Radboud University

Innovations in permutation-based crypto

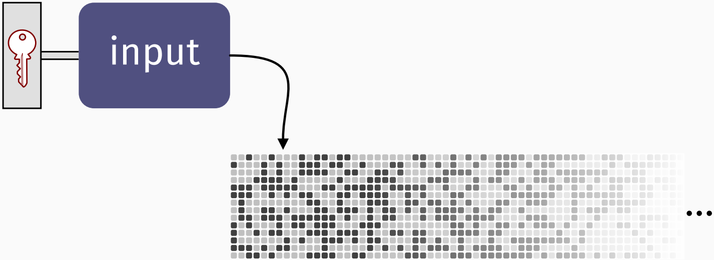
Joan Daemen[1]

based on joint work with Guido Bertoni[3], Seth Hoffert, Bart Mennink[1], Michaël Peeters[2], Gilles Van Assche[2] and Ronny Van Keer[2]

COINS Winter School, 5-10 May 2019, Finse

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

The simplest possible keyed cryptographic function



$$Z \leftarrow F_K(m, \ell)$$

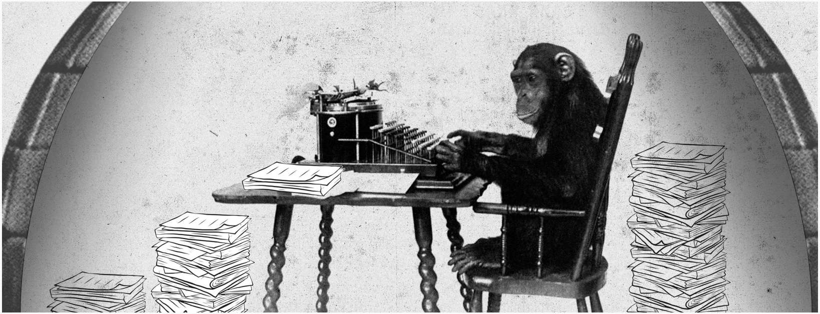
The ideal cryptographic function

- What would the ideal cryptographic function look like?
- It is called a Random Oracle ($\mathcal{RO}$) [Bellare-Rogaway 1993]
- Random Oracle can be built but is not practical

Random Oracle Inc.: letter answering service!

1. Letter with (m, ℓ) arrives at Random Oracle Inc.

2. Manager checks archive for presence of a file (m, Z)



3a. If no (m, Z) in archive, employee generates random Z with $|Z| = \ell$

3b. Else if $|Z| < \ell$, employee extends Z to length ℓ with random string

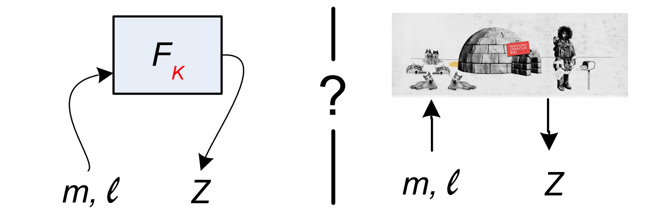
4. Manager copies Z

4. Manager puts file with (m, Z) (back) in archive

Random Oracle Inc.

5. Manager sends response Z truncated to length ℓ to sender

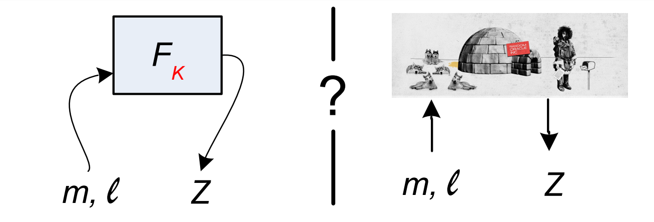
Security notion as a distinguishing game



Distinguishing game for an *Adversary* $\mathcal{A}$:

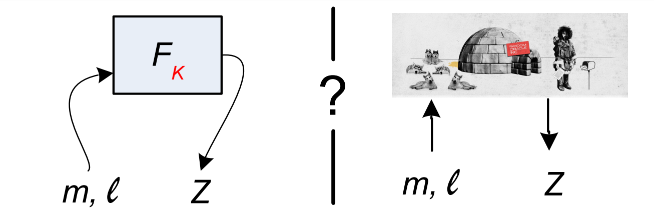
- $\mathcal{A}$ faces system S that it can query (m, ℓ) to get Z
- but does not know the world it lives in
  - in real world $S = F_K$
  - in ideal world $S = \mathcal{RO}$
- in both worlds, $\mathcal{A}$ has the specifications of F
- $\mathcal{A}$ can make queries and do *computations*
- and should guess the world it is in

Security notion: PRF security



- F is PRF-secure if $\Pr(\text{success})$ is $1/2 + \epsilon$ with ϵ negligible
  - for any reasonable amount of queries and computation
  - we call 2ϵ the *($\mathcal{RO}$ distinguishing) advantage* Adv
- Quantifying effort of adversary $\mathcal{A}$
  - *online* complexity M: sum of data $|m| + \ell$ over all queries
  - *offline* complexity N: computational effort (per some unit)
- PRF security of F is a bound on Adv as $f(M, N)$
- Implication: for any attack $\Pr(\text{succ.}|F) \leq f(M, N) + \Pr(\text{succ.}|\mathcal{RO})$

What can we do with (PRF) security bounds?

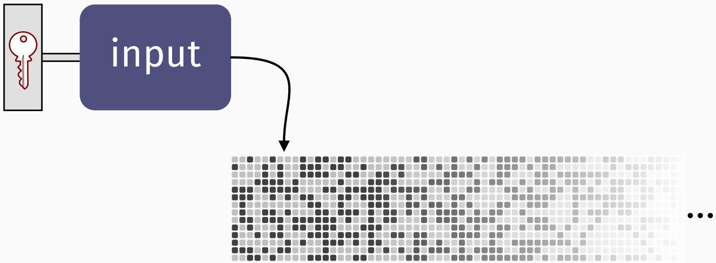


- We cannot prove a bound for any concrete F
- But we can formulate one and use in a security claim for F
  - statement on expected security
  - made by the designers (or standardization organization)
- Claim serves as challenge for cryptanalysts
  - break: distinguishing attack with $\mathsf{Adv} > f(M, N)$
- Claim serves as security specification for user
  - ...as long as it is not broken
- Assurance grows as years and public scrutiny pile up

Back to our cryptographic function F

What can we do with a concrete F?

