Introduction to sponge-based cryptography
Part 2: Keyed modes

Joan Daemen

STMicroelectronics and Radboud University

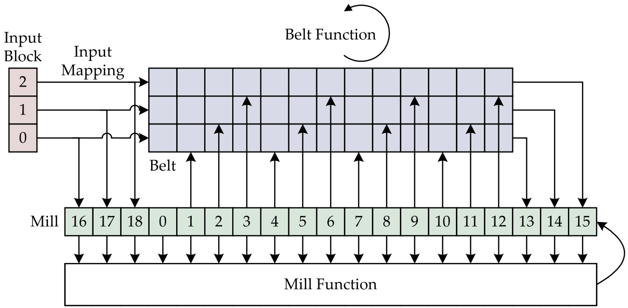
SPACE 2016

Hyderabad, India, December 14, 2016

Outline

Outline

RadioGatún [Keccak team, NIST 2nd hash workshop 2006]



- XOF: eXtendable Output Function
- Problem: **expressing security claim**
- Search for random oracle but then with inner collisions

(Early) Sponge at Dagstuhl, January 2007

Screenshot:

- Description:
  - Internal state $S = (S_A, S_G) \in Z_2 \times Z_2^c$ with initial value $S = (0,0)$
  - Absorbing: for each bit p of the input:

    $S = f(S_A + p,\ S_G)$
  - Resting:

    $S = f(S_A + 1,\ S_G)$
  - Squeezing: for each bit z of the output:

    $z = S_A$

    $S = f(S_A + 0,\ S_G)$
- We call c: the *sponge capacity*

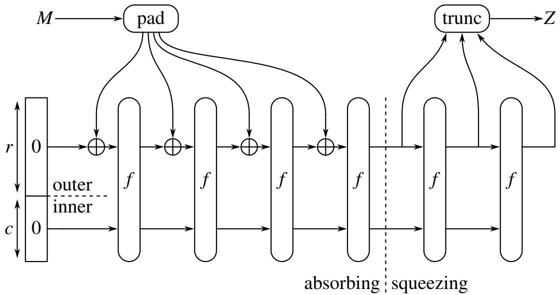
Generic security of Sponge [KT, Ecrypt hash, September 2007]

- Random sponges:
  - T-sponge: f is random transformation
  - P-sponge: f is random permutation
- Theorem: if no inner collisions, output is uniformly random
  - inner collision: different inputs leading to same inner state
  - Probability of inner collision:

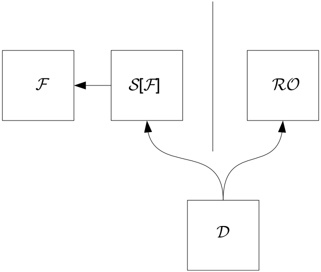
$$\frac{M^2}{2^{c+1}} \text{ with } M : \# \text{ calls to } f$$

Promoting sponge from reference to usage (2007-2008)

- RadioGatún cryptanalysis (1st & 3rd party): not promising
- NIST SHA-3 deadline approaching …U-turn
- Sponge with *strong* permutation *f*: Keccak [KT, SHA-3, 2008]

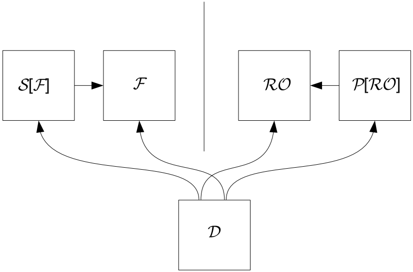


Distinguishing random sponge from random oracle



- Distinguishing advantage: $2^{-c-1}M^2$
- Problem: in real world, adversary has access to f

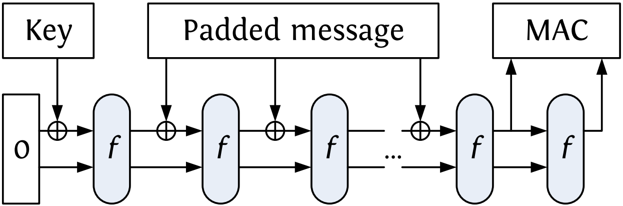
Differentiating random sponge from random oracle



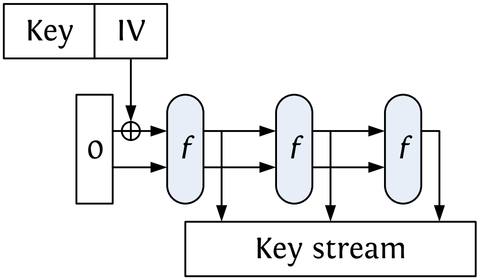
- Indifferentiability framework [Maurer, Renner & Holenstein, 2004]
- Applied to hashing [Coron, Dodis, Malinaud & Puniya, 2005]
- Random oracle augmented with *simulator* for sake of proof
- Differentiating advantage: $M^2/2^{c+1}$ [KT, Eurocrypt 2008]

