Hashing and sponge functions
Part 2: What we built and how we did it

Joan Daemen[1]

Joint work with
Guido BERTONI[1], Michaël PEETERS[2] and Gilles Van Assche[1]

[1]STMicroelectronics [2]NXP Semiconductors

NISNet Winter School,
Finse
May 26, 2011

Outline

Outline

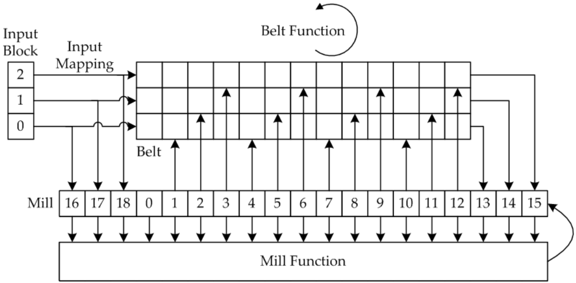
The early years

- SUBTERRANEAN: Daemen (1991)
  - hashing mode: Subhash
  - stream cipher mode: Substream
  - permutation-based, hardware oriented
- STEPRIGHTUP: Daemen (1994)
  - hashing and streaming modes
  - permutation-based, software oriented
- PANAMA: Daemen and Clapp (1998)
  - improved version of STEPRIGHTUP
  - stream cipher mode unbroken till today
  - hash mode broken in 2002 by Rijmen et al.

From Panama to RadioGatún

- Initiative to design hash/stream function (late 2005)
  - rumours about NIST call for hash functions
  - forming of Keccak Team
  - adopting the principles underlying Panama
- RadioGatún(2006)
  - more conservative than Panama
  - belt-and-mill structure
  - variable-length output
  - expressing security claim non-trivial exercise
- Sponge functions (early 2007)
  - solution to security claim expression

From RadioGatún to Keccak



- RadioGatún confidence crisis (2007-2008)
  - experiments did not inspire confidence in RadioGatún
  - follow-up design Gnoblio went nowhere
  - NIST SHA-3 deadline approaching ...
  - U-turn: design a sponge with strong permutation f
- Keccak (2008)

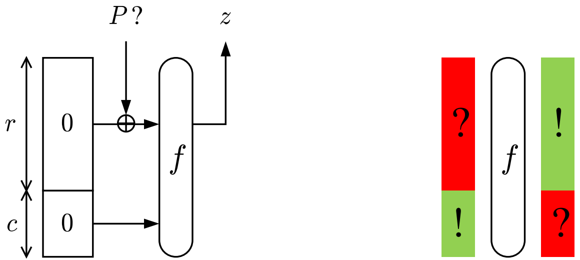
Outline

Desired properties of f

- Efficiency and flexibility
  - fast and compact, straight and hardened
  - ...on a wide range of CPU platforms and in hardware
- Classical LC/DC criteria
  - absence of large differential propagation probabilities
  - absence of large input-output correlations
- infeasibility of the CICO problem
- Immunity to
  - integral cryptanalysis
  - algebraic attacks
  - slide and symmetry-exploiting attacks
  - ...

The CICO problem

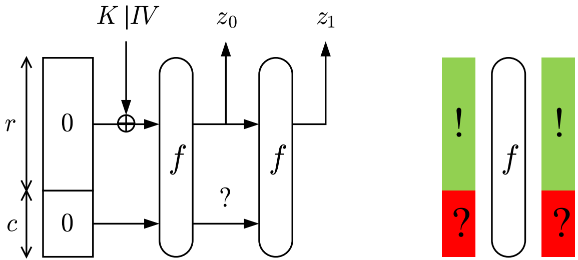
- Given partial input and output, determine remaining parts
- Important in many attacks



Pre-image generation in hashing

The CICO problem

- Given partial input and output, determine remaining parts
- Important in many attacks