Say we have:

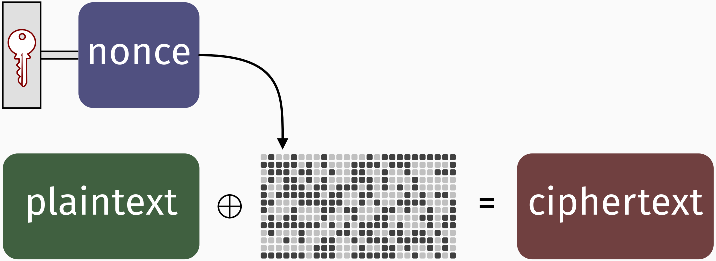


with claim, say,

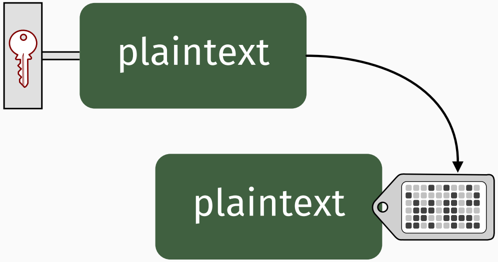
$$\mathrm{Adv} \leq \frac{N}{2^{256}} + \frac{M^2}{2^{256}}$$

for K chosen uniformly from space of 256-bit keys

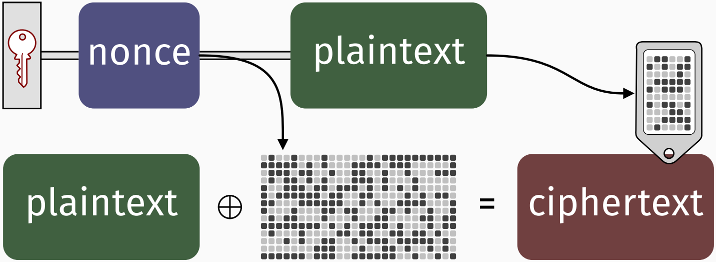
Stream encryption



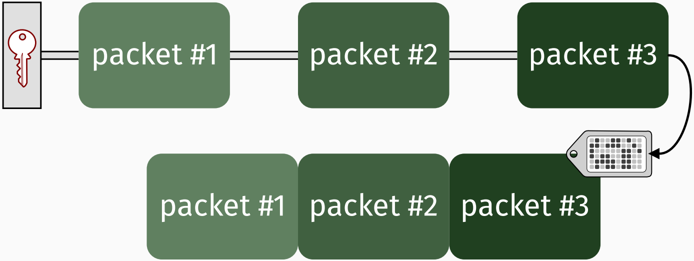
Message authentication (MAC)



Authenticated encryption



String sequence input and incrementality

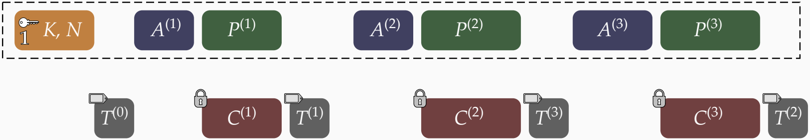


$$F_K\left(P^{(3)} \circ P^{(2)} \circ P^{(1)}\right)$$

We call this: *doubly-extendable cryptographic keyed function*

deck function

Session authenticated encryption (SAE) [KT, SAC 2011]



Initialization taking nonce N

$T \leftarrow 0^t + F_K(N)$

$\text{history} \leftarrow N$

return tag T of length t

Wrap taking metadata A and plaintext P

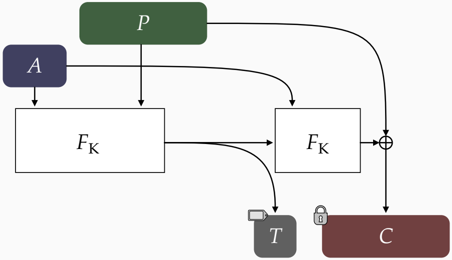
$C \leftarrow P + F_K(A \circ \text{history})$

$T \leftarrow 0^t + F_K(C \circ A \circ \text{history})$

$\text{history} \leftarrow C \circ A \circ \text{history}$

return ciphertext C of length $|P|$ and tag T of length t

Synthetic initialization value (SIV) of [KT, eprint 2016/1188]



Unwrap taking metadata A, ciphertext C and tag T

$P \leftarrow C + F_K(T \circ A)$

$\tau \leftarrow 0^t + F_K(P \circ A)$

if $\tau \neq T$ **then return** error!

else return plaintext P of length $|C|$

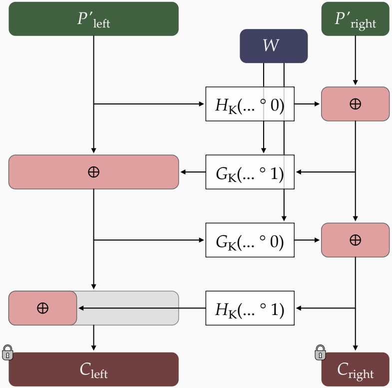
Variant of SIV of [Rogaway & Shrimpton, EC 2006]

Wide block cipher (WBC), as in [KT, eprint 2016/1188]

Encipher P with K and tweak W

$$
\begin{array}{llll}
(L, R) & \leftarrow & & \mathrm{split}(P) \\
R_0 & \leftarrow R_0 & + & H_K(L \circ 0) \\
L & \leftarrow L & + & G_K(R \circ W \circ 1) \\
R & \leftarrow R & + & G_K(L \circ W \circ 0) \\
L_0 & \leftarrow L_0 & + & H_K(R \circ 1) \\
C & \leftarrow L & \| & R
\end{array}
$$

return ciphertext C of length $|P|$

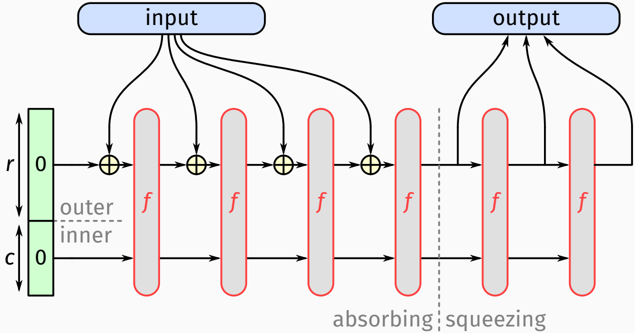


Inspired by HHFHFH of [Bernstein, Nandi & Sarkar, Dagstuhl 2016]

How to build a deck function?

