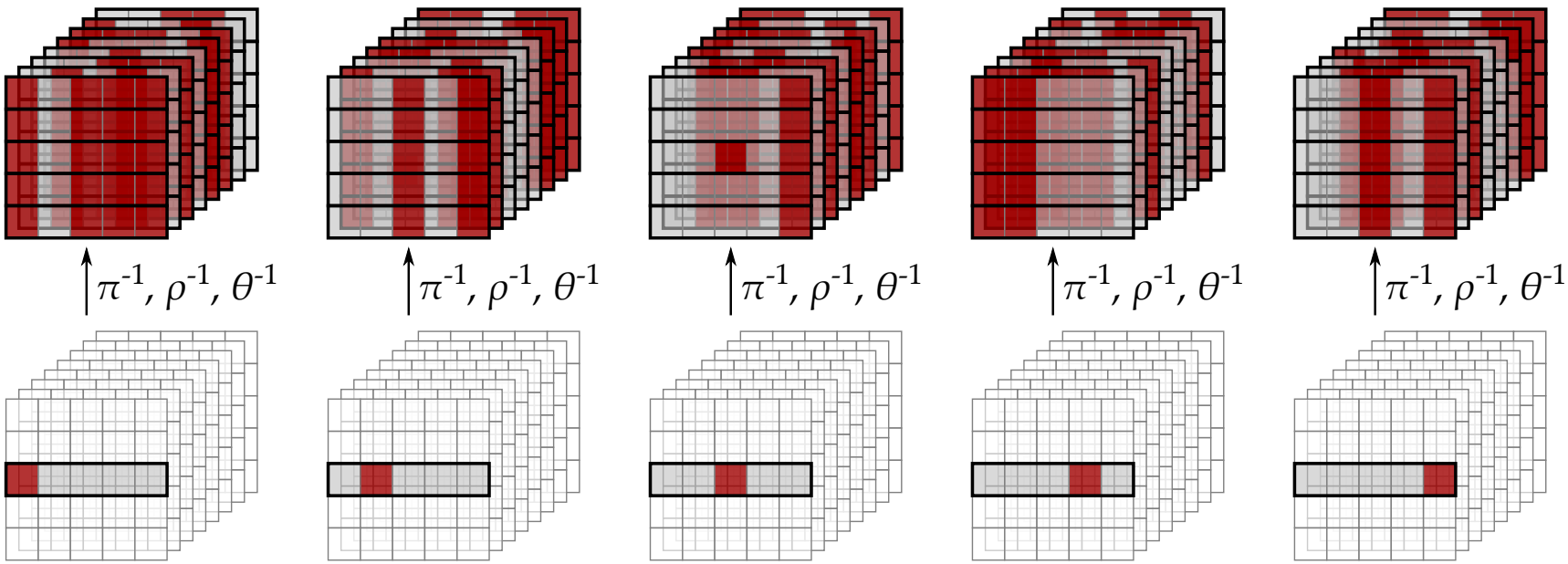
Alignment

- Property of round function [On alignment in KECCAK, Hash Workshop 2011]
  - relative to partition of state in blocks
- **Strong alignment**
  - Low uncertainty in propagation along block boundaries
  - E.g., RIJNDAEL strongly aligned on byte boundaries
- Weak alignment
  - High uncertainty in propagation along block boundaries
  - E.g., KECCAK weakly aligned on row boundaries...