State recovery in stream encryption

Goal: prevent control over difference propagation

- Differential (A, B) is composed of trails Q from A to B:

$$\#\text{pairs}(A, B) = \sum_{Q \in (A,B)} \#\text{pairs}(Q)$$

- $\mathrm{w_r}(Q)$: number of conditions Q imposes on its pairs:

$$\mathrm{w_r}(Q) = \sum_{\text{active S-boxes}} \mathrm{w_r}(q_i, q_o)$$

- If $\mathrm{w_r}(Q) < b : \#\text{pairs}(Q) \approx 2^{b - \mathrm{w_r}(Q)}$, else few or no pairs
- Ambition is to assure:
  - $\forall Q : \mathrm{w_r}(Q) > b$: wide trail strategy
  - absence of systematic clustering of trails

Goal: avoid large input-output correlations

- Correlation (v, u) is composed of trails Q to u from v

$$C(v, u) = \sum_{Q \in (v,u)} C(Q)$$

- Correlation contribution: $C(Q) = (-1)^{\mathrm{sign}(Q)} 2^{-\mathrm{w_c}(Q)/2}$ with

$$\mathrm{w_c}(Q) = \sum_{\text{active S-boxes}} \mathrm{w_c}(q_i, q_o)$$

- If $\mathrm{w_c}(Q) > b$, Q contributes very little
- Ambition is to assure:
  - $\forall Q : \mathrm{w_c}(Q) > b$: wide trail strategy
  - absence of systematic clustering of trails

Outline

Designing the permutation f

- Required width b:
  - long term: *security strength* up to 256 bits
  - capacity up to 512 bits
  - rate: $b - 512$ bits
  - width ranges from 600 to 2400 bits
- Like a block cipher
  - sequence of identical rounds
  - round function that is nonlinear and has good diffusion
- ...but not quite
  - no need for key schedule
  - round constants instead of round keys
  - inverse permutation need not be efficient

The obvious choices

- ARX
  - appears very powerful, but …
  - unsuited for dedicated hardware and DPA protection
  - hard to evaluate strength
  - all of the MD4 and SHA family is already based on ARX
- Square-inspired, like Rijndael (AES)
  - S-box with optimum worst-case LC and DC properties
  - mixing layer with optimum worst-case diffusion: MDS
  - transposition layer with optimum dispersion
  - results in strong bounds for trail weights
  - let's try it!

AES-based approach: size parameters

- AES structure must be scaled up from 128 to 600-2400 bits
- Three size parameters:
  - S-box width: n bits
  - MDS width: m S-boxes
  - Dimension: d
- Permutation width: $b = m^d n$
- AES: $n = 8$, $m = 4$, $d = 2$

Scaling up AES structure

- Increase S-box width n?
  - software: # elements in lookup tables: 2^n
  - hardware: strong increase in # gates
  - decreasing S-box width would be a better idea ...
- Increase MDS matrix size m?
  - SW with T-tables: size of elements is nm
  - HW and compact SW: strong increase in # operations/gates
- Increase the dimension d?
  - slows down diffusion
  - strong increase in number of rounds
- All in all, scaling up appears very expensive

A *greedy* aspect in AES-inspired design

- Choice of nonlinear layer
  - for given width n, choose S-boxes
  - with optimum worst-case nonlinearity
  - irrespective of implementation cost
- Choice of mixing layer
  - for given size m, choose mixing transformation
  - with optimum worst-case diffusion
  - irrespective of implementation cost
- Focus on worst-case LC/DC propagation over few rounds
- Excellent choice for T-table based implementations
- Can be costly in other types of implementations