Outline

Message authentication codes

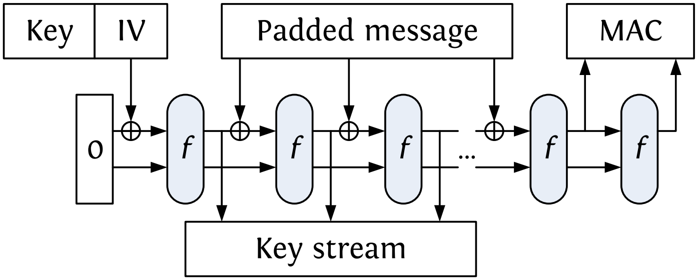


Stream encryption



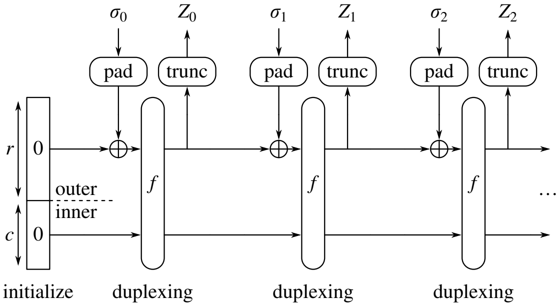
- Long output stream per IV: similar to OFB mode
- Short output stream per IV: similar to counter mode

Authenticated encryption: spongeWrap [KT, SAC 2011]



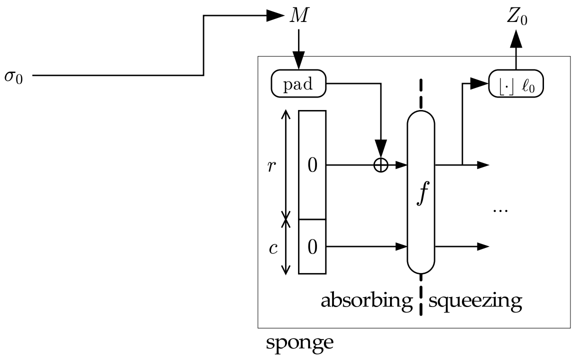
- Adopted by several CAESAR candidates
- But this is no longer sponge

The duplex construction [KT, SAC 2011]



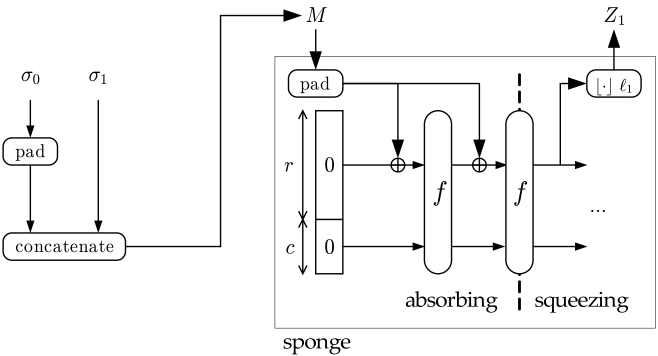
Generic security equivalent to that of sponge

Generating duplex responses with a sponge



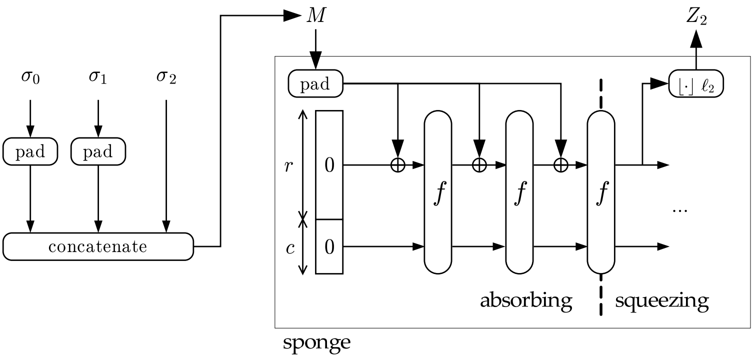
$$Z_0 = \text{sponge}(\sigma_0, \ell_0)$$

Generating duplex responses with a sponge



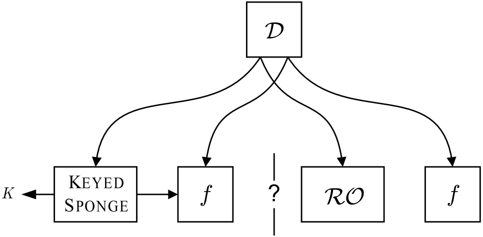
$$Z_1 = \text{sponge}(\text{pad}(\sigma_0)||\sigma_1, \ell_1)$$

Generating duplex responses with a sponge



$$Z_2 = \text{sponge}(\text{pad}(\sigma_0)||\text{pad}(\sigma_1)||\sigma_2, \ell_2)$$