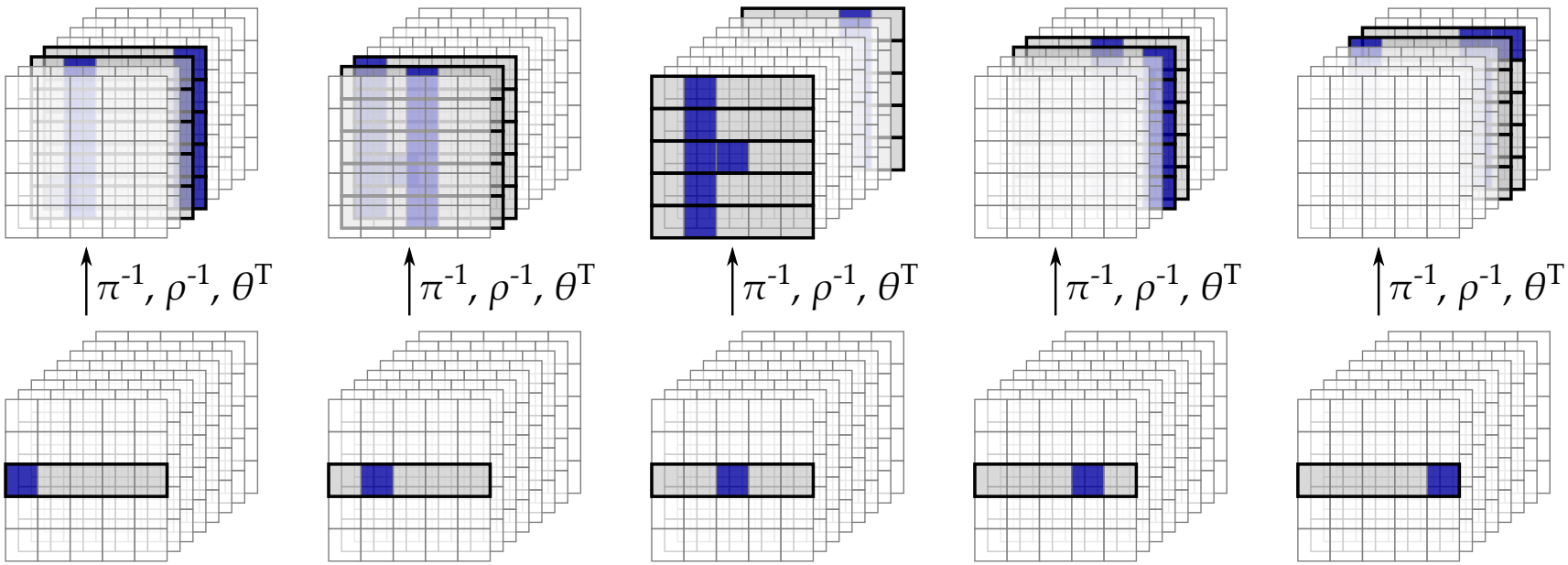
Differential patterns



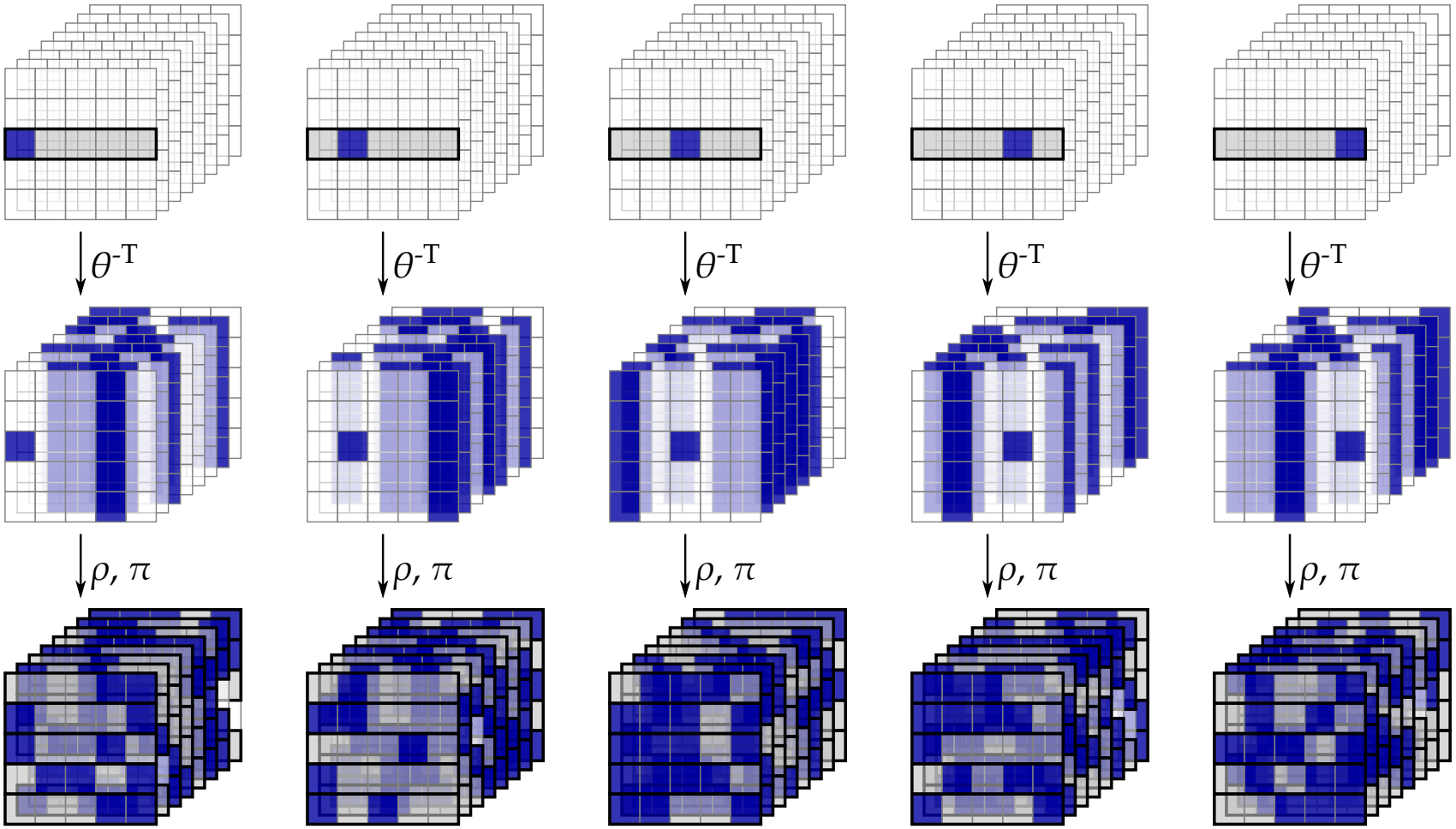