Stick with the less greedy approach

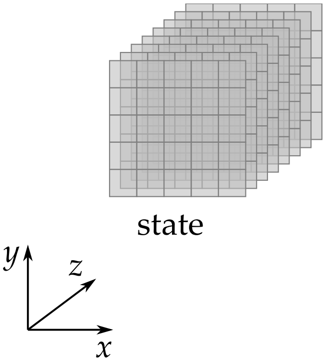
- A modest nonlinear layer
  - no requirement for high worst-case nonlinearity
  - $-\log(\mathrm{DP}(a, b)) \approx \mathrm{O}(HW(a))$
  - $-\log(C^2(v, u)) \approx \mathrm{O}(HW(u))$
- A modest mixing layer
  - no requirement for high worst-case diffusion
  - average diffusion preferably high
- A matching transposition layer should prevent
  - chaining of low-weight structures into narrow trails
  - clustering of trails
- Ambition: cheaper round function, more rounds but globally more efficient

Outline

KECCAK

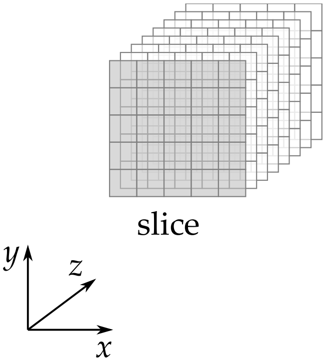
- Instantiation of a sponge function
  - variable-length input and output
  - 10^*1 padding
- KECCAK uses a permutation KECCAK-f
  - 7 permutations: $b \in \{25, 50, 100, 200, 400, 800, 1600\}$
- Security-speed trade-offs using the same permutation
- All values c and r with $c + r = b$ supported
- Examples
  - SHA-3: $r = 1024$ and $c = 576$ for $2^{c/2} = 2^{288}$ security
  - lightweight: $r = 40$ and $c = 160$ for $2^{c/2} = 2^{80}$ security

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



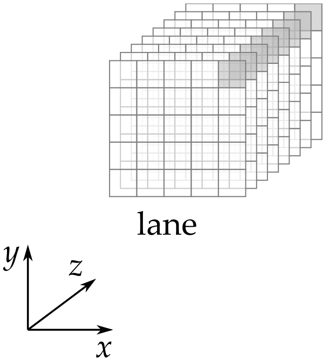
- 5×5 lanes, each containing 2^ℓ bits (1, 2, 4, 8, 16, 32 or 64)
- (5×5)-bit slices, 2^ℓ of them
- 7 widths b

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



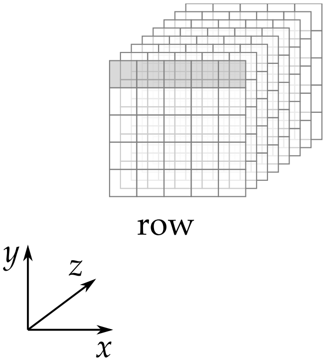
- 5×5 lanes, each containing 2^{ℓ} bits (1, 2, 4, 8, 16, 32 or 64)
- (5×5)-bit slices, 2^{ℓ} of them
- 7 widths b

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



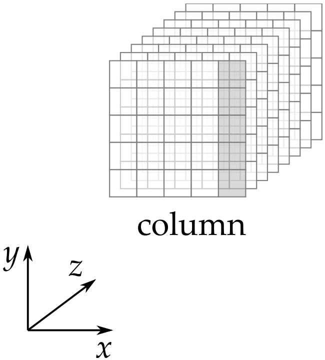
- 5×5 lanes, each containing 2^ℓ bits (1, 2, 4, 8, 16, 32 or 64)
- (5×5)-bit slices, 2^ℓ of them
- 7 widths b

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



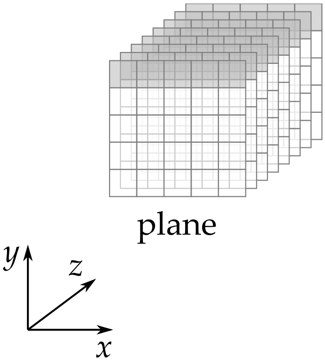
- 5×5 lanes, each containing 2^ℓ bits (1, 2, 4, 8, 16, 32 or 64)
- (5×5)-bit slices, 2^ℓ of them
- 7 widths b