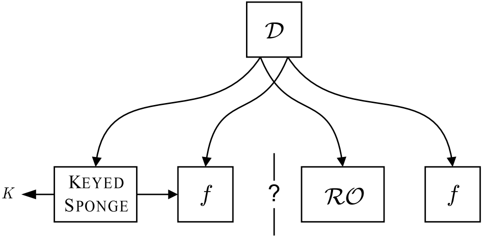
Keyed sponge: distinguishing setting



- Straightforward bound: $M^2/2^{c+1} + M/2^k$
- Security strength s: expected complexity of succesful attack
  - strength s means attack complexity 2^s
  - bounds can be converted to security strength statements
- Here: $s \leq \min(c/2, k)$
  - e.g., $s = 128$ requires $c = 256$ and $k = 128$
  - $c/2$: birthday bound

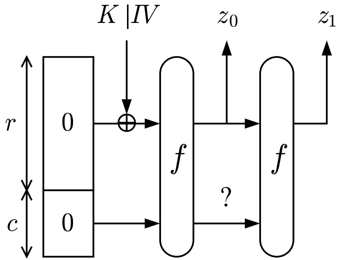
Outline

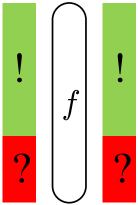
More fine-grained attack complexity



- Splitting attack complexity:
  - queries to construction: data complexity M
  - queries to f or f^{-1}: computational complexity N
- Our ambition around 2010: $M^2/2^{c+1} + NM/2^c + N/2^k$
- If we limit data complexity $M \leq 2^a \lll 2^{c/2}$:
  - $s \leq \min(c - a, k)$
  - e.g., $s = 128$ and $a = 64$ require $c = 192$ and $k = 128$

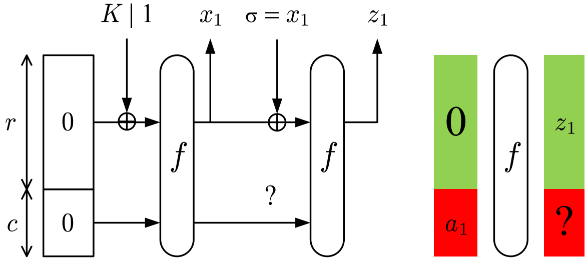
Intuition behind $NM/2^c$





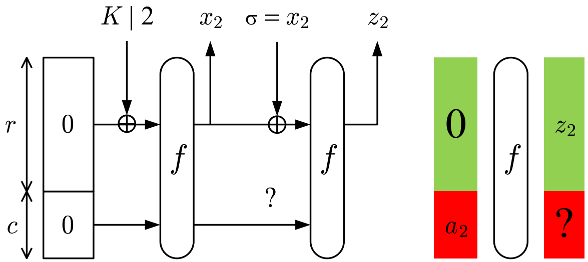
- success probability per guess: $1/2^c$

Intuition behind $NM/2^c$



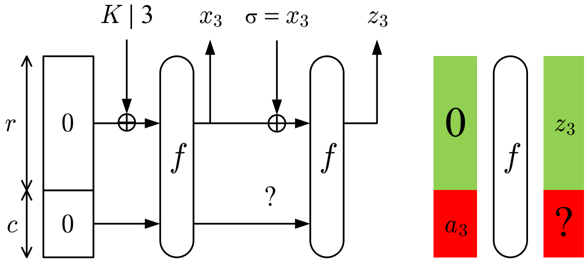
- $\mu \leq M$ instances with same partial r-bit input
- success probability per guess: $\mu/2^c$

Intuition behind $NM/2^c$



- $\mu \leq M$ instances with same partial r-bit input
- success probability per guess: $\mu/2^c$

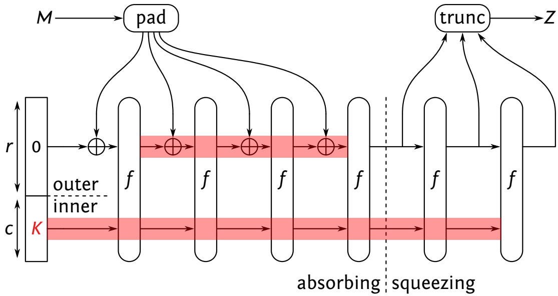
Intuition behind $NM/2^c$



- $\mu \leq M$ instances with same partial r-bit input
- success probability per guess: $\mu/2^c$

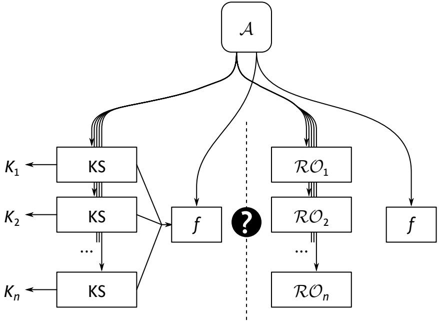
An initial attempt [KT, SKEW 2011]

- bound: $M^2/2^{c+1} + NM/2^{c-1} + N/2^k$
- Problems and limitations
  - bound did not cover multi-target (key) attacks
  - proof did not convince reviewers
  - new variant (a.o. in CAESAR): inner-keyed sponge:

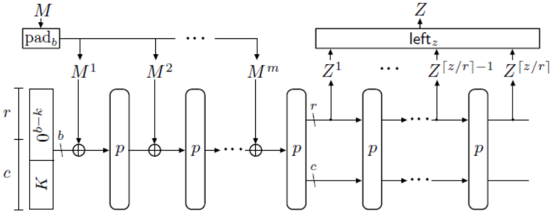


[Andreeva, Daemen, Mennink, Van Assche, FSE 2015]

- Inner/outer-keyed, multi-target (n), multiplicity μ
- Modular proof using Patarin's H-coefficient technique
- Bound: $M^2/2^{c+1} + \mu N/2^{c-1} + nN/2^k + \dots$

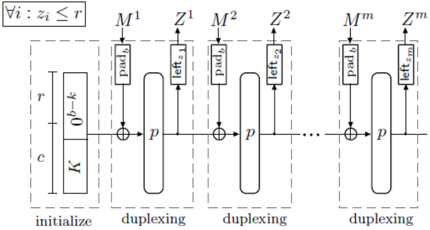


Full-state absorbing! [Mennink, Reyhanitabar and Vizár, Asiacrypt 2015]



- Absorbing on full permutation width does not degrade bounds
- We decided to use that insight in KEYAK v2
- But proven bounds had some limitations and problems:
  - term $\mu N/2^k$ rather than $\mu N/2^c$
  - no multi-key security
  - multiplicity μ only known a posteriori

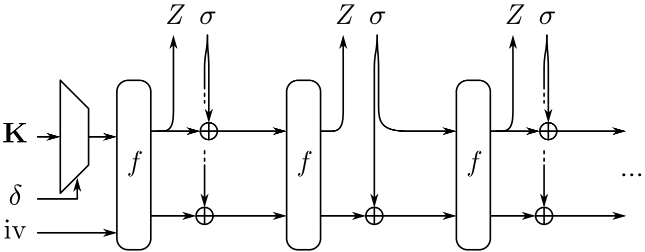
Full-state absorbing! [Mennink, Reyhanitabar and Vizár, Asiacrypt 2015]



- Absorbing on full permutation width does not degrade bounds
- We decided to use that insight in KEYAK v2
- But proven bounds had some limitations and problems:
  - term $\mu N/2^k$ rather than $\mu N/2^c$
  - no multi-key security
  - multiplicity μ only known a posteriori

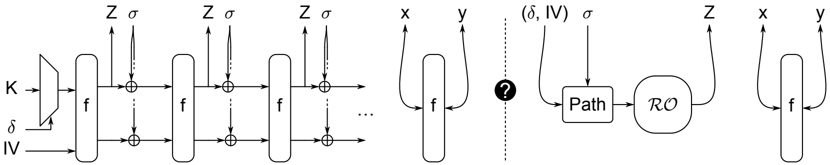
Outline

The new core: (full-state) keyed duplex



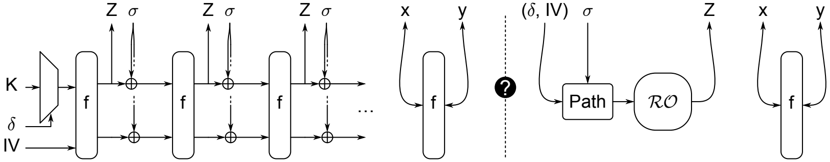
- Full-state absorbing, no padding: $|\sigma| = b$
- Initial state: concatenation of key k and IV
- Multi-key: k selected from an array $\mathbf{K}$ with index δ
- Re-phased: f, Z, σ instead of σ, f, Z
- $\approx$ all keyed sponge functions are modes of this

Generic security of keyed duplex: the setup



- Ideal function: Ideal eXtendable Input Function (IXIF)
  - $\mathcal{RO}$-based object with duplex interface
  - Independent outputs Z for different paths
- Further refine adversary's capability
  - L: # queries to keyed duplex/$\mathcal{RO}$ with repeated path
  - q_{IV} : $\max_{IV}$ # init queries with different keys

Generic security of keyed duplex: the bound



$$L^2/2^{c+1} + (L + 2\nu)N/2^c + q_{\mathrm{IV}}N/2^k + \ldots$$

with ν: chosen such that probability of ν-wise multi-collision in set of M r-bit values is negligible

Joint work with Gilles Van Assche and Bart Mennink, in submission

Application: counter-like stream cipher

- Only init calls, each taking Z as keystream block
- IV is nonce, so $L = 0$
- Assume $M \lll 2^{r/2}$: $\nu = 1$

Bound:

$$(2\nu)N/2^c + q_{\mathrm{IV}}N/2^k + \dots$$

Strength:

$$s \leq \min(c - 1, k - \log_2(q_{\mathrm{IV}}))$$

Application: lightweight MAC

- Message padded and fed via IV and σ blocks
- t-bit tag, squeezed in chunks of r bits: $c = b - r$
- adversary chooses IV so $L \approx M = 2^a$
- q_{IV} is total number of keys n

Bound:

$$M^2/2^{c+1} + MN/2^{c-1} + nN/2^k + \dots$$

Strength:

$$s \leq \min(b - a - r - 1, k - \log_2(n))$$

Imposes a minimum width of the permutation:

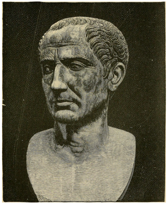
$$b > s + a + r$$

Outline

What is authenticated encryption (AE)?

