Cryptographic properties of KECCAK

Gilles VAN ASSCHE[1]

[1]STMicroelectronics

Indocrypt, New Delhi, December 2018

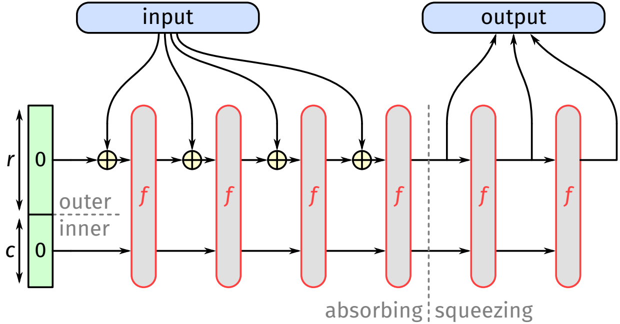
Based on joint work with Guido BERTONI, Joan DAEMEN, Silvia MELLA, Michaël PEETERS, Ronny VAN KEER

Outline

Outline

Keccak

Keccak is a *sponge function* ...



... that uses the Keccak-f permutation

$$b = r + c \in \{25, 50, 100, 200, 400, 800, 1600\}$$

NIST FIPS 202 (August 2015)

- Four drop-in replacements to SHA-2
- Two *extendable output functions* (XOF)

XOF	**SHA-2 drop-in replacements**
$\text{Keccak}[c = 256](M\|\|11\|\|11)$	
	first 224 bits of $\text{Keccak}[c = 448](M\|\|01)$
$\text{Keccak}[c = 512](M\|\|11\|\|11)$	first 256 bits of $\text{Keccak}[c = 512](M\|\|01)$
	first 384 bits of $\text{Keccak}[c = 768](M\|\|01)$
	first 512 bits of $\text{Keccak}[c = 1024](M\|\|01)$
SHAKE128 and **SHAKE256**	**SHA3-224** to **SHA3-512**

- Toolbox for building other functions

NIST SP 800-185 (December 2016)

Customized SHAKE (**cSHAKE**)

- $H(x) = \text{cSHAKE}(x, \text{name}, \text{customization string})$
- E.g., $\text{cSHAKE128}(x, N, S) = \textsc{Keccak}[c = 256](\text{encode}(N, S)||x||\texttt{00})$
- $\text{cSHAKE128}(x, N, S) \triangleq \text{SHAKE128}$ when $N = S =$ “”

KMAC: message authentication code (no need for HMAC-SHA-3!)

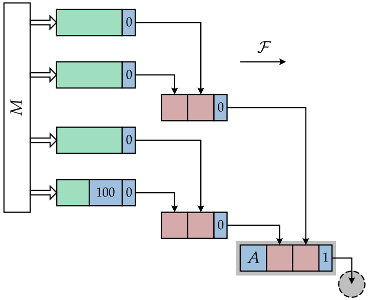
$$\text{KMAC}(K, x, S) = \text{cSHAKE}(\text{encode}(K)||x, \text{"KMAC"}, S)$$

TupleHash: hashing a sequence of strings $\mathbf{x} = x_n \circ x_{n-1} \circ \cdots \circ x_1$

$$\text{TupleHash}(\mathbf{x}, S) = \text{cSHAKE}(\text{encode}(\mathbf{x}), \text{"TupleHash"}, S)$$

NIST SP 800-185 (December 2016)

ParallelHash: faster hashing with parallelism



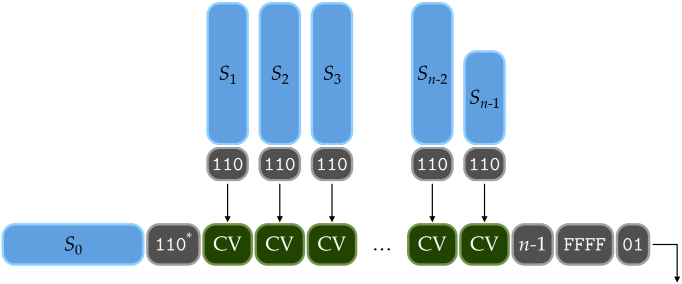
function	instruction set	cycles/byte
KECCAK$[c = 256] \times 1$	x86_64	6.29
KECCAK$[c = 256] \times 2$	AVX2	4.32
KECCAK$[c = 256] \times 4$	AVX2	2.31

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

KANGAROOTWELVE: a fast variant of KECCAK

Similar to ParallelHash, but:

- Same permutation, reduced round count (12 instead of 24)
- Kangaroo hopping



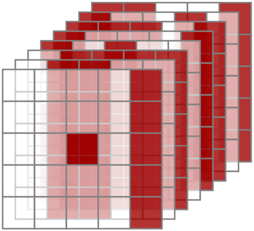
[KECCAK Team, Viguier, ACNS 2018]

Other schemes using Keccak-p

- Ketje: lightweight authenticated encryption
  Keccak-$p[200 \text{ or } 400]$ in MonkeyDuplex
- Keyak: authenticated encryption
  Keccak-$p[800 \text{ or } 1600]$ in full-state keyed duplex
- Kravatte: pseudo-random function and full-feature AE
  Keccak-$p[1600]$ in Farfalle

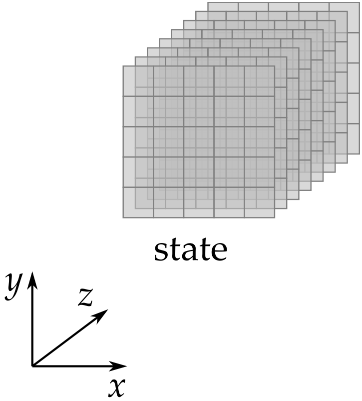
Outline

KECCAK-f



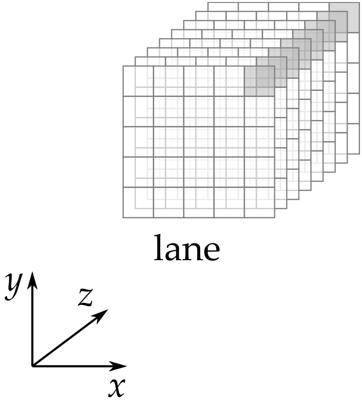
- The seven permutation army:
  - 25, 50, 100, 200, 400, 800, 1600 bits
  - toy, lightweight, fastest
  - standardized in [FIPS 202]
- Repetition of a simple round function
  - that operates on a 3D state
  - (5×5) lanes
  - up to 64-bit each

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



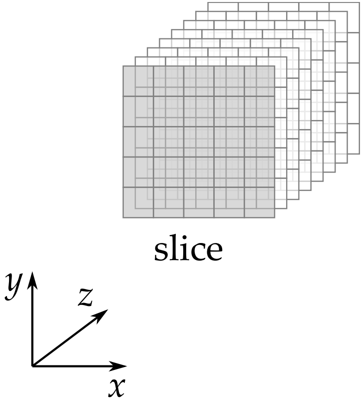
- 5×5 lanes, each containing 2^ℓ bits (1, 2, 4, 8, 16, 32 or 64)
- (5×5)-bit slices, 2^ℓ of them

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



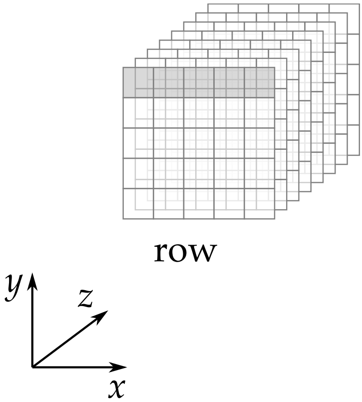
- 5×5 lanes, each containing 2^ℓ bits (1, 2, 4, 8, 16, 32 or 64)
- (5×5)-bit slices, 2^ℓ of them