Differential patterns (backwards)



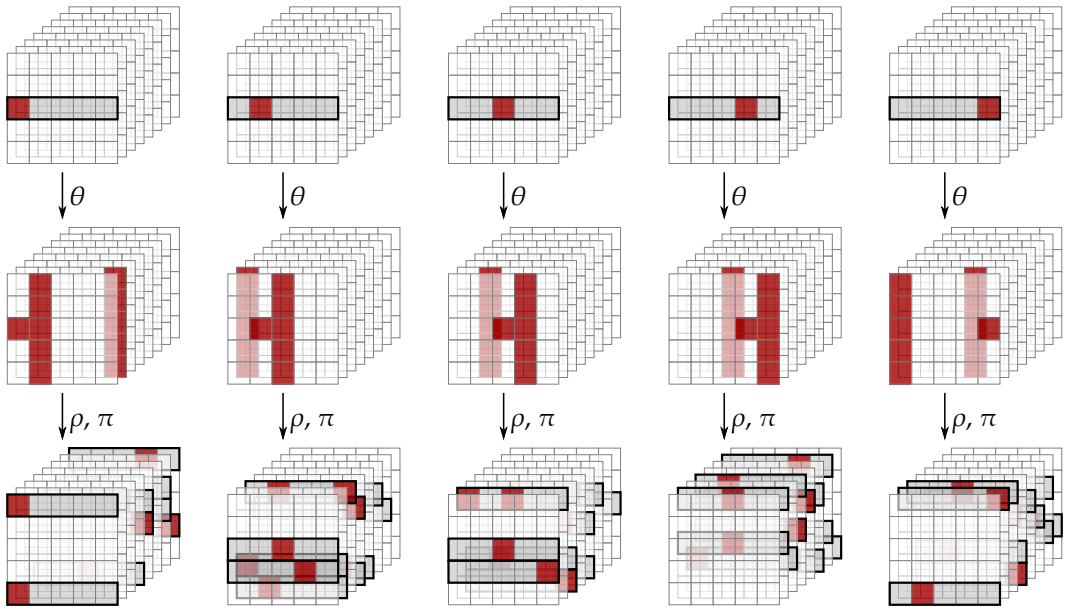
Linear patterns



Linear patterns (backwards)