- Messages and cryptograms
  - $M = (AD, P)$ message with associated data and plaintext
  - $M_c = (AD, C, T)$ cryptogram with assoc. data, ciphertext and tag
- All of M is authenticated but only P is encrypted
  - wrapping: M to M_c
  - unwrapping: M_c to M
- Symmetric cryptography: same key used for both operations
- Authentication aspect
  - unwrapping includes *verification* of tag T
  - if not valid, it returns an error $\perp$
- Note: this is usually called AEAD

The CAESAR competition



- Public competition for AE schemes
  - consortium from academia and industry
  - aims for portfolio instead of single winner
  - CAESAR committee (secretary Dan Bernstein)
- Timeline
  - submission deadline: March 15, 2014
  - end of round 1: July 7, 2015
  - end of round 2: August 15, 2016
  - target end date: December 2017
- Status:
  - Round 1: 57 candidates
  - Round 2: 29
  - Round 3: 15 left

`http://competitions.cr.yp.to/caesar-submissions.html`

Limitations of AE

- No protection against traffic analysis
  - AE does not hide length and number of messages
  - to be addressed separately: random padding and dummy messages
- Determinism: equal messages lead to equal cryptograms
  - information leakage
  - concern of replay attacks
  - solution: ensure message uniqueness at wrapping end
  - include nonce N in input when wrapping
    - wrapping becomes stateful
    - a simple message counter suffices
  - From now on we always include a nonce N

Functional behaviour

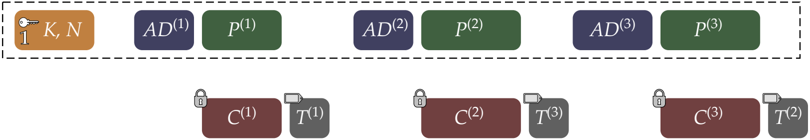
- Wrapping:
  - state: K and past nonces $\mathcal{N}$
  - input: $M = (N, AD, P)$
  - output: C, T or $\perp$
  - processing:
    - if $(N \in \mathcal{N})$ return $\perp$
    - else add N to $\mathcal{N}$ and return $C, T \leftarrow \text{Wrap}[K](N, AD, P)$
- Unwrapping:
  - state: K
  - input: $M_c = (N, AD, C, T)$
  - output: P or $\perp$
  - processing:
    - return $\text{Unwrap}[K](N, AD, C, T)$: P if valid and $\perp$ otherwise

Sessions

- Session: tag in cryptogram authenticates also previous messages
  - full sequence of messages since the session started
- Additional protection against:
  - insertion,
  - omission,
  - re-ordering of messages within a session
- Attention point: *last* message of session
- Alternative views:
  - split of a long cryptogram in shorter ones
  - intermediate tags

See [Bellare, Kohno and Namprempre, ACM 2003], [Keccak Team, SAC 2011], [Boldyreva, Degabriele, Paterson, Stam, EC 2012] and [Hoang, Reyhanitabar, Rogaway and Vizár, 2015]

Functional behaviour, with sessions



- Session start: creation of stateful session object D
  - if $(N \in \mathcal{N})$ (past nonces) return $\perp$
  - else add N to $\mathcal{N}$ and create D with $\textsc{State} \leftarrow \mathsf{Start}(K, N)$
- Wrapping
  - return $C^{(i)}, T^{(i)} \leftarrow \mathsf{D.Wrap}(AD^{(i)}, P^{(i)})$
  - this updates $\textsc{State}$
- Unwrapping
  - return $\mathsf{D.Unwrap}(AD^{(i)}, C^{(i)}, T^{(i)})$: $P^{(i)}$ or $\perp$
  - in case of no error, this updates $\textsc{State}$

Why (session-based) authenticated encryption?

- Convenience
  - often both are confidentiality and integrity are needed
  - one scheme to choose instead of two
- Efficiency
  - combination can be more efficient than sum of the two, e.g.,
  - CBC encryption and CMAC: 2 block cipher calls per input block
  - OCB3 AE: 1 block cipher call per input block
  - sponge-based AE: 1 permutation call per input block
- Reduction of attack surface
  - differential attacks limited to session setup due to nonce
  - chosen ciphertext attacks ineffective due to $\perp$
- Increase of robustness against fault attacks
  - in wrap due to nonce requirement
  - in unwrap due to $\perp$