By icelight (flickr.com)

Sponge [Keccak Team, Ecrypt 2008]

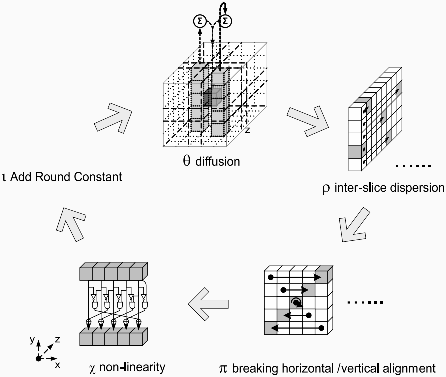


- Uses b-bit permutation, has rate r and capacity c with $b = r + c$
- Taking K as first part of input gives a deck function (almost)
- We can prove Adv $< \frac{M^2}{2^{c+1}} + \frac{N}{2^{|K|}}$ if f and K are randomly chosen
- So sponge construction is sound but f must still be built

Intermezzo: how to build a suitable permutation f?

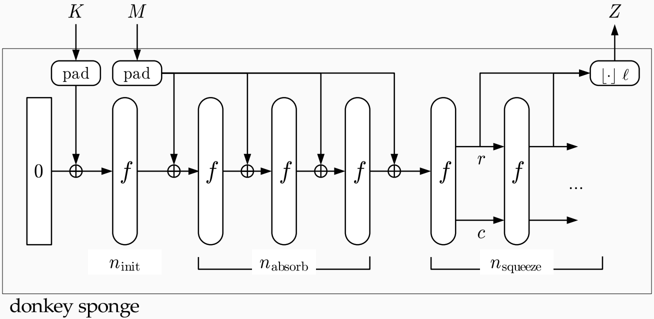
- Same as a block cipher (e.g. AES):
  - design an efficient round function and repeat that
  - resistance to attacks grows (hopefully fast) with # rounds
  - determine # rounds that is broken and take some more
- Steps of a good round function:
  - nonlinear step: combines nearby bits non-linearly
  - mixing layer: combines nearby bits linearly
  - transposition layer: moves nearby bits far away
- Difference with block ciphers
  - no key schedule nor round keys but instead round constants
  - no need for efficient inverse

For example: KECCAK-$p[1600, n_r]$

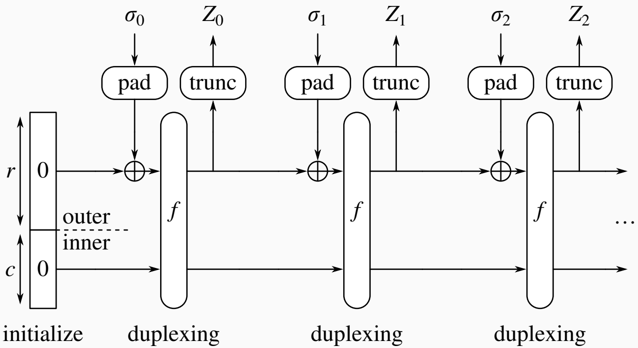


- Bit-oriented round function with high amount of symmetry:
  - software with cyclic shift and Boolean instructions only
  - fast and compact in hardware
- Non-linear step χ: algebraic degree 2
- Lightweight round function with heavy inverse

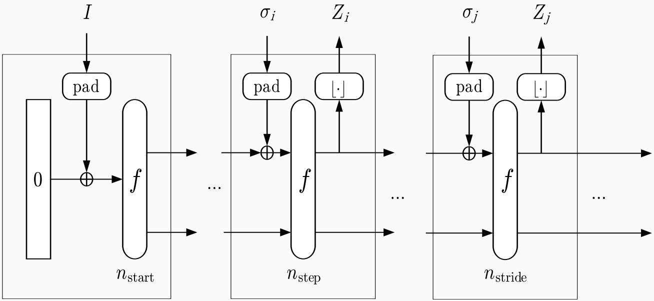
Speed up absorbing: sponge → donkeySponge [KT, DIAC 2012]



Incrementality: duplex [Keccak Team, SAC 2011]



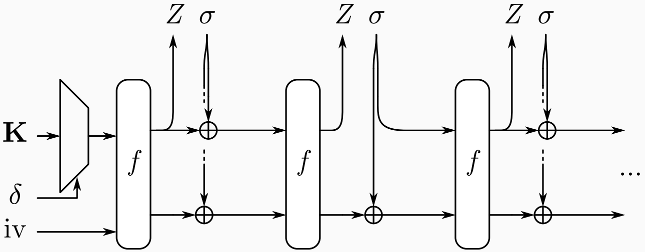
Speed up: duplex → monkeyDuplex [KT, DIAC 2012]



Very popular:

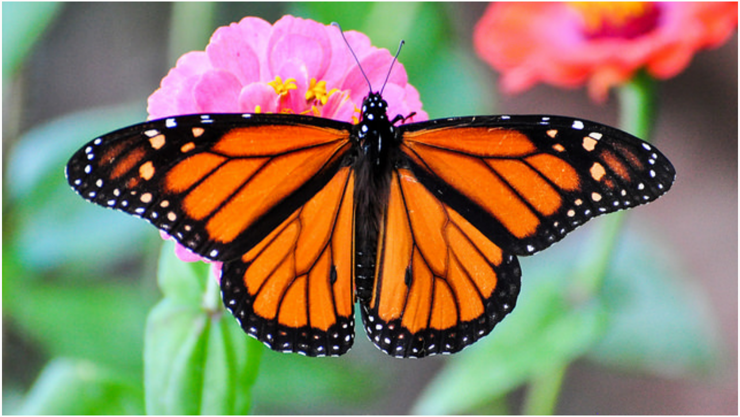
- Adopted by half a dozen CAESAR submissions
- including our proposal KETJE [KT, CAESAR 2014]