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



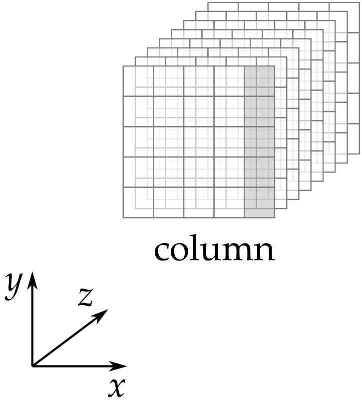
- 5×5 lanes, each containing 2^ℓ bits (1, 2, 4, 8, 16, 32 or 64)
- (5×5)-bit slices, 2^ℓ of them

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



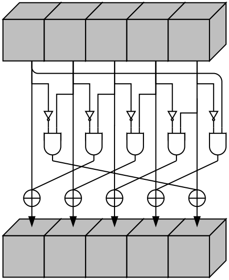
- 5×5 lanes, each containing 2^ℓ bits (1, 2, 4, 8, 16, 32 or 64)
- (5×5)-bit slices, 2^ℓ of them

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

The nonlinear layer χ

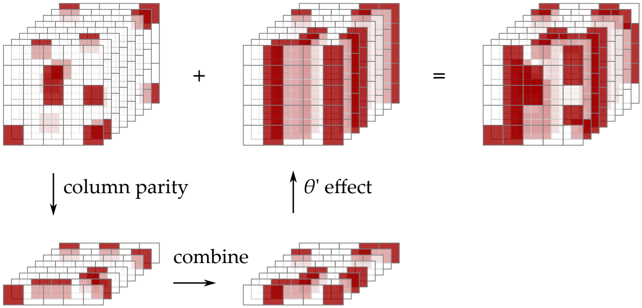


- "Flip bit if neighbors exhibit 01 pattern"
- Operates independently and in parallel on 5-bit rows
- Algebraic degree 2, inverse has degree 3

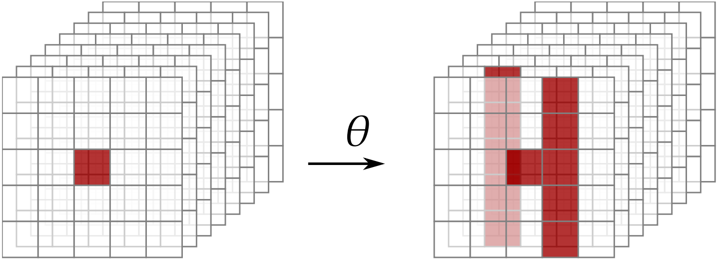
The mixing layer θ

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

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

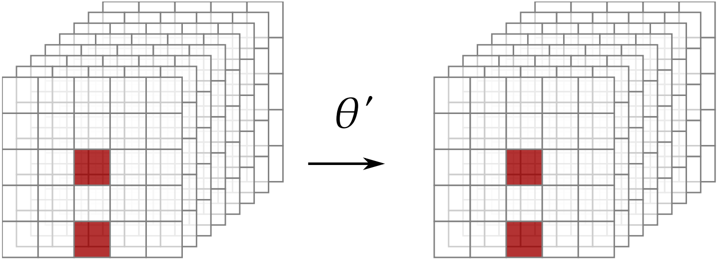


Difference propagation due to θ



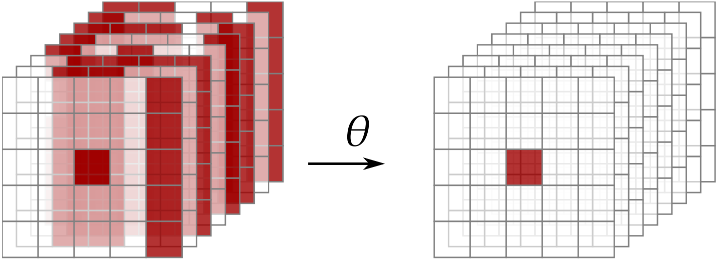
$$1 + \left(1 + y + y^2 + y^3 + y^4\right)\left(x + x^4 z\right)$$
$$\left(\bmod \left\langle 1 + x^5, 1 + y^5, 1 + z^w \right\rangle\right)$$

Difference propagation due to θ (kernel)



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

Inverse of θ is dense

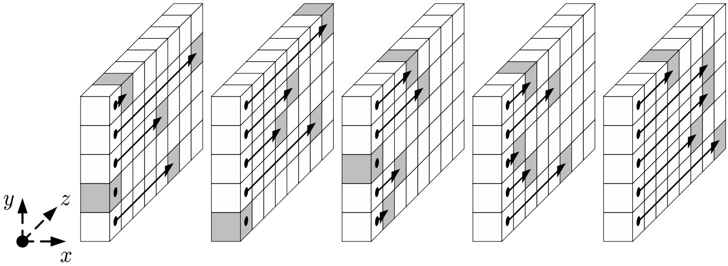


$$1 + \left(1 + y + y^2 + y^3 + y^4\right) \mathbf{Q},$$

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

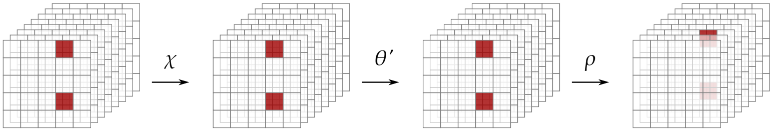
The inter-slice dispersion step ρ

- We need diffusion between the slices ...
- ρ: cyclic shifts of lanes
- Offsets cycle through all values below 2^ℓ



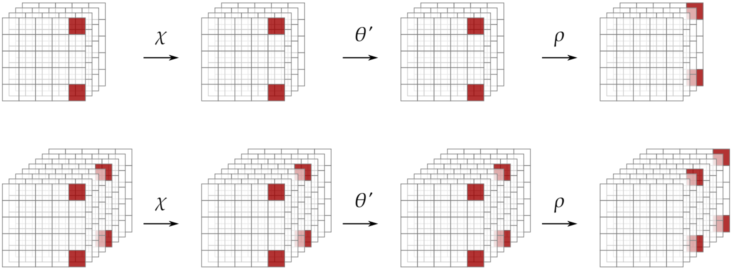
A first attempt at KECCAK-f

- Round function: $\mathrm{R} = \iota \circ \rho \circ \theta \circ \chi$
- 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

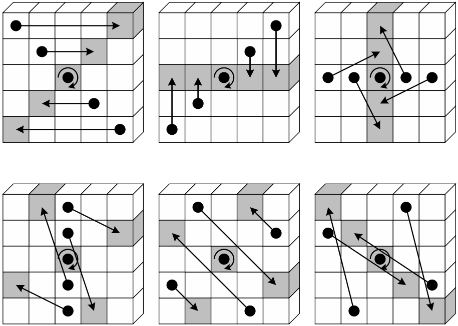
The Matryoshka property



- 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)

The intra-slice dispersion step π

We need to disturb horizontal/vertical alignment



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

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)

KECCAK-*f* summary

- Round function:
  - θ for diffusion
  - ρ for inter-slice dispersion
  - π for disturbing horizontal/vertical alignment
  - χ for non-linearity
  - ι to break symmetry

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

- Number of rounds: $12 + 2\ell$
  - KECCAK-$f[25]$ has 12 rounds
  - KECCAK-$f[1600]$ has 24 rounds