An ideal AE scheme

- Underlying primitive: random oracle $\mathcal{RO}$
  - output length ℓ implied by the context
  - $\mathcal{RO}_e(\cdot) = \mathcal{RO}(\cdot||1)$ for encryption
  - $\mathcal{RO}_a(\cdot) = \mathcal{RO}(\cdot||0)$ for tag computation
- Wrapping
  - if $(N \in \mathcal{N})$ it return $\perp$
  - $C \leftarrow \mathcal{RO}_e(K||N||AD) \oplus P$
  - $T \leftarrow \mathcal{RO}_a(K||N||AD||P)$
- Unwrapping
  - $P \leftarrow \mathcal{RO}_e(K||N||AD) \oplus C$
  - $T' \leftarrow \mathcal{RO}_a(K||N||AD||P)$
  - If $(T' \neq T)$ return $\perp$, else return P
- Note: $\mathcal{RO}$ input shall be uniquely decodable in K, N AD & P

Ideal AE scheme, now supporting sessions

- Starting the session
  - if $(N \in \mathcal{N})$ it return $\perp$
  - History $\leftarrow K||N$
- Wrapping of $M^{(i)} = (AD^{(i)}, P^{(i)})$
  - History $\leftarrow$ History$||AD^{(i)}||1$ and $C^{(i)} \leftarrow \mathcal{RO}(\text{History}) \oplus P^{(i)}$
  - History $\leftarrow$ History$||P^{(i)}||0$ and $T^{(i)} \leftarrow \mathcal{RO}(\text{History})$
  - return $(C^{(i)}, T^{(i)})$
- Unwrapping of $M_c^{(i)} = (AD^{(i)}, C^{(i)}, T^{(i)})$
  - save current state in case of error: $S' \leftarrow$ History
  - History $\leftarrow$ History$||AD^{(i)}||1$ and $P^{(i)} \leftarrow \mathcal{RO}(\text{History}) \oplus C^{(i)}$
  - History $\leftarrow$ History$||P^{(i)}||0$ and $\tau \leftarrow \mathcal{RO}(\text{History})$
  - if $(\tau = T^{(i)})$ return $P^{(i)}$,
  - else History $\leftarrow S'$ and return $\perp$
- Note: History shall be uniquely decodable in K, N $AD^{(i)}$ & $P^{(i)}$

Security of the ideal AE scheme

- Attack model: adversary can adaptively query:
  - Start, respecting nonce uniqueness (not counted),
  - D.Wrap (q_w times) and D.Unwrap (q_u times)
  - $\mathcal{RO}(x)$: n times
- Input to $\mathcal{RO}(K||\cdot)$ never repeats: outputs are uniformly random
  - intra-session: each input to $\mathcal{RO}$ is longer than previous one
  - inter-session: first part of $\mathcal{RO}$ input (N, K) never repeated
  - So cryptograms $C^{(i)}$ and tags $T^{(i)}$ are uniformly random

Security of our ideal AE scheme (cont'd)

- Forgery:
  - building sequence of valid cryptograms $M_c^{(1)} \dots M_c^{(\ell)}$
  - not obtained from calls to wrap for some $M^{(1)} \dots M^{(\ell)}$
- Privacy break:
  - learning on plaintext bits of $M_c^{(\ell)}$
  - without unwrapping all of $M_c^{(1)} \dots M_c^{(\ell)}$
- Complete security breakdown: key recovery
  - single target key: getting one specific key
  - multiple target: getting one key out of m target keys

Security of our ideal AE scheme (cont'd 2)

- Forgery
  - best strategy: send random but well-formatted cryptograms
  - success probability for q_u attempts: $q_u 2^{-|T|}$
- Privacy break
  - best strategy: unwrap cryptograms with modified C_i or T_i
  - success probability for q_u attempts: $q_u 2^{-|T|}$
- Key retrieval
  - best strategy: exhaustive key search
  - single target: success prob. for n key guesses $\approx n2^{-|K|}$
  - multi-target: success prob. for n key guesses $\leq (m+1)n2^{-|K|}$
  - Remedy against multi-target security erosion: global nonce
- Summary:
  - 1-of-m key recovery after $2^{|K|-\log_2(m+1)}$ offline calls to $\mathcal{RO}(\cdot)$
  - single privacy break/forgery after $2^{|T|}$ online calls to D.Unwrap

Instantiating our ideal AE scheme

- Replace $\mathcal{RO}$ by full-state keyed duplex calling e.g. KECCAK-f
- Due to distinguishing bound :
  - key recovery: $\min(2^{|K|-\log_2 m}, 2^{c-\epsilon})$ offline calls to f
  - privacy break/forgery: $\min(2^{|T|}, 2^{c/2})$ online calls to f
  - ... assuming f (i.e., KECCAK-f) has no exploitable properties
- Practical scheme?
  - History includes all previous messages
  - storing it may require huge buffer
- Practical scheme!
  - keyed duplex is hard to distinguish from $\mathcal{RO}$
  - it compresses all History in its b-bit state S
  - at any point S: keyed hash of History
  - instantiations: our CAESAR submission KEYAK (and KETJE)