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



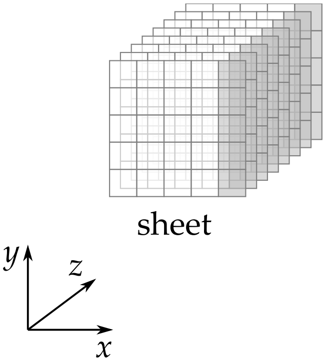
- 5×5 lanes, each containing 2^{ℓ} bits (1, 2, 4, 8, 16, 32 or 64)
- (5×5)-bit slices, 2^{ℓ} of them
- 7 widths b

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



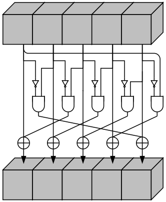
- 5×5 lanes, each containing 2^{ℓ} bits (1, 2, 4, 8, 16, 32 or 64)
- (5×5)-bit slices, 2^{ℓ} of them
- 7 widths b

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
- 7 widths b

χ, the nonlinear mapping in KECCAK-f

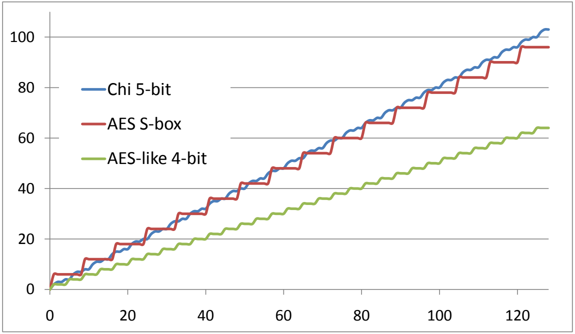


- "Flip bit if neighbors exhibit 01 pattern"
- Operates independently and in parallel on 5-bit rows
- Small number of operations per bit
- Algebraic degree 2, inverse has degree 3
- LC/DC propagation properties easy to describe and analyze

Comparing χ with AES S-box

Particular criterion:

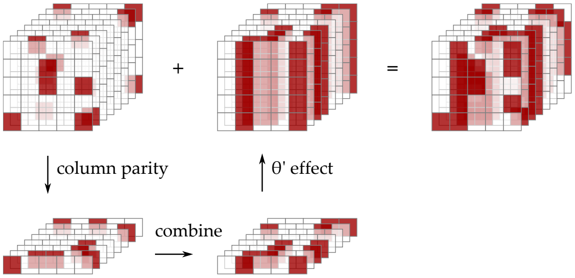
- X-axis: Hamming Weight HW(a)
- Y-axis: given HW(a), minimum weight: $\log_2(1/DC(a, b))$



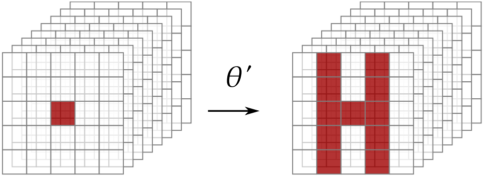
θ', a mixing layer

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

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

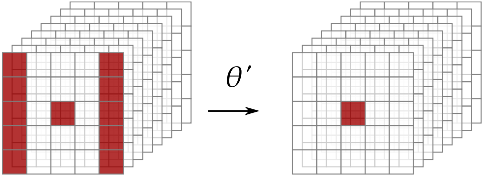


Diffusion of θ'



- θ' is linear:
  - $B = \theta'(A)$
  - $v^{\mathsf{T}} a = u^{\mathsf{T}} b$ with $v = \theta'^{\mathsf{T}}(u)$
- Good diffusion?
  - input bit propagates to eleven output bits
  - output bit depends on eleven input bits

Inverse of θ'



- Similar to θ' itself
  - bit at output propagates to eleven bits at input
  - input bit depends on eleven output bits