KECCAK-f in pseudo-code

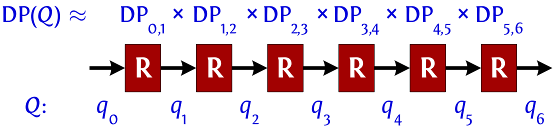
```
KECCAK-F[b](A) {
  forall i in 0…nr-1
    A = Round[b](A, RC[i])
  return A
}

Round[b](A,RC) {
  θ step
  C[x] = A[x,0] xor A[x,1] xor A[x,2] xor A[x,3] xor A[x,4], forall x in 0…4
  D[x] = C[x-1] xor rot(C[x+1],1),                            forall x in 0…4
  A[x,y] = A[x,y] xor D[x],                                   forall (x,y) in (0…4,0…4)

  ρ and π steps
  B[y,2*x+3*y] = rot(A[x,y], r[x,y]),                         forall (x,y) in (0…4,0…4)

  χ step
  A[x,y] = B[x,y] xor ((not B[x+1,y]) and B[x+2,y]),          forall (x,y) in (0…4,0…4)

  ι step
  A[0,0] = A[0,0] xor RC

  return A
}
```

https://keccak.team/keccak_specs_summary.html

Outline

Outline

Differential trails in iterated mappings

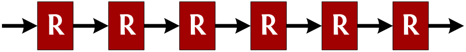


- Trail: sequence of differences
- $\mathrm{DP}(Q)$: fraction of pairs that exhibit differences q_i

Differential trails and weight

$$\mathrm{w} = -\log_2(DP)$$

$$w(Q) = w_{0,1} + w_{1,2} + w_{2,3} + w_{3,4} + w_{4,5} + w_{5,6}$$



Q: q_0 q_1 q_2 q_3 q_4 q_5 q_6

If *independent* rounds and $\mathrm{w}(Q) < b$: $\#\text{pairs}(Q) \approx 2^{b-\mathrm{w}(Q)}$

Goal

- Security of KECCAK-$f[b]$ relies **not** on presumed hardness of
  - finding low-weight trails
  - finding pairs given a trail Q
- But on hardness to exploit trails with at most a few pairs

KECCAK-$f[b]$ design goal

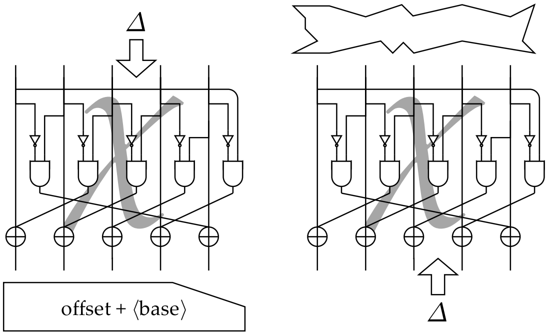
Absence of trails with $\mathrm{w}(Q) < b$

- Goal of this effort:
  - exhaustively generate trails up to some weight
  - to build assurance that there are no low-weight trails
  - inspired by similar efforts for Noekeon and MD6

	width	weight bound per round
Noekeon	128	12.0 [Nessie, 2000]
MD6	4096	2.5 [Rivest et al., 2008][Heilman, 2011]

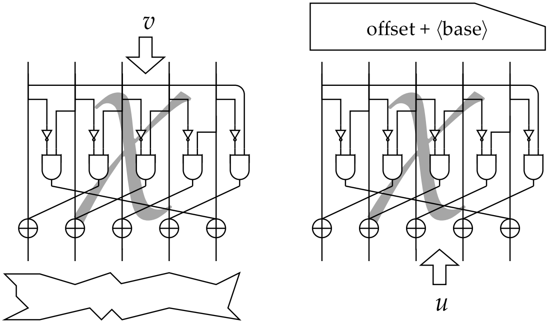
Outline

Propagating differences through χ



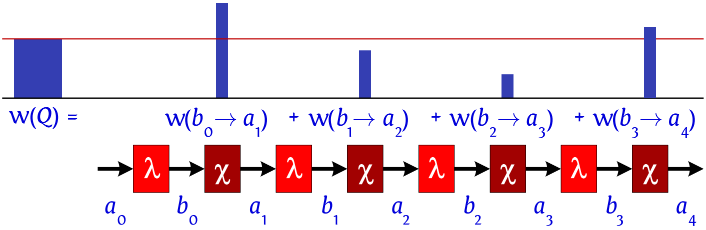
- The propagation weight...
  - ... is determined by input difference only;
  - ... is the size of the affine base;
  - ... is the number of affine conditions.

Propagating linear masks through χ



- The propagation weight...
  - ... is determined by output mask only;
  - ... is the size of the affine base.

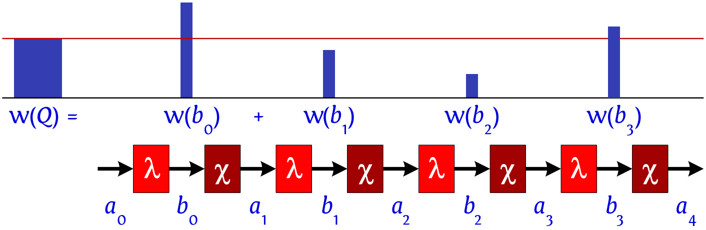
Trails in KECCAK-f



Round: linear step $\lambda = \pi \circ \rho \circ \theta$ and non-linear step χ

- a_i fully determines $b_i = \lambda(a_i)$
- $\mathrm{w}(Q) = \sum_i \mathrm{w}(b_{i-1} \xrightarrow{\chi} a_i)$

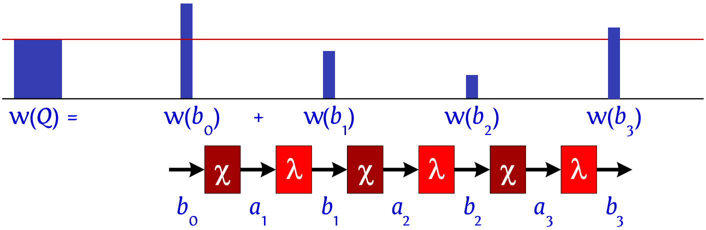
Trails in KECCAK-f



Nonlinear step χ has algebraic degree 2

- for input b_{i-1}, the outputs a_i form affine space $\mathcal{A}(b_{i-1})$
- dimension of $\mathcal{A}(b_{i-1})$ is $\mathrm{w}(b_{i-1}, a_i) = \mathrm{w}(b_{i-1})$

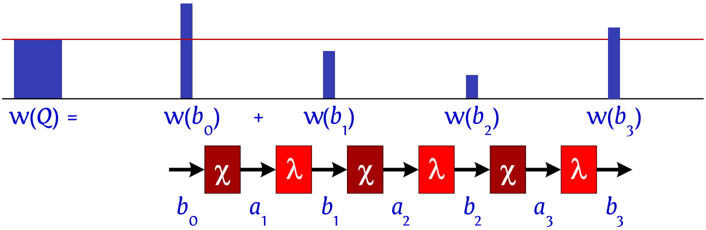
Trails in KECCAK-f



Trail weight fully determined by b_i

- We can ignore a_4: trail **prefix**
- We can ignore a_0

Trails in KECCAK-f



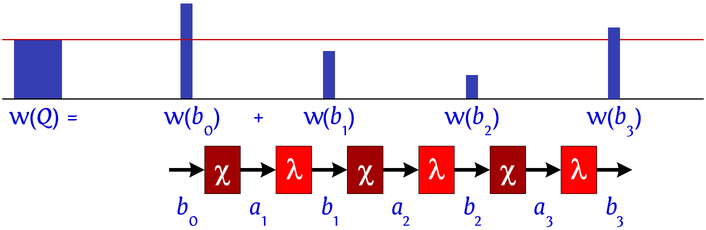
$\mathrm{w}(Q) > b$ now has a simple meaning:

$\mathrm{w}(Q)$: $\#$ conditions on intermediate state bits

b: $\#$ input bits

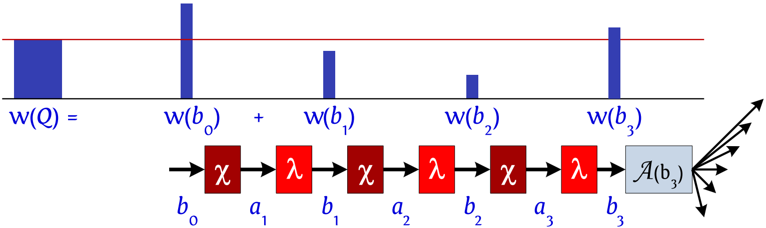
Trail extension

- Given a trail, we can extend it:
  - forward: iterate a_{r+1} over $\mathcal{A}(b_r)$
  - backward: iterate b_{-1} over all differences χ^{-1}-compatible with $a_0 = \lambda^{-1}(b_0)$
- Tree search:
  - extension can be done recursively
  - limited by total weight



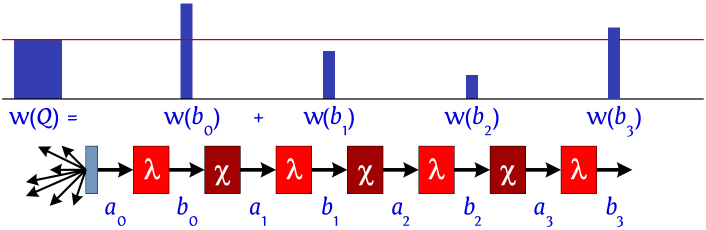
Trail extension

- Given a trail, we can extend it:
  - forward: iterate a_{r+1} over $\mathcal{A}(b_r)$
  - backward: iterate b_{-1} over all differences χ^{-1}-compatible with $a_0 = \lambda^{-1}(b_0)$
- Tree search:
  - extension can be done recursively
  - limited by total weight



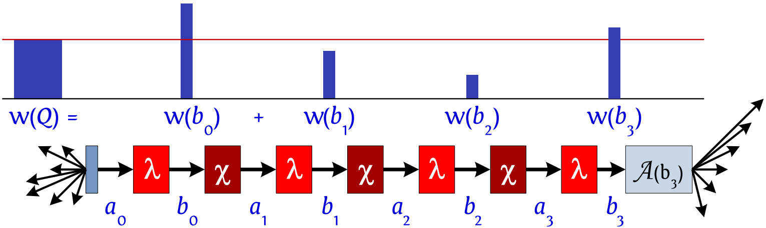
Trail extension

- Given a trail, we can extend it:
  - forward: iterate a_{r+1} over $\mathcal{A}(b_r)$
  - backward: iterate b_{-1} over all differences χ^{-1}-compatible with $a_0 = \lambda^{-1}(b_0)$
- Tree search:
  - extension can be done recursively
  - limited by total weight



Trail extension

- Given a trail, we can extend it:
  - forward: iterate a_{r+1} over $\mathcal{A}(b_r)$
  - backward: iterate b_{-1} over all differences χ^{-1}-compatible with $a_0 = \lambda^{-1}(b_0)$
- Tree search:
  - extension can be done recursively
  - limited by total weight

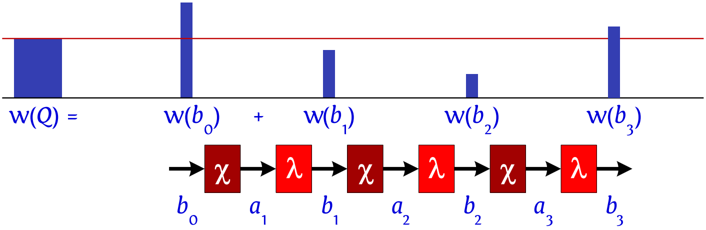


Trail core

- **Minimum reverse weight**: lower bound of weight given difference after χ

$$\mathrm{w}^{\mathrm{rev}}(a) \triangleq \min_{b\,:\,a \in \mathcal{A}(b)} \mathrm{w}(b)$$

- Can be used to lower bound of set of trails
- Trail **core**: set of trails with $b_1, b_2, \ldots$ in common

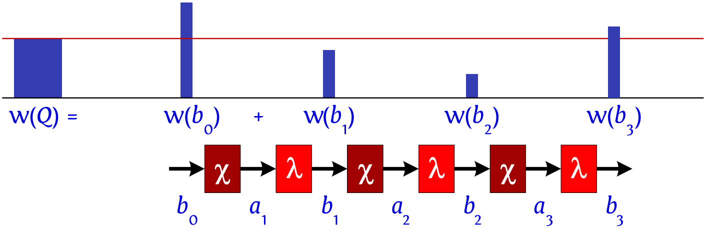


Trail core

- **Minimum reverse weight**: lower bound of weight given difference after χ

$$\mathrm{w}^{\mathrm{rev}}(a) \triangleq \min_{b\,:\,a \in \mathcal{A}(b)} \mathrm{w}(b)$$

- Can be used to lower bound of set of trails
- Trail **core**: set of trails with $b_1, b_2, \ldots$ in common

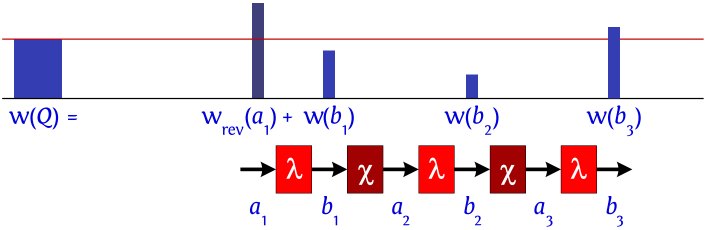


Trail core

- **Minimum reverse weight**: lower bound of weight given difference after χ

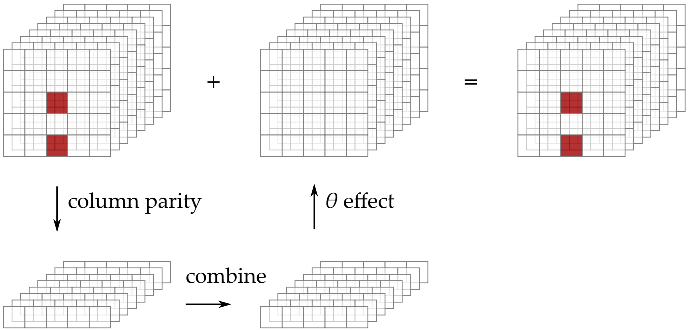
$$\mathrm{w}^{\mathrm{rev}}(a) \triangleq \min_{b\,:\,a \in \mathcal{A}(b)} \mathrm{w}(b)$$

- Can be used to lower bound of set of trails
- Trail **core**: set of trails with $b_1, b_2, \ldots$ in common



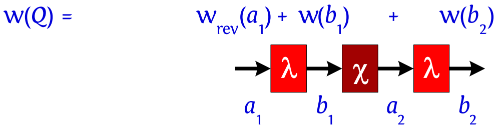
Outline

The parity kernel



- θ acts as the identity if parity is zero
- A state with parity zero is in the kernel (or in $|K|$)
- A state with parity non-zero is outside the kernel (or in $|N|$)

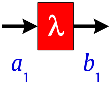
Covering the space of 3-round trail cores



- Space split based on parity of a_i
- Four classes: $|K|K|$, $|K|N|$, $|N|K|$ and $|N|N|$

Covering the space of 3-round trail cores

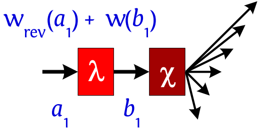
$w(Q) = w_{rev}(a_1) + w(b_1)$



- Generating (a_1, b_1)
- Extending forward by one round

Covering the space of 3-round trail cores

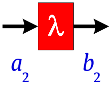
$w(Q) =$



- Generating (a_1, b_1)
- Extending forward by one round

Covering the space of 3-round trail cores

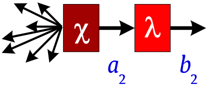
$\mathrm{w}(Q) = \qquad \mathrm{w}_{\mathrm{rev}}(a_2) + \mathrm{w}(b_2)$