Instantiating our ideal AE scheme

- Replace $\mathcal{RO}$ by full-state keyed duplex calling e.g. KECCAK-f
- Due to distinguishing bound :
  - key recovery: $\min(2^{|K|-\log_2 m}, 2^{c-\epsilon})$ offline calls to f
  - privacy break/forgery: $\min(2^{|T|}, 2^{c/2})$ online calls to f
  - … assuming f (i.e., KECCAK-f) has no exploitable properties
- Practical scheme?
  - History includes all previous messages
  - storing it may require huge buffer
- Practical scheme!
  - keyed duplex is hard to distinguish from $\mathcal{RO}$
  - it compresses all History in its b-bit state S
  - at any point S: keyed hash of History
  - instantiations: our CAESAR submission KEYAK (and KETJE)

Advantages of sponge-based AE

- Smaller surface for cryptanalysis than block-cipher modes
  - there are no round keys
  - evolving state during session: moving target
- Cheaper protection against side channel attacks
  - DPA and DEMA limited to session setup due to nonce unicity
  - moving target during session
- Optimization of ratio security strength vs memory usage

Wish for being *online*

- Online: *being able to wrap or unwrap a message on-the-fly*
- Avoid having to buffer long messages
- Online unwrapping implies returning unverified plaintext
  - but security of our scheme relies on it
  - two ways to tackle this problem
- Tolerating Release of Unverified Plaintext (RUP)
  - catastrophic fragmentation attack [Albrecht et al., IEEE S&P 2009]
  - add security notions and attacks [Andreeva et al., ASIACRYPT 2014]
  - try to satisfy (some of) these: costly
- This can be addressed with sessions
  - split long cryptogram into short ones, each with tag
  - shorten cryptograms til they fit the unwrap buffer

Wish for surviving sloppy nonce management

- Our assumption: K, N is unique per (wrapping) Session Start
  - users/implementers do not always respect this
  - wish to limit consequences of nonce violation
- All online AE schemes leak in case of nonce violation
  - equality of first messages of session leaks in any case
    - stream encryption: re-use of keystream
    - block encryption: just equality of block(s) leaks
  - low entropy plaintexts become an issue
  - successful active attacks for quasi all proposed schemes
- Consensus among experts on following:
  - ideal security in case of nonce misuse hard to define
  - user shall be warned to not allow nonce violation
- Just avoid nonce violation

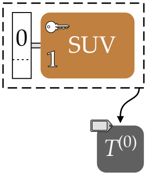
Wish for parallelism

- Many CAESAR submissions use AES
- Modern CPUs have dedicated AES instruction, e.g. AES-NI on Intel
  - pipelining: 1 cycle per round but latency of 8 to 16 cycles
  - performing a single AES: 80 cycles
  - performing 8 independent AES: 88 cycles
- Expoiting the pipeline requires ability to parallelize
- Also non-AES based schemes can benefit from parallelism, e.g.
  - pipelined architectures
  - superscalar architectures
  - SIMD instructions
- Parallelism can be supported, e.g., Keyak

Wish for lightweight

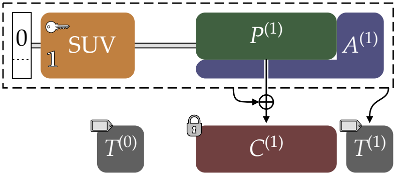
- Whole world of buzzwords:
  - IoT, Smart Grid, RFID, ad-hoc sensor and body area network, ...
- Strongly constrained resources
  - low area: reduce chip cost
  - low power: RF powered
  - low energy: battery-life
- Specific conditions
  - short messages
  - transaction time, ...
- Compromising on
  - target security strength
  - provable security of mode
  - consequences of improper usage, ...
- Hence: KETJE is dedicated for lightweight

The Motorist mode of use



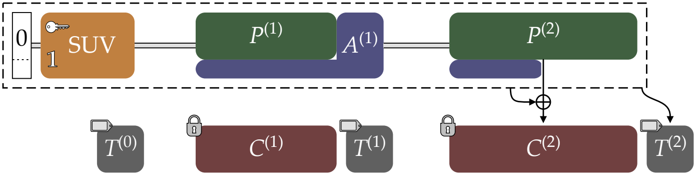
- SUV = Secret and Unique Value
- Plaintext absorbed in outer part, AD in inner part also
- Tag and keystream from same output block Z_i
- Specified in three layers:
  - Piston: Π of them, each one an FSKD
  - Engine: finite state machine steering the Piston(s)
  - Motorist: session starting and (un)-wrapping, using the Engine

The Motorist mode of use



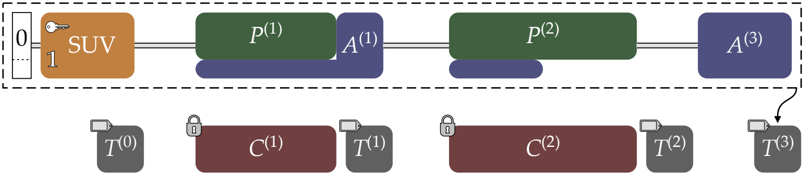
- SUV = Secret and Unique Value
- Plaintext absorbed in outer part, AD in inner part also
- Tag and keystream from same output block Z_i
- Specified in three layers:
  - Piston: Π of them, each one an FSKD
  - Engine: finite state machine steering the Piston(s)
  - Motorist: session starting and (un)-wrapping, using the Engine