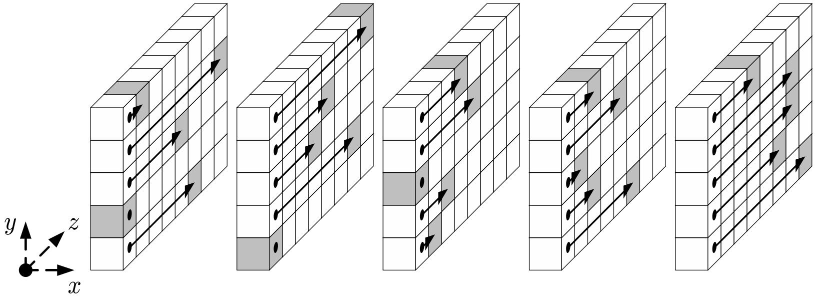
ρ for inter-slice dispersion

- Motivation:
  - χ makes bits within rows interact
  - θ linearly mixes between rows in a slice
  - we need diffusion between the slices …
- ρ: cyclic shifts of lanes with offsets:

$$\text{for } 0 \leq i < 25 : i(i+1)/2 \bmod 2^{\ell}$$

- Offsets cycle through all values below 2^{ℓ}

ρ In KECCAK-f



- Lanes are translated (cyclically) by different amounts
- Moves bits of a slice to different slices
- Translation-invariant in the direction of the z-axis

An initial attempt at KECCAK-f

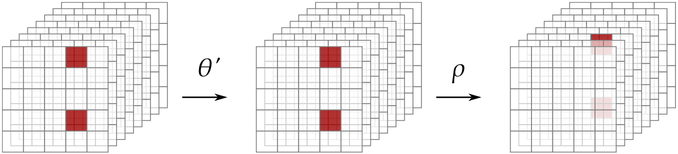
- Round function: $\mathrm{R} = \rho \circ \theta' \circ \chi$
- Repeat R until all trails have sufficient weight
- But ...
  - all-0 state is a fixed point of R
  - all-1 state too
- In general:
  - let α be a fixed point of $\theta' \circ \chi$
  - then the state value with all slices $= \alpha$ is a fixed point
- Problem: symmetry

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
  - advantage in analysis: *Matryoshka* structure
- Without ι, all rounds would be the same
  - susceptibility to *slide* attacks
  - defective cycle structure

Another attempt at KECCAK-f

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

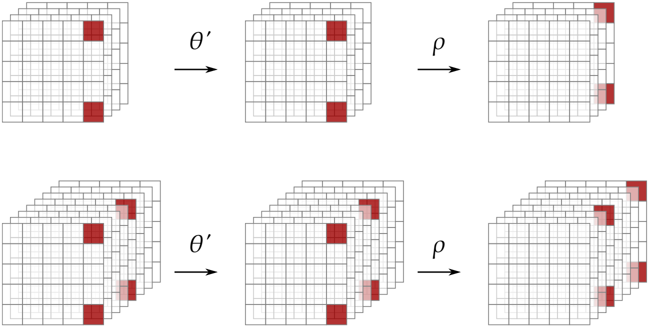


  - χ: may propagate unchanged
  - θ': propagates unchanged, because all column parities are 0
  - ρ: in general moves active bits to different slices ...
  - ...but not always

The cause of this problem

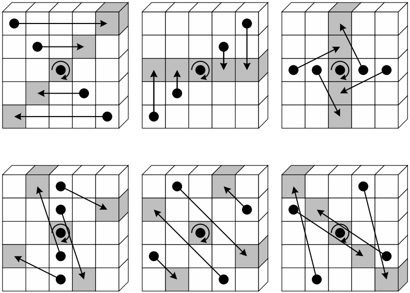
- Weak worst-case diffusion in θ'
  - two-bit difference/mask within column remains as is
  - (column-parity) kernel: subset of states with all $c_{x,z} = 0$
  - state values in kernel are invariant under θ'
- Weak worst-case dispersion of ρ
  - ρ should move bits in a column to 5 different columns
  - this is *impossible* for lane size 4 and smaller
- Affects security of KECCAK-$f[b]$ with $b \in \{25, 50, 100\}$
- Why bother?