- Generating (a_2, b_2)
- Extending backward by one round

Covering the space of 3-round trail cores

$w(Q) =$ $w_{rev}(a_2) + w(b_2)$



- Generating (a_2, b_2)
- Extending backward by one round

Outline

Definition

Set U of *units* with a total order relation $\prec$

Tree

- Node: subset of U, represented as a *unit list*

$$a = (u_i)_{i=1,\ldots,n} \quad u_1 \prec u_2 \prec \cdots \prec u_n$$

- Children of a node a:

$$a \cup \{u_{n+1}\} \quad \forall\, u_{n+1} \,:\, u_n \prec u_{n+1}$$

- Root: the empty set $a = \emptyset$

Example

- $U = \{(x, y, z)\}$, the coordinates in the KECCAK-f state
- $\prec$ is $[x, y, z]$ the lexicographical order in x, then y, then z
- State value a as unit list $(x_i, y_i, z_i)_{i=1,\ldots,n}$ for all $a_{x,y,z} = 1$

Bounding the cost

Goal: tree traversal up to given cost target T

Cost-related functions

- Cost function $w(a)$
- Subtree bounding function $L(a)$

$$L(a) \leq w(a') \quad \text{for all descendants } a' \text{ of } a$$

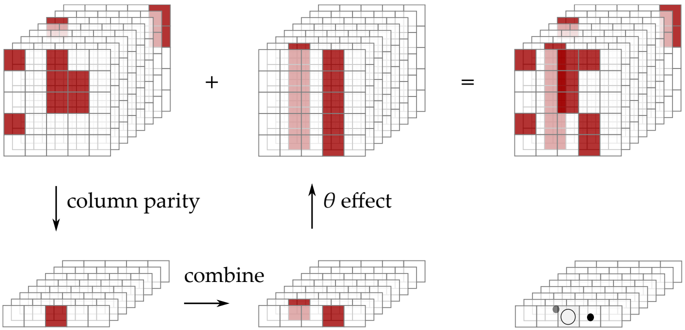
$\Rightarrow$ Skip all the subtrees with $L(a) > T$

Example (continued)

- $U = \{(x, y, z)\}$, the coordinates in the KECCAK-f state
- $\prec$ is $[x, y, z]$ the lexicographical order in x, then y, then z
- State value a as unit list $(x_i, y_i, z_i)_{i=1,\ldots,n}$ for all $a_{x,y,z} = 1$

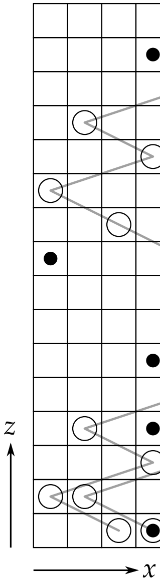
- Cost $w(a)$: Hamming, differential or restriction weight
- $L(a) = w(a)$ due to monotonicity in KECCAK-f

Properties of θ



- Affected columns are complemented
- Unaffected columns are not changed

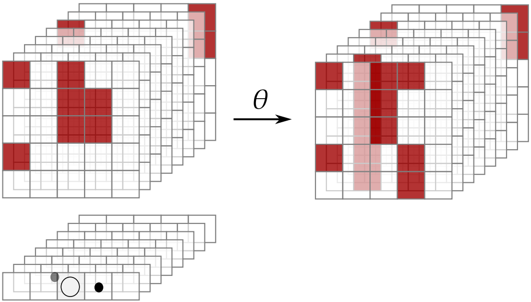
Properties of θ (continued)



Parity-bare state and orbitals

Lemma

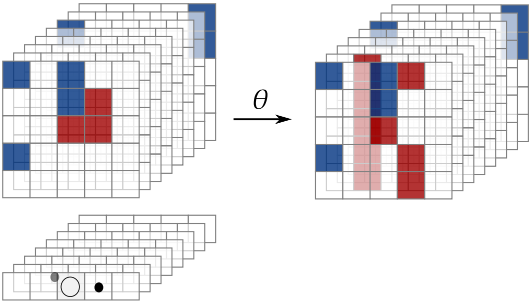
Each state can be decomposed in a unique way in a parity-bare state and a list of orbitals



Parity-bare state and orbitals

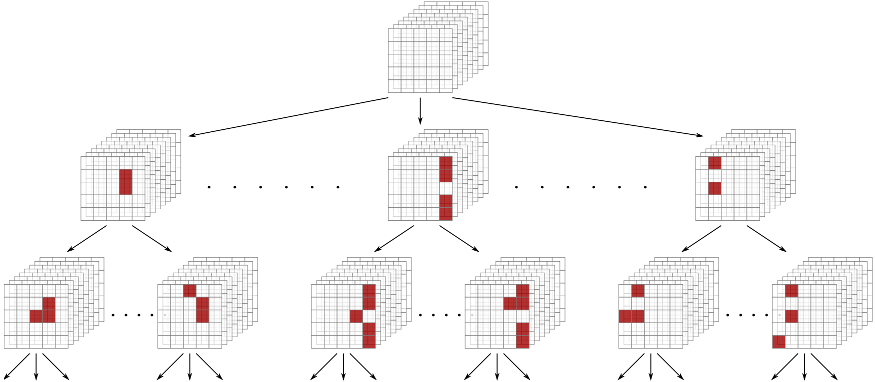
Lemma

Each state can be decomposed in a unique way in a parity-bare state and a list of orbitals



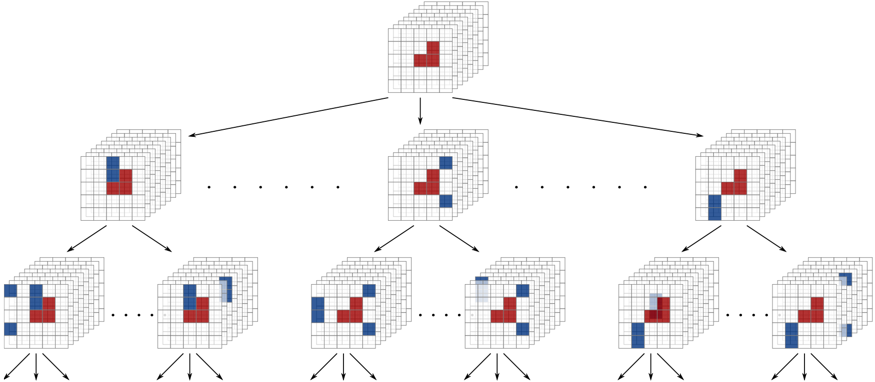
Generating parity-bare states

- Root: the empty state
- Units: column assignments $(x, z, \text{odd}/\text{affected}, \text{column value})$
- Bound: weight minus potential loss due to new CAs



Completing with orbitals

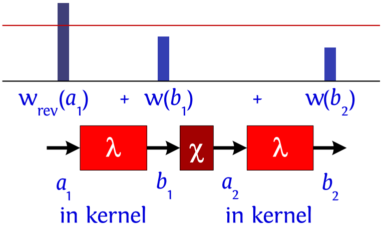
- Root: a parity-bare state
- Units: orbitals in unaffected columns (x, y_1, y_2, z)
- Bound: weight of the trail itself



Outline

Dealing with the kernel

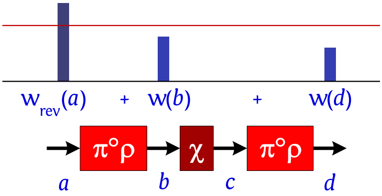
- Problem: too many states in $|K|$
- Problematic case:



- Trail cores (b, d) with $w^{rev}(a) + w(b) + w(d) \leq 3T/r$
  - $a = \lambda^{-1}(b)$ is in the kernel
  - intersection of $\mathcal{A}(b)$ and kernel is not empty
  - *b is tame*