Benefits of weak alignment



Weak alignment means trails tend to diverge

- Low clustering of trails
- Hard to build truncated differential trails
- Rebound attacks become very expensive

  e.g., [Duc et al., Unaligned Rebound Attack: Appl. to KECCAK, FSE 2012]

See [On alignment in KECCAK] for more details

Bounding differential and linear trail weights

Why bound trail weights?

- We want to base KECCAK security on **absence** of exploitable trails
  ...and not on presumed hardness of finding them
- Future: use of reduced-round versions of KECCAK-f

$\Rightarrow$ Find good bound on differential and linear trails

Bounds for differential and linear trails in Keccak-$f[b]$

- Tight bounds for Keccak-$f[25]$ to Keccak-$f[200]$ [The Keccak reference]
- Current bounds for differential trails in Keccak-$f[1600]$ [FSE 2012]

Rounds	Lower bound		Best known	
1	2		2	
2	8		8	
3	32	[Keccak team]	32	[Duc et al.]
4			134	[Keccak team]
5			510	[Naya-Plasencia et al.]
6	74	[Keccak team]	1360	[Keccak team]
24	296		???	

- Open problems:
  - Narrow the gap between bounds and known trails
  - Look more closely at Keccak-$f[400]$ and Keccak-$f[800]$
  - Bounds for linear trails in Keccak-$f[1600]$

Bounds for differential and linear trails in Keccak-$f[b]$

- Tight bounds for Keccak-$f[25]$ to Keccak-$f[200]$ [The Keccak reference]
- Current bounds for differential trails in Keccak-$f[1600]$ [FSE 2012]

Rounds	Lower bound		Best known	
1	2		2	
2	8		8	
3	32	[Keccak team]	32	[Duc et al.]
4			134	[Keccak team]
5			510	[Naya-Plasencia et al.]
6	74	[Keccak team]	1360	[Keccak team]
24	296		???	

- Open problems:
  - Narrow the gap between bounds and known trails
  - Look more closely at Keccak-$f[400]$ and Keccak-$f[800]$
  - Bounds for linear trails in Keccak-$f[1600]$

What textbooks and intro's should say from now on :-)