Consolidation: Full-state keyed duplex



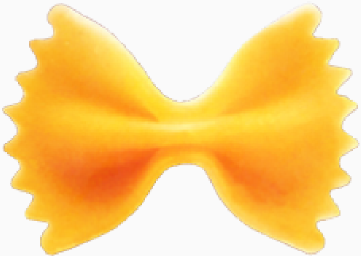
[Mennink, Reyhanitabar, & Vizar, Asiacrypt 2015]

[Daemen, Mennink & Van Assche, Asiacrypt 2017]



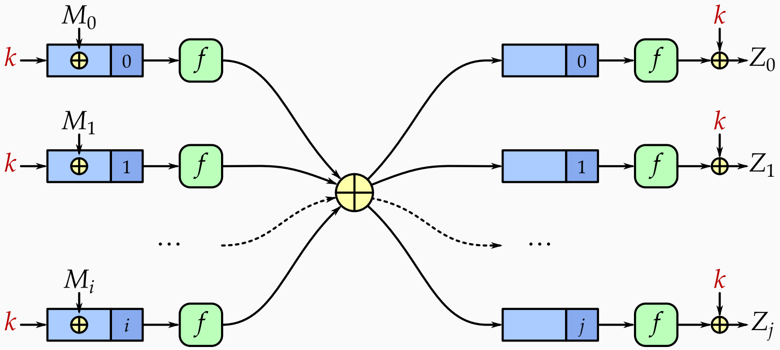
by Peter Miller (flick.com)

How to build a parallelizable deck function?



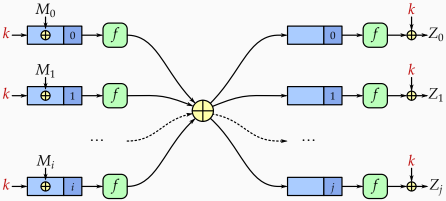
by Barilla Food Service

Farfalle: early attempt [KT 2014-2016]



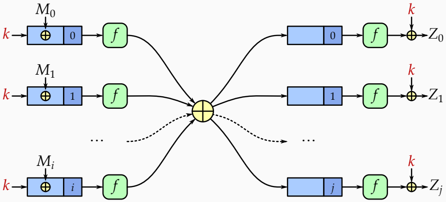
- Reminds of Protected Counter Sums [Bernstein, "stretch", JOC 1999]
- In Protected Counter Sums, f is assumed to be a PRF
- We had in mind for f: Keccak-$p[1600, n_r]$ with few rounds

Problem of early Farfalle: higher-order-differential collisions



- Differential $\Delta_v f(x) = f(x+v) + f(x)$ is kind of *derivative* of f
- The algebraic degree of $\Delta_v f$ is at most that of f minus one
- Derive $\Delta_v f$ in turn:
  $\Delta_u \Delta_v f(x) = f(x+v+u) + f(x+u) + f(x+v) + f(x)$
- d-th derivative is $\Delta_V f(x) = \sum_{v \in V} f(x+v)$ with V a vector space
- Degree of n_r-round KECCAK-$p[1600, n_\mathrm{r}]$: 2^{n_r}
- if $\dim(V) = 2^{n_\mathrm{r}}$: $\Delta_V f(x)$ is a constant

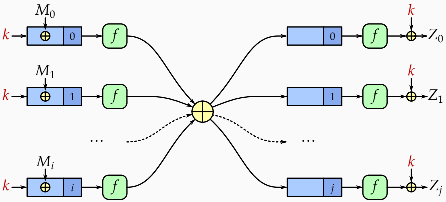
Problem of early Farfalle: higher-order-differential collisions (cont'd)



Collision-generating attack:

- Choose a message m of 2^{n_r} blocks m_i that form a vector space
- Encodings of block numbers also form a vector space
- Inputs to f also form a vector space
- Accumulator is a constant independent of m or k
- n-fold multicollision with online cost $M = n2^{2^{n_r}}$ input blocks
- With carefully chosen blocks m_i this reduces to $M = n2^{2^{n_r-1}}$
- Practical up to $n_r = 6$: each such message is *only* 0.5 Terabyte

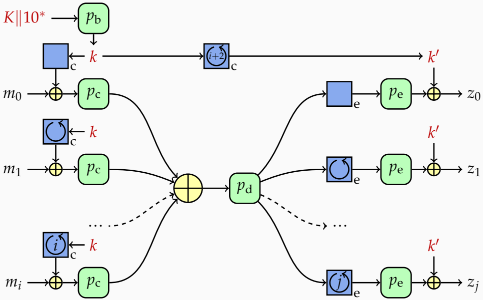
Higher-order-differential collisions: attempts at mitigation



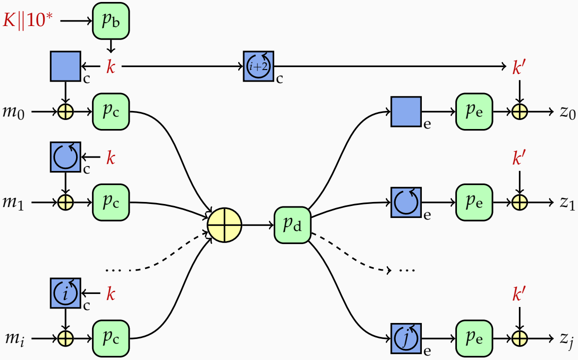
Fancy encoding $\mathsf{enc}(i)$ of block numbers