The Motorist mode of use



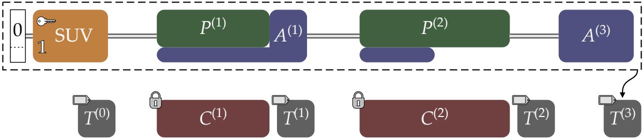
- SUV = Secret and Unique Value
- Plaintext absorbed in outer part, AD in inner part also
- Tag and keystream from same output block Z_i
- Specified in three layers:
  - Piston: Π of them, each one an FSKD
  - Engine: finite state machine steering the Piston(s)
  - Motorist: session starting and (un)-wrapping, using the Engine

The Motorist mode of use



- SUV = Secret and Unique Value
- Plaintext absorbed in outer part, AD in inner part also
- Tag and keystream from same output block Z_i
- Specified in three layers:
  - Piston: Π of them, each one an FSKD
  - Engine: finite state machine steering the Piston(s)
  - Motorist: session starting and (un)-wrapping, using the Engine

Generic security of Motorist AE session mode



Used in KEYAK v2 [KT & Ronny Van Keer, 2015]

- Plaintext absorbed in outer part, AD in inner part also
- Used in KEYAK with $c = 256$ and $b = 1600$ or $b = 800$
- Rate 544 or 1344 so we can take $\nu = 1$
- bounds:
  - nonce-respecting: $N/2^{c-1} + q_{\mathrm{IV}} N/2^k + \ldots$
  - nonce-violating: $MN/2^c + q_{\mathrm{IV}} N/2^k + \ldots$

Outline

KEYAK [KECCAK team + Ronny Van Keer]

- AE scheme submitted to CAESAR (tweaked for round 2)
- Permutation-based mode called Motorist
- Makes use of KECCAK-p permutations
  - KECCAK-p: reduced-round version of KECCAK-f
  - KECCAK-f: permutations underlying KECCAK
  - all 6 functions in SHA-3 based on KECCAK-$f[1600]$ (24 rounds)
- Generic definition with 5 parameters

c	capacity
τ	tag length
b	width of KECCAK-p
n_{r}	number of rounds in KECCAK-p
Π	degree of parallelism

Keyak named instances

- 5 named instances with $c = 256, \tau = 128, n_{\mathrm{r}} = 12$
- Efficiency:
  - Short messages: Π calls to Keccak-p
  - Long messages: twice as fast as SHAKE128

Name	Width b	Parallelism Π
River Keyak	800	1
Lake Keyak	1600	1
Sea Keyak	1600	2
Ocean Keyak	1600	4
Lunar Keyak	1600	8

KETJE [KECCAK team + Ronny Van Keer]

- AE scheme submitted to CAESAR (made it to round 2)
- Two instances
- Functionally similar to KEYAK
- Lightweight:
  - using reduced-round KECCAK-f[400] or KECCAK-f[200]
  - small footprint
  - low computation for short messages
- How?
  - 96-bit or 128-bit security (incl. multi-target)
  - more ad-hoc: MONKEYDUPLEX instead of FSKD
  - reliance on nonce uniqueness for key protection

KETJE instances and lightweight features

feature		Ketje Jr	Ketje Sr
state size		25 bytes	50 bytes
block size		2 bytes	4 bytes
processing		**computational cost**	
session start	per session	12 rounds	12 rounds
wrapping	per block	1 round	1 round
8-byte tag comp.	per message	9 rounds	7 rounds

More on KETJE and KEYAK:
http://ketje.noekeon.org
http://keyak.noekeon.org

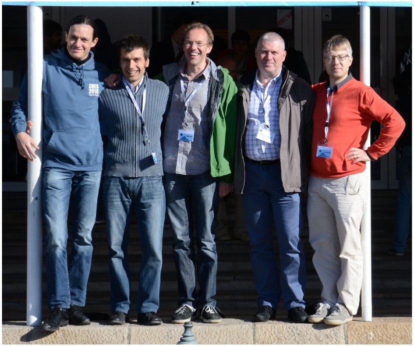
Safety margin of Keyak and Ketje

S. Huang, M. Wang, X. Wang and J. Zhao, Conditional cube attack on reduced-round keccak sponge function [IACR eprint 2016/790]

- Best current cryptanalysis of keyed Keccak-f modes
- Cube attack
  - exploits low algebraic degree of permutation
  - n rounds has degree 2^n
  - summing over inputs in affine space acts as differentiation
  - attack requires summing over around 2^n inputs
  - smart tricks allow peeling off rounds
- Most powerful attacks on Keyak (12 rounds)
  - 7-round variant: requires 2^{42} blocks of chosen data
  - 8-round variant: requires 2^{74} blocks of chosen data

Conclusion

Permutations: good alternative for block ciphers



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