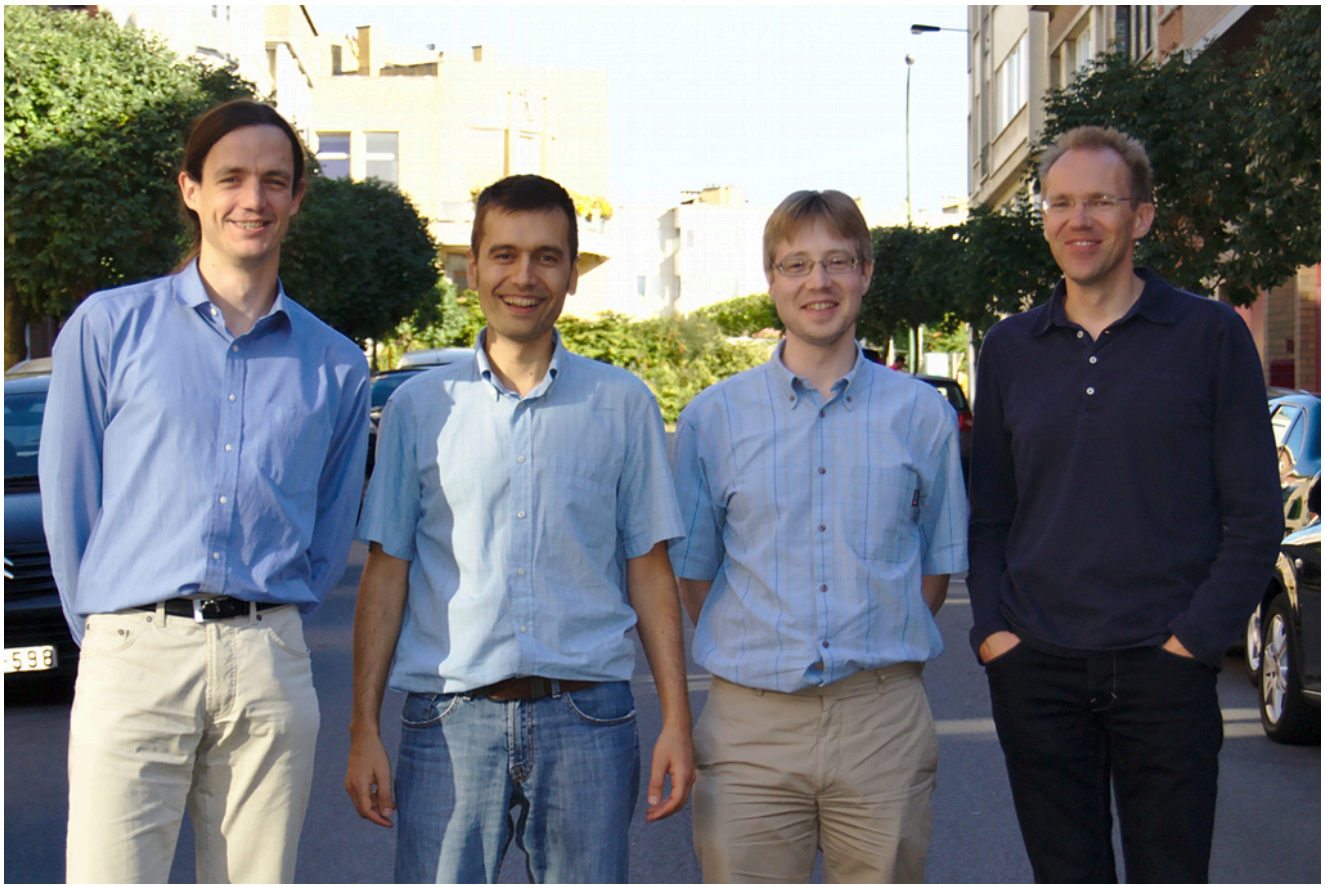
Symmetric cryptographic primitives:

- **Permutations**
- Block ciphers
- Stream ciphers
- Hash functions

And their modes-of-use

Picture by Sébastien Wiertz

Questions?



http://sponge.noekeon.org/
http://keccak.noekeon.org/

Our references

- Sakura: *a flexible coding for tree hashing*, ePrint 2013
- Debande, Le and KT, *PA of HW impl. protected with secret sharing*, HASP 2012
- *Permutation-based enc., auth. and auth. enc.*, DIAC 2012
- *Differential propagation in* Keccak, FSE 2012
- Van Keer and KT, Keccak *implementation overview* (version 3.1 or later)
- KeccakTools (version 3.2 or later)
- *Duplexing the sponge: authenticated enc. and other applications*, SAC 2011
- *On alignment in* Keccak, Ecrypt II Hash Workshop 2011
- *On the security of the keyed sponge construction*, SKEW 2011
- *The* Keccak *reference* (version 3.0 or later)
- *The* Keccak *SHA-3 submission*, 2011
- *Building power analysis resistant implementations of* Keccak, SHA-3 2010
- *Sponge-based pseudo-random number generators*, CHES 2010
- *Note on zero-sum distinguishers of* Keccak-*f*, NIST hash forum 2010
- *Note on* Keccak *parameters and usage*, NIST hash forum 2010
- *Sufficient conditions for sound tree and seq. hashing modes*, ePrint 2009
- *Note on side-channel attacks and their counterm...*, NIST hash forum 2009
- *The road from* Panama *to* Keccak *via* RadioGatún, Dagstuhl 2009
- *Cryptographic sponge functions* (version 0.1 or later)
- *On the indifferentiability of the sponge construction*, Eurocrypt 2008
- *Sponge functions*, comment to NIST and Ecrypt Hash Workshop 2007

http://sponge.noekeon.org/papers.html

http://keccak.noekeon.org/papers.html