- To fully prevent high-dimensional affine spaces at f input
- We tried many things …
- Nicest one: $\mathsf{enc}(i) = x^i \bmod p(x) \| x^{-i} \bmod p(x)$
  - with $p(x)$ an primitive polynomial
  - computing $\mathsf{enc}(i+1)$ from $\mathsf{enc}(i)$ takes two LFSR updates
  - No affine spaces exist with dimension > 2 for same reason that AES S-box has differential uniformity 4

- Not to prevent affine spaces but just to make them *hard to find*
- Computing of f-input for block i: $m_i + (x^i k) \bmod p(x)$
- We call this *input mask rolling*
  - k is full-width secret *mask* derived from user key K
  - If $p(x)$ not sparse, choosing m_i to form exploitable affine space at input to f is infeasible
- Additional benefit: increases rate of blocks m_i to full-width

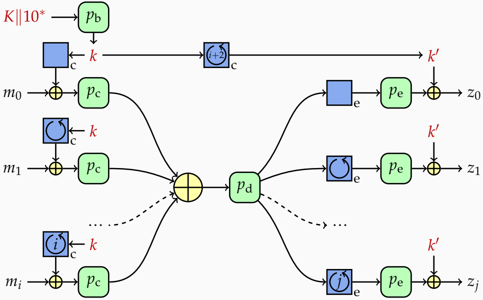


Farfalle: the final construction (modulo some details)



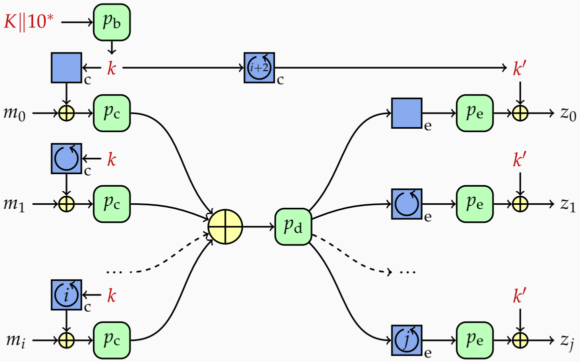
- Derivation of mask k from user key K using p_b
- Input mask rolling and p_c to prevent higher-order-differentials
- State rolling, p_e and mask against state retrieval at output
- Middle p_d against accumulator-affine-space attack
- Input-output attacks have to deal with $p_e \circ p_d \circ p_c$

Farfalle: accumulator-affine-space attack



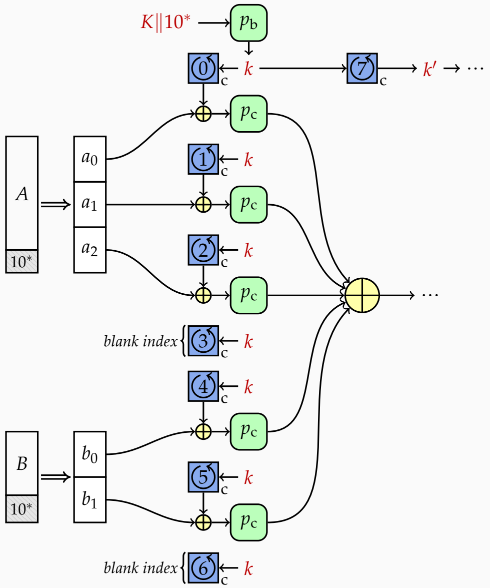
- Apply 2^2 2-block messages: $m_0\|m_1, m_0'\|m_1, m_0\|m_1', m_0'\|m_1'$
- Let $a_i = p_\mathrm{c}(m_i + k_i)$ and $a_i' = p_\mathrm{c}(m_i' + k_i)$
- Affine space in accumulator: $a_0 + a_1, a_0' + a_1, a_0 + a_1', a_0' + a_1'$
- Generalizes to dim. d taking 2^d messages of each d blocks
- Can distinguish construction if $p_\mathrm{e} \circ p_\mathrm{d}$ has too low degree
- This is what forced us to add p_d

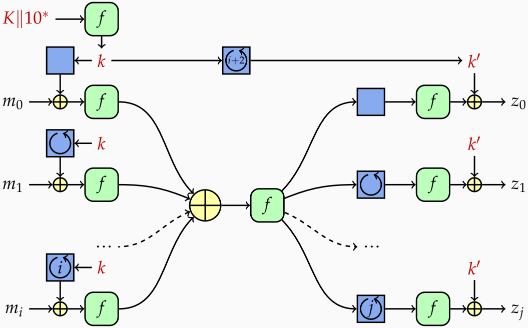
Farfalle now [Keccak Team + Seth Hoffert, ToSC 2018]



- LFSR input mask rolling and p_c against accumulator collisions
- Middle p_d against accumulator-affine-space attack
- NLFSR state rolling, p_e and mask against state retrieval at output
- Input-output attacks have to deal with $p_e \circ p_d \circ p_c$

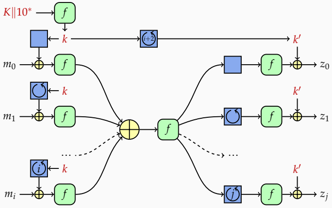
Multi-string input and incrementality



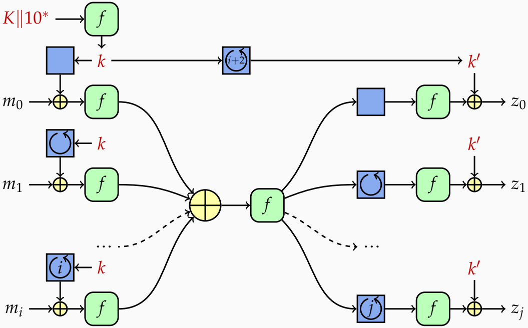


- $p_i = \text{KECCAK-}p[1600, n_r]$ with # rounds in p_b, p_c, p_d, p_e: 6644
- Input mask and state rolling with LFSR over 320 of 1600 bits
- With claim targeting 128-bit security