Dealing with the kernel

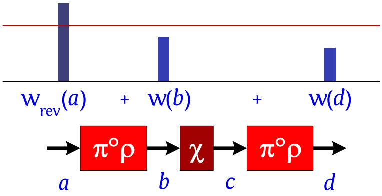
- Problem: too many states in $|K|$
- Problematic case:



- Trail cores (b, d) with $\mathrm{w}^{\mathrm{rev}}(a) + \mathrm{w}(b) + \mathrm{w}(d) \leq 3T/r$
  - $a = \lambda^{-1}(b)$ is in the kernel
  - intersection of $\mathcal{A}(b)$ and kernel is not empty
  - b *is tame*

Dealing with the kernel

- Problem: too many states in $|K|$
- Problematic case:



- Trail cores (b, d) with $\mathrm{w}^{\mathrm{rev}}(a) + \mathrm{w}(b) + \mathrm{w}(d) \leq 3T/r$
  - $a = \lambda^{-1}(b)$ is in the kernel
  - intersection of $\mathcal{A}(b)$ and kernel is not empty
  - *b is tame*

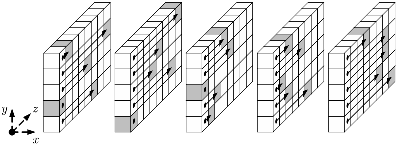
Third-order approach: tame states

- Condition that a is in kernel
  - One-to-one mapping of active bit positions between a and b
  - translate conditions to b
- Tameness of slices of b
  - empty slice is tame
  - single-bit slice cannot be tame
  - two-bit slice is tame iff bits are in same column (**orbital**)
  - more than 2 bits: **knot**
- **Chains**: sequences of active bits p_i that:
  - start and end in a knot
  - p_{2i} and p_{2i+1} are in same column in a
  - p_{2i+1} and p_{2i} are in same column in b

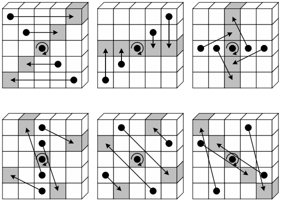
ρ, π and chains

Bit transpositions ρ and π

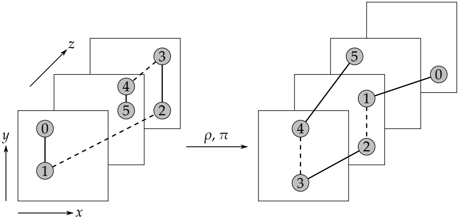
- ρ: inter-slice



- π: intra-slice



Example of a chain:



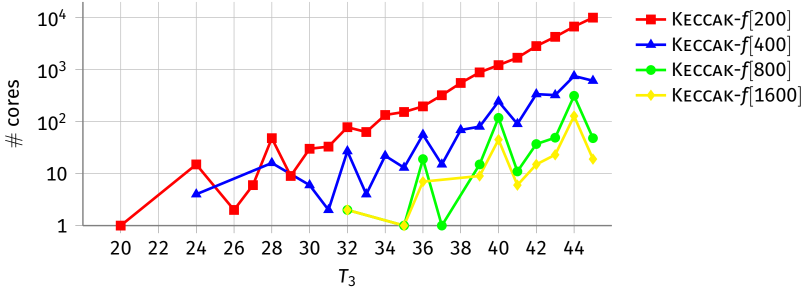
Third-order approach

- Representation of tame states:
  - set of chains between knots
  - plus some circular chains: **vortices**
- Efficiently iterating over tame states:
  - start from empty state
  - recursively add chains and vortices until predicted weight exceeds $3T/r$
  - if all knots are tame, valid output
- Full coverage guaranteed by
  - monotonous weight prediction function
  - well-defined order of chains

Outline

Summary of current results

- All 3-round trail cores with weight ≤ 45



- No 6-round trail with weight ≤ 91

Summary of current results (cont'd)

rounds	$b = 200$	$b = 400$	$b = 800$	$b = 1600$
2	8	8	8	8
3	20	24	32	32
4	46	[48,63]	[48,104]	[48,134]
5	[50,89]	[50,147]	[50,247]	[50,372]
6	[92,142]	[92,278]	[92,556]	[92,1112]

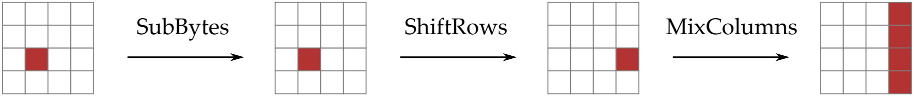
Table: Current bounds for the minimum weight of differential trails

Outline

Outline

Difference propagation in RIJNDAEL: strong alignment

- Propagation of differences:
  - `MixColumns`, `ShiftRows` and `AddRoundKey`: 1-to-1
  - `SubBytes`: 1-to-N
    - state with x active bytes at input: $N = 126^x \approx 2^{7x}$
- Propagation of truncated differences (active/passive bytes)
  - `SubBytes`, `ShiftRows` and `AddRoundKey`: 1-to-1
  - `MixColumns`: 1-to-N
    - column with 1 active bytes at input: $N = 1$
    - column with 2 active bytes in input: $N = 5$
    - column with 3 active bytes in input: $N = 11$
    - column with 4 active bytes in input: $N = 15$

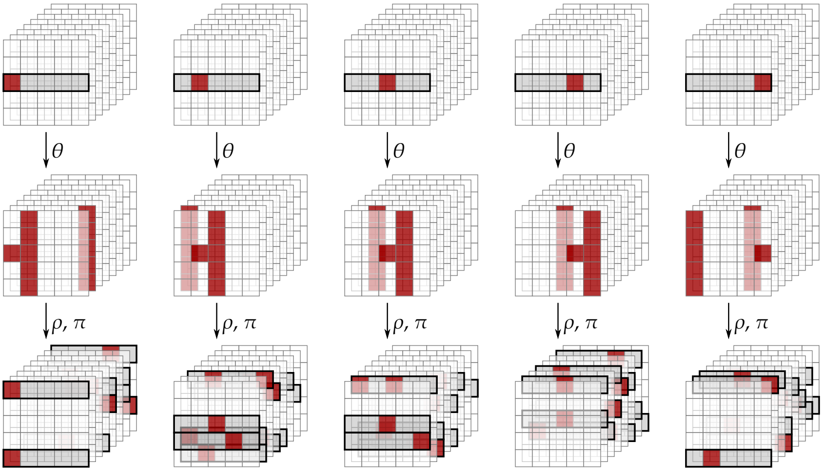


Alignment

- Property of round function
  - 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...

Outline

Differential patterns



Attempt at quantifying alignment

For a given input activity pattern (specified in blocks)

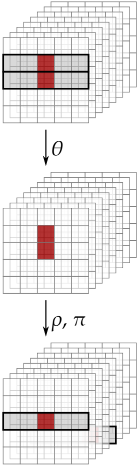
- N: number of possible different output activity patterns
  - e.g., `MixColumns` 1 active byte: $N = 1$ (4 active bytes)
  - e.g., `MixColumns` 4 active bytes: $N = 15$ (1-4 active bytes)
- $h = -\sum_z \Pr(z|A) \log_2 \Pr(z|A)$: "entropy"
  - e.g., `MixColumns` 4 active bytes: $h \approx 0$ (most often 4)
- $\overline{w}$: average number of active blocks
  - e.g., `MixColumns` 4 active bytes: $\overline{w} \approx 4$ (most often 4)

Row activity: typical results

Output row-activity for single-row differences in row $y = 0$ at round input:

2^ℓ	N	h	$\overline{w}$
1	1	0.00	5.00
2	11	1.97	9.35
4	26	4.60	15.54
8	31	4.95	19.22
16	31	4.95	23.09
32	31	4.95	25.29
64	31	4.95	25.54

Differential patterns (kernel)