The Matryoshka property



- Structure Q for $w = 2^{\ell}$ implies symmetric Q' for $w = 2^{\ell+n}$
- Patterns in Q' are z-periodic versions of patterns in Q
- Weight of trail Q' is 2^n times that of trail Q

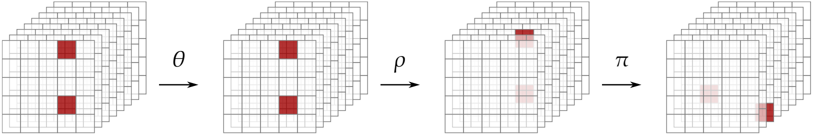
π for disturbing horizontal/vertical alignment



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

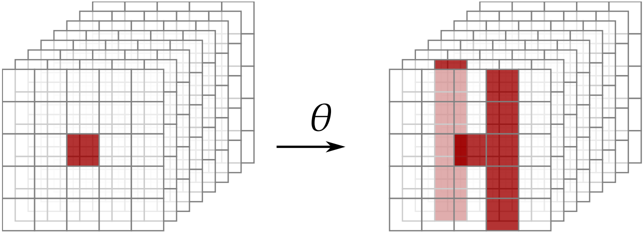
Yet another attempt at KECCAK-f

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



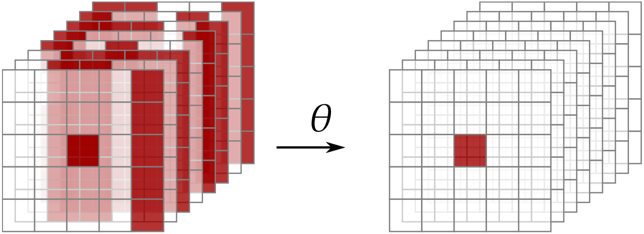
- π moves bits in same column to different columns!
- One more change though: tweaking θ'

Tweaking θ' to θ



- Add to $a_{x,y,z}$ column parities $c_{x-1,z}$ and $c_{x+1,z-1}$
- Diffusion from single-bit input similar to that of θ'
- but ...

Inverse of θ



- Diffusion from single-bit output to input very high
- Output leading to low-weight input implies specific parity
- Increases resistance against LC/DC and algebraic attacks

KECCAK-*f* summary

- Round function:

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

- Number of rounds: $12 + 2\ell$
  - KECCAK-$f[25]$ has 12 rounds
  - KECCAK-$f[1600]$ has 24 rounds
- Efficiency
  - high level of parallellism
  - flexibility: bit-interleaving
  - software: competitive on wide range of CPU
  - dedicated hardware: very competitive
  - Suited for DPA protection

KECCAK-f propagation properties summary

- χ: propagation weight $\approx$ Hamming weight
- θ: high diffusion except for low-weight in-kernel patterns
- π and ρ: drag those patterns out of the kernel
  - ...for trails over 4 rounds or more
  - not for 3 rounds: *kernel vortices*
- Additional benefit: *weak alignment*
  - no significant trail clustering
  - no truncated trails exploitable in rebound attacks
- Algebraic attacks: low degree of round function
  - marginal theoretical distinguishers: zero-sum
  - no impact on security claim

Outline

KECCAK resources

- KECCAK documentation, a.o.:
  - KECCAK reference
  - KECCAK implementation overview
  - Cryptographic sponge functions
- KECCAKTOOLS: set of documented C++ classes supporting:
  - individual steps θ, ρ, π, χ and ι
  - and their **inverses** $\iota^{-1} = \iota$, χ^{-1}, π^{-1}, ρ^{-1} and θ^{-1}
  - equations in GF(2) of rounds or steps
  - trail propagation for DC and LC ,including base + offset
- All freely available on `http://keccak.noekeon.org`

Questions?

Thanks for your attention!

Q?

More information on
http://keccak.noekeon.org/
http://sponge.noekeon.org/