KRAVATTE IACR eprint July 2017 under pressure

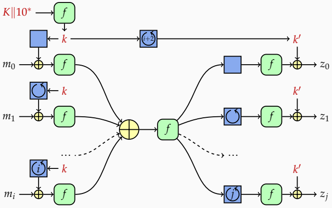


- October 2017: attacks by Colin Chaigneau, Thomas Fuhr, Henri Gilbert, Jian Guo, Jérémy Jean, Jean-René Reinhard and Ling Song
- Extension of accumulator-affine-space attack
  - degree of $p_e \circ p_d$ is 2^8, so infeasible?
  - peel off 1 round by guessing a few key bits, now degree 2^7
  - peel off 2 rounds with advanced techniques: break
- New attack: state recovery using output only
  - expansion is just nonlinearly filtered LFSR!
  - linearization and meet-in-the-middle techniques
  - massive complexities M and N but still a break

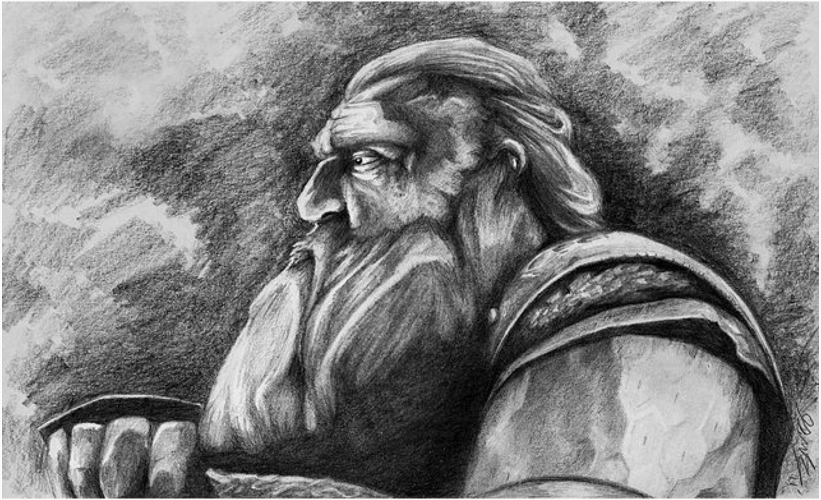


- Target security: still 128 bits
- p_i = KECCAK-$p[1600, n_r]$ with # rounds 6666 *Achouffe configuration*
- Input mask rolling with 320-bit LFSR
- State rolling with 640-bit NLFSR

Is KRAVATTE lightweight?

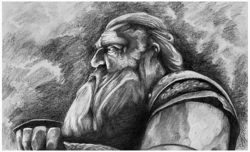


- Marginal cost per input or output block:
  - f execution + (N)LFSR update + mask addition
- In Farfalle: 6R KECCAK-p plays role similar to 4R AES
- Workload per round (in HW or bit-slice SW)
  - AES takes 20 operations per bit: 16 XOR and 4 AND
  - KECCAK-p takes 4 operations per bit: 3 XOR and 1 AND
- Workload per execution (in HW or bit-slice SW)
  - 4R AES: 10 ops/byte
  - 6R KECCAK-p: 3 ops/byte
- Disadvantage of KRAVATTE: 200-byte granularity



by Perrie Nicholas Smith (perriesmith.deviantart.com)

Gimli [Bernstein, Kölbl, Lucks, Massolino, Mendel, Nawaz, Schneider, Schwabe, Standaert, Todo, Viguier, CHES 2017]



- Ideal size and shape: 48 bytes in 12 words of 32 bits
  - compact on low-end: fits registers of ARM Cortex M3/M4
  - fast on high-end: suitable for SIMD
- For low-end platforms: locality of operations to limit swapping
  - limits diffusion, see e.g. [Mike Hamburg, 2017]
  - no problem for nominal number of rounds: 24
  - not clear how many rounds needed in Farfalle

Xoodoo · *[noun, mythical]* · /zu: du:/ · Alpine mammal that lives in compact herds, can survive avalanches and is appreciated for the wide trails it creates in the landscape. Despite its fluffy appearance it is very robust and does not get distracted by side channels.

XOODOO [Keccak team with Seth Hoffert]

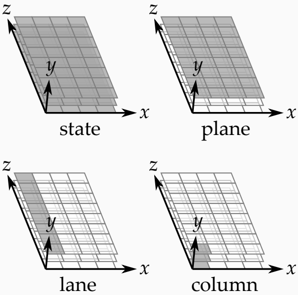
ref. code: `github.com/XoodooTeam/Xoodoo`

XOODOO cookbook: `eprint.iacr.org/2018/767`

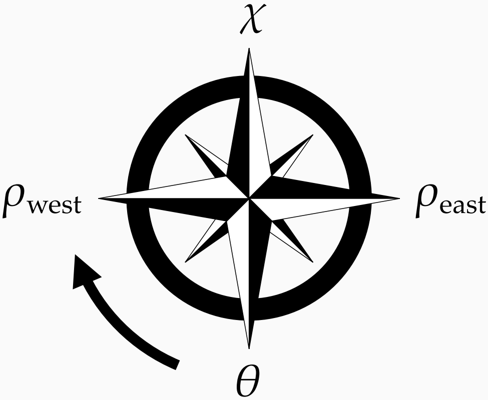
XOODOO and XOOFFF paper at TOSC 2018/4

- 384-bit permutation *KECCAK philosophy ported to Gimli shape*
- Main purpose: usage in Farfalle: XOOFFF
  - Achouffe configuration
  - efficient on wide range of platforms
- XOODOO cookbook also specifies:
  - XOOFFF-SANE: session AE relying on user nonce
  - XOOFFF-SANSE: session AE using SIV technique
  - XOOFFF-WBC: tweakable wide block cipher
  - XOODYAK: duplex object submitted to NIST lightweight competition

Xoodoo state



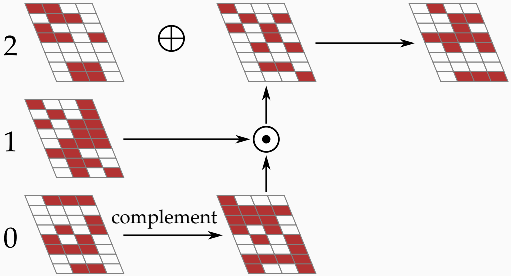
- State: 3 horizontal planes each consisting of 4 32-bit lanes