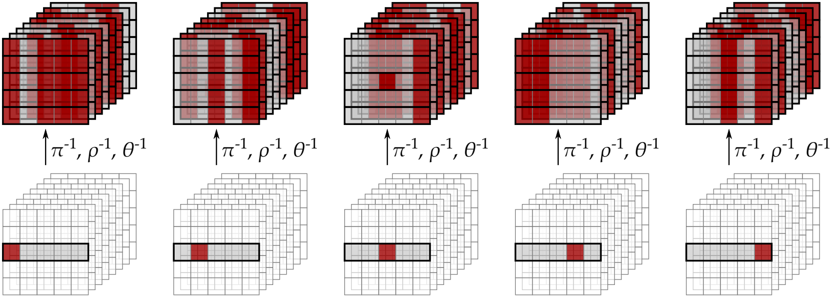


Slice activity: the results

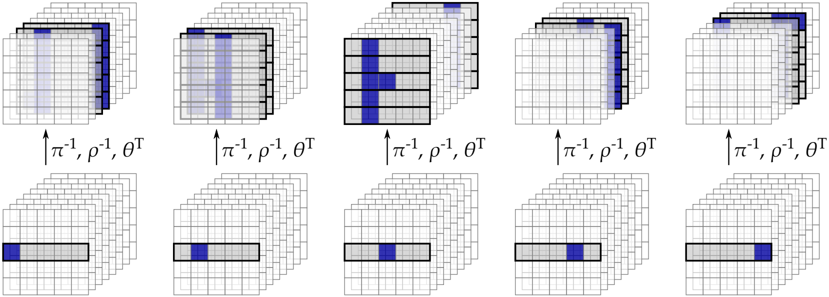
Output slice-activity for single-slice differences at round input:

2^ℓ	full single-slice set			in-kernel subset		
	N	h	$\overline{w}$	N	h	$\overline{w}$
1	1	0.00	1.00	1	0.00	1.00
2	3	0.0002	1.99	3	0.005	1.99
4	15	0.04	3.99	15	0.41	3.94
8	247	0.98	7.85	247	4.14	7.06
16	50622	7.86	13.93	49999	14.18	10.25
32	5611775	19.66	20.25	1048575	20.00	12.50
64	12599295	22.87	22.50	1048575	20.00	12.50

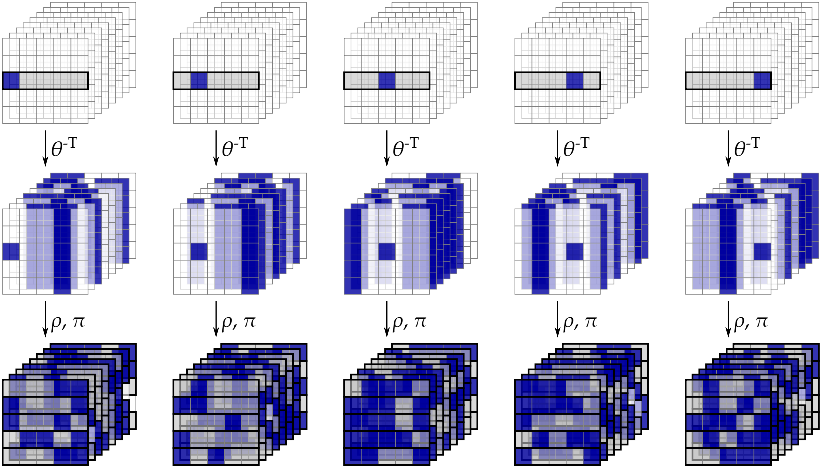
Differential patterns (backwards)



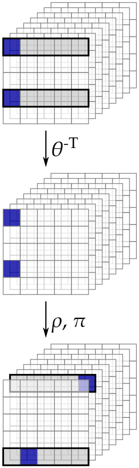
Linear patterns



Linear patterns (backwards)



Linear patterns (backwards, kernel)

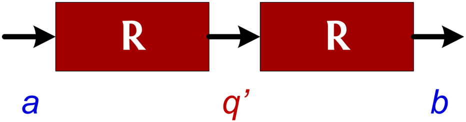


Outline

Strong versus weak alignment

- Benefits of strong alignment
  - propagation analysis easy to describe and understand
  - strong trail bounds with simple proofs, e.g. 4R AES: 25 S-boxes
  - allows efficient table-lookup implementations
- Benefits of weak alignment
  - low clustering of trails
  - hard to build truncated differential trails
  - rebound attacks become very expensive
- impacts how attacks work: integral, impossible, zero-correlation, ...

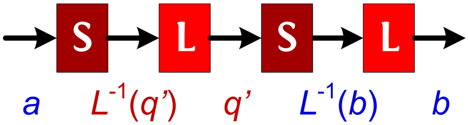
Clustering of differential trails



$$\mathrm{DP}_{2R}(a,b) = \sum_{Q \in (a,b)} DP(Q) \approx \sum_{q'} \mathrm{DP}_{R}(a,q')\mathrm{DP}_{R}(q',b)$$

- Necessary conditions for a trail Q to contribute to (a,b):
  - a and q have same S-box activity pattern
  - b' and $L(q)$ have same S-box activity pattern
- Relevance of alignment of L along S-box boundaries:
  - strong alignment: $L(q)$ has low variety in activity pattern
  - weak alignment: $L(q)$ has wide variety in activity pattern
- Similar arguments apply for correlations and linear trails

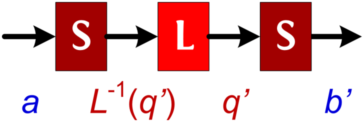
Clustering of differential trails



$$\mathrm{DP}_{2\mathrm{R}}(a,b) = \sum_{Q \in (a,b)} DP(Q) \approx \sum_{q'} \mathrm{DP}_{\mathrm{R}}(a,q')\mathrm{DP}_{\mathrm{R}}(q',b)$$

- Necessary conditions for a trail Q to contribute to (a,b):
  - a and q have same S-box activity pattern
  - b' and $L(q)$ have same S-box activity pattern
- Relevance of alignment of L along S-box boundaries:
  - strong alignment: $L(q)$ has low variety in activity pattern
  - weak alignment: $L(q)$ has wide variety in activity pattern
- Similar arguments apply for correlations and linear trails

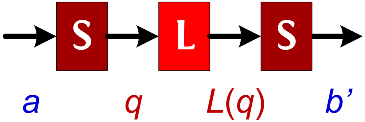
Clustering of differential trails



$$\mathrm{DP}_{2\mathrm{R}}(a,b) = \sum_{Q \in (a,b)} DP(Q) \approx \sum_{q'} \mathrm{DP}_{\mathrm{R}}(a,q')\mathrm{DP}_{\mathrm{R}}(q',b)$$

- Necessary conditions for a trail Q to contribute to (a, b):
  - a and q have same S-box activity pattern
  - b' and $L(q)$ have same S-box activity pattern
- Relevance of alignment of L along S-box boundaries:
  - strong alignment: $L(q)$ has low variety in activity pattern
  - weak alignment: $L(q)$ has wide variety in activity pattern
- Similar arguments apply for correlations and linear trails

Clustering of differential trails



$$\mathrm{DP}_{2\mathrm{R}}(a,b) = \sum_{Q \in (a,b)} DP(Q) \approx \sum_{q} \mathrm{DP}_{\mathrm{S}}(a,q)\mathrm{DP}_{\mathrm{S}}(L(q),b')$$

- Necessary conditions for a trail Q to contribute to (a, b):
  - a and q have same S-box activity pattern
  - b' and $L(q)$ have same S-box activity pattern
- Relevance of alignment of L along S-box boundaries:
  - strong alignment: $L(q)$ has low variety in activity pattern
  - weak alignment: $L(q)$ has wide variety in activity pattern
- Similar arguments apply for correlations and linear trails

Truncated differentials and rebound attacks

- Weak alignment means trails tend to diverge
  - low clustering of differential trails
  - hard to construct a truncated differential trail
- Open question for KECCAK
  - generalize truncation other than on block boundaries?
- Rebound attack typically requires truncated trails
  - it can also be done exploiting saturation
    [Duc et al., Unaligned Rebound Attack: Appl. to KECCAK, FSE 2012]
  - still rather expensive

Outline

What is KECCAKTOOLS?