Iterated: n_{r} rounds that differ only by round constant

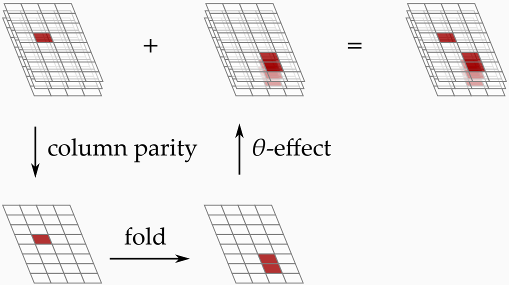
Nonlinear mapping χ

Effect on one plane:



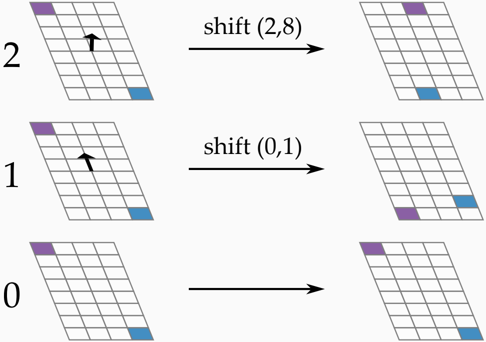
- χ as in KECCAK-p, operating on 3-bit columns
- Involution and same propagation differentially and linearly

Mixing layer θ



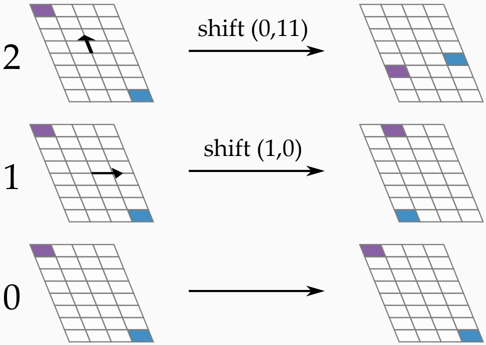
- Column parity mixer: compute parity, fold and add to state
- good average diffusion, identity for states in *kernel*
- heavy inverse

Plane shift ρ_{east}



- After χ and before θ
- Shifts planes $y = 1$ and $y = 2$ over different directions

Plane shift ρ_{west}



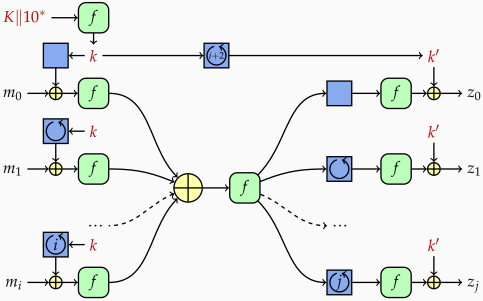
- After θ and before χ
- Shifts planes $y = 1$ and $y = 2$ over different directions

Xoodoo pseudocode

n_r rounds from $i = 1 - n_\mathrm{r}$ to 0, with a 5-step round function:

$$
\begin{aligned}
\theta :& \\
& P \leftarrow A_0 + A_1 + A_2 \\
& E \leftarrow P \lll (1,5) + P \lll (1,14) \\
& A_y \leftarrow A_y + E \text{ for } y \in \{0,1,2\} \\
\rho_\mathrm{west} :& \\
& A_1 \leftarrow A_1 \lll (1,0) \\
& A_2 \leftarrow A_2 \lll (0,11) \\
\iota :& \\
& A_{0,0} \leftarrow A_{0,0} + \mathrm{C}_i \\
\chi :& \\
& B_0 \leftarrow \overline{A_1} \cdot A_2 \\
& B_1 \leftarrow \overline{A_2} \cdot A_0 \\
& B_2 \leftarrow \overline{A_0} \cdot A_1 \\
& A_y \leftarrow A_y + B_y \text{ for } y \in \{0,1,2\} \\
\rho_\mathrm{east} :& \\
& A_1 \leftarrow A_1 \lll (0,1) \\
& A_2 \leftarrow A_2 \lll (2,8)
\end{aligned}
$$

Xoodoo + Farfalle = Xoofff



- $f = \textsc{Xoodoo}[6]$
- Input mask rolling with LFSR, state rolling with NLFSR
- Claimed PRF bound (simplified):

$$\frac{N}{2^{|K|}} + \frac{N}{2^{192}} + \frac{M}{2^{128}}$$

with $N = \#\, f$ executions and $M = \#$ 48-byte blocks

The input mask rolling function in XOOFFF

- State V as a 12-stage feedback shift register V at time t:

$$\begin{pmatrix} A_{0,2} & A_{1,2} & A_{2,2} & A_{3,2} \\ A_{0,1} & A_{1,1} & A_{2,1} & A_{3,1} \\ A_{0,0} & A_{1,0} & A_{2,0} & A_{3,0} \end{pmatrix} = \begin{pmatrix} V_{t+2} & V_{t+5} & V_{t+8} & V_{t+11} \\ V_{t+1} & V_{t+4} & V_{t+7} & V_{t+10} \\ V_{t} & V_{t+3} & V_{t+6} & V_{t+9} \end{pmatrix}$$

- Lightweight linear recursion:

$$V_{t+12} \leftarrow V_t + (V_t \ll 13) + (V_{t+1} \lll 3)$$

- Inspired by [Granger, Jovanovic, Mennink & Neves, EC 2016]
- Allows computing $V_{t+12}, V_{t+13}, \ldots, V_{t+22}$ in parallel
- Invertible, with minimal polynomial that is *primitive*:

$$\begin{aligned} &1 + x^{46} + x^{92} + x^{94} + x^{138} + x^{142} + x^{186} + x^{188} + x^{190} + x^{199} + x^{223} \\ &+ x^{238} + x^{245} + x^{247} + x^{269} + x^{271} + x^{284} + x^{286} + x^{295} + x^{319} + x^{330} \\ &+ x^{334} + x^{341} + x^{343} + x^{352} + x^{365} + x^{367} + x^{378} + x^{380} + x^{382} + x^{384} \end{aligned}$$