A set of documented C++ classes to help analyze KECCAK-f

You said "documented"?

- Documentation in Doxygen format
- Various example routines in `main.cpp`
- Sample of differential and linear trails

https://github.com/KeccakTeam/KeccakTools

Functionality overview

- Seven permutations, from Keccak-$f[25]$ to Keccak-$f[1600]$
  - Individual steps θ, ρ, π, χ and ι
  - And all the **inverses** $\iota^{-1} = \iota$, χ^{-1}, π^{-1}, ρ^{-1} and θ^{-1}
- Sponge construction on any permutation
- Equations in GF(2) of rounds or steps
- Optimized C code (lane complementing, bit interleaving)
  - Macros currently in our optimized implementations
- Differential and linear cryptanalysis

The seven permutation army

Instantiating and using Keccak-*f*

```
KeccakF f(200);

KeccakF g(800, 123);
vector<LaneValue> state(25, 0);
g.forward(state);
g.inverseRound(state);
g.chi(state);
g.inverseTheta(state);
```

- Two ways to represent the state:
  - `vector<LaneValue>`, 25 lanes
  - `vector<SliceValue>`, from 1 to 64 slices

Variable naming convention

Lane naming convention

	$x=0$	$x=1$	$x=2$	$x=3$	$x=4$
$y=0$	ba	be	bi	bo	bu
$y=1$	ga	ge	gi	go	gu
$y=2$	ka	ke	ki	ko	ku
$y=3$	ma	me	mi	mo	mu
$y=4$	sa	se	si	so	su

- z coordinate as a suffix
  - E.g., bu21 is bit at $x=4$, $y=0$ and $z=21$
- Alphabetical order = bit ordering at sponge level
  - Makes it easier to express concrete CICO problems (preimage, etc.)

Examples of generated equations

Equations for θ and θ^{-1} in Keccak-$f[100]$

```
Oba0 = Iba0 + Ibe3 + Ige3 + Ike3 + Ime3 + Ise3 +
Ibu0 + Igu0 + Iku0 + Imu0 + Isu0

Iba0 = Oba0 + Obi0 + Ogi0 + Oki0 + Omi0 + Osi0 +
Obo3 + Ogo3 + Oko3 + Omo3 + Oso3 + Obi3 + Ogi3 +
Oki3 + Omi3 + Osi3 + Oba2 + Oga2 + Oka2 + Oma2 +
Osa2 + Obo2 + Ogo2 + Oko2 + Omo2 + Oso2 + Obi2 +
Ogi2 + Oki2 + Omi2 + Osi2 + Obe2 + Oge2 + Oke2 +
Ome2 + Ose2 + Oba1 + Oga1 + Oka1 + Oma1 + Osa1 +
Obo1 + Ogo1 + Oko1 + Omo1 + Oso1 + Obe1 + Oge1 +
Oke1 + Ome1 + Ose1
```

Examples of generated equations

Equations for χ and χ^{-1} in Keccak-$f[100]$

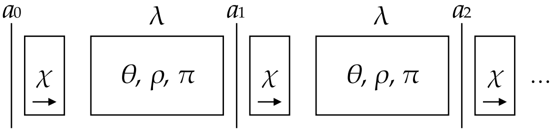
```
Obo2 = Ibo2 + (Ibu2 + 1)*Iba2

Ibo2 = Obo2 + (Oba2 + Obi2*(Obe2 + 1))*(Obu2 + 1)
```

Equations for full round in Keccak-$f[100]$

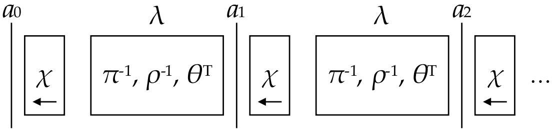
```
Bgo3 = Ame2 + Abi1 + Agi1 + Aki1 + Ami1 + Asi1 +
Aba2 + Aga2 + Aka2 + Ama2 + Asa2 + (Asi2 + Abo1 +
Ago1 + Ako1 + Amo1 + Aso1 + Abe2 + Age2 + Ake2 +
Ame2 + Ase2 + 1)*(Abo3 + Abu2 + Agu2 + Aku2 + Amu2 +
Asu2 + Abi3 + Agi3 + Aki3 + Ami3 + Asi3)
```

Differential and linear trails

- Trail: states $(a_0, a_1, \ldots)$ "before" χ
- Propagation through "affine" direction:
  - Differential trails



  - Linear trails: forward propagation means backwards in time



Differential or linear propagation context?

- Class `KeccakFDCLC`
  - Inherits from `KeccakF`
  - Computes all the propagation tables, both DC and LC
- Class `KeccakFPropagation`
  - Uses `KeccakFDCLC`
  - Specializes in either DC or LC
  - Allows uniform implementation of trail search

Class instantiation for DC and LC

```
KeccakFDCLC f(200);
KeccakFPropagation DC(f, KeccakFPropagation::DC);
...
KeccakFPropagation LC(f, KeccakFPropagation::LC);
...
```

Displaying trails

- Examples of trails in package
- Text files, one trail per line

Checking and displaying trails

```
KeccakFDCLC f(50);
KeccakFPropagation DC(f, KeccakFPropagation::DC);
fileName = DC.buildFileName(“-trails”);
Trail::produceHumanReadableFile(DC, fileName);
```

Forward propagation

Example: extending a linear trail

```
Trail trail(inputFile);
const vector<SliceValue>& lastStateOfTrail =
trail.states.back();
affineSpace = LC.buildStateBase(lastStateOfTrail);
for(SlicesAffineSpaceIterator
i=affineSpace.getIterator(); […]) {
   Trail newTrail(trail);
   newTrail.append(*i, LC.getWeight(*i));
   newTrail.save(outputFile);
}
Trail::produceHumanReadableFile(LC, outputFileName);
```

Backward propagation

Example: extending a differential trail

```
Trail trail(inputFile);
const vector<SliceValue>& firstStateOfTrail =
trail.states.front();
DC.reverseLambda(firstStateOfTrail, […]);
for(i=DC.getReverseStateIterator(λ⁻¹(firstStateOfTrail));
[…]) {
   Trail newTrail(trail);
   newTrail.prepend(*i, DC.getWeight(*i));
   newTrail.save(outputFile);
}
Trail::produceHumanReadableFile(DC, outputFileName);
```

Differential trail equations

Constructing equations to follow a trail

```
KeccakFDCEquations f(50);
Trail trail(inputFile);
f.genDCEquations(outputFile, trail);
```

Example of generated equations

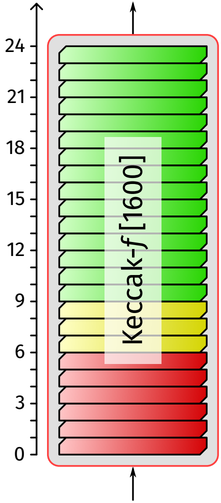
```
Ake1 = 0, Ako1 + 1 = 0, Ami1 = 0, […]
Bba0 = Aba0 + […]
Bbe0 + 1 = 0, Bbi0 + Bbo0 + 1 = 0, Bbu0 = 0, […]
Cba0 = Bba0 + […]
…
```

Outline

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

Status of KECCAK



- 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]
  - 9 rounds (2^{256} time, academic)
    [Dinur et al., EUROCRYPT 2015]
- Lots of third-party cryptanalysis available at:
  `https://keccak.team/third_party.html`

Any questions?

Thanks for your attention!

`https://keccak.team/`