- We computed it with Berlekamp-Massey

The state rolling function in XOOFFF

- State V as a 12-stage feedback shift register V at time t:

$$\begin{pmatrix} A_{0,2} & A_{1,2} & A_{2,2} & A_{3,2} \\ A_{0,1} & A_{1,1} & A_{2,1} & A_{3,1} \\ A_{0,0} & A_{1,0} & A_{2,0} & A_{3,0} \end{pmatrix} = \begin{pmatrix} V_{t+2} & V_{t+5} & V_{t+8} & V_{t+11} \\ V_{t+1} & V_{t+4} & V_{t+7} & V_{t+10} \\ V_{t} & V_{t+3} & V_{t+6} & V_{t+9} \end{pmatrix}$$

- Lightweight non-linear recursion:

$$V_{t+12} \leftarrow (V_t \lll 5) + (V_{t+1} \cdot V_{t+2}) + (V_{t+1} \lll 13) + \texttt{00000007}$$

- Allows computing $V_{t+12}, V_{t+13}, \ldots, V_{t+21}$ in parallel
- Invertible and avoiding fixed points and (some) short cycles
- Main criterion for recursion formula: monomial count
  - bits of V_t are functions of $V_0, V_1, \ldots V_{11}$
  - for small t we can reconstruct the algebraic normal form
  - for large t we can sample the ANF
  - chosen recursion has high increase of $\#$ monomials in t

Xoofff performance

Xoofff		
mask derivation	1985	cycles
less than 48 bytes	5658	cycles
MAC computation use case:		
long inputs	26.0	cycles/byte
Stream encryption use case:		
long outputs	25.1	cycles/byte
AES-128 counter mode	121.4	cycles/byte

ARM Cortex-M0

Xoofff		
mask derivation	781	cycles
less than 48 bytes	2568	cycles
MAC computation use case:		
long inputs	8.8	cycles/byte
Stream encryption use case:		
long outputs	8.1	cycles/byte
AES-128 counter mode	33.2	cycles/byte

ARM Cortex-M3

Xoofff performance

Xoofff		
mask derivation	168	cycles
less than 48 bytes	504	cycles
MAC computation use case:		
long inputs	0.90	cycles/byte
Stream encryption use case:		
long outputs	0.94	cycles/byte
AES-128 counter mode	0.65	cycles/byte

Intel Core i5-6500 (Skylake), single core, Turbo Boost disabled
(256-bit SIMD)

Xoofff performance

Xoofff		
mask derivation	74	cycles
less than 48 bytes	358	cycles
MAC computation use case:		
long inputs	0.40	cycles/byte
Stream encryption use case:		
long outputs	0.51	cycles/byte
AES-128 counter mode	0.65	cycles/byte

Intel Core i7-7800X (SkylakeX), single core, Turbo Boost disabled
(512-bit SIMD)

How fast does a 1-bit difference spread in XOODOO?

- Dependency D_{av}: number of bits affected
- Hamming weight $\overline{w}_{\mathrm{av}}$: average Hamming weight of difference
- Bitwise entropy H_{av}: uncertainty about flipping of bits

	δ_a			δ_K			δ_b		
stage	D_{av}	$\overline{w}_{\mathrm{av}}$	H_{av}	D_{av}	$\overline{w}_{\mathrm{av}}$	H_{av}	D_{av}	$\overline{w}_{\mathrm{av}}$	H_{av}
a_{-2}	384	191.9	383.9	384	191.9	383.9	384	191.9	383.9
b_{-2}	381	187.6	357.4	384	189.7	376.8	384	191.9	383.9
a_{-1}	293	176.5	224.0	346	183.9	315.9	384	191.9	383.9
b_{-1}	3	2.0	2.0	6	3.9	4.0	279	168.5	220.9
$\mathbf{a_0}$	**1**	**1.0**	**0.0**	**2**	**2.0**	**0.0**	133	133.0	0.0
$\mathbf{b_0}$	7	7.0	0.000	**2**	**2.0**	**0.0**	**1**	**1.0**	**0.0**
a_1	21	14.0	14.0	6	3.9	4.000	3	1.9	2.0
b_1	102	64.4	75.0	42	28.0	28.0	21	13.9	14.0
a_2	210	94.7	187.2	105	48.4	87.7	63	28.003	50.7
b_2	371	181.0	366.1	293	140.4	268.1	207	94.9	182.9
a_3	384	188.5	382.6	357	164.8	343.2	321	128.6	293.3
b_3	384	191.9	383.9	384	191.9	383.9	384	188.0	381.6

Xoodoo differential propagation (and correlation)

- Security of Farfalle and sponge limited by $\max \mathrm{DP}(\Delta_a, \Delta_b)$
  - $\max \mathrm{DP}(\Delta_a, \Delta_b)$ by itself hard to determine
- For Xoodoo: $\max \mathrm{DP}(\Delta_a, \Delta_b) \approx \max_Q \mathrm{DP}(Q)$ with
  - Q a trail of difference patterns: $\Delta_0, \Delta_1, \Delta_2, \ldots \Delta_r$ and
  - $\mathrm{DP}(Q)$: probability that pair with input difference Δ_0 has difference Δ_j after round j
  - Trail weight $\mathrm{w}(Q)$ defined by $2^{-\mathrm{w}(Q)} = \mathrm{DP}(Q)$

Bounds on trail weights, using [Mella, Daemen, Van Assche, ToSC 2016]:

# rounds:	1	2	3	4	5	6
differential:	2	8	36	[74, 80]	≥ 90	≥ 104
linear:	2	8	36	[74, 80]	≥ 90	≥ 104

Conclusions

- Secure deck functions are very powerful primitives
  - stream encryption
  - MAC function
  - nonce-based (session) AE
  - SIV-based (session) AE
  - Wide block encryption
- Deck functions can be built from permutations
  - compact: (full-state) keyed duplex
  - parallel: farfalle
- Using XOODOO gives very competitive deck function XOOFFF

Thanks for your attention!