On authenticated encryption and the CAESAR competition

Joan Daemen

STMicroelectronics and Radboud University

Crypto summer school 2015
Šibenik, Croatia, May 31 - June 5, 2015

Outline

What is authenticated encryption (AE)?

- Messages and cryptograms
  - $M = (AD, P)$ message with associated data and plaintext
  - $M_c = (AD, C)$ cryptogram with associated data and ciphertext
- 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 M_c
  - if not valid, it returns an error $\perp$
  - wrap operation adds redundancy: $|C| > |P|$
  - often redundancy coded at the end of C: tag T
- Note: this is usually called AEAD

Limitation of AE: traffic analysis

- Traffic analysis:
  - length of messages
  - number of messages
- Solution
  - creating dummy messages
  - random-length padding of plaintext
  - to be done on higher layer
- AE scheme security should be independent from this layer

Limitation of AE: need for message uniqueness

- Concrete AE proposals are deterministic
- Equal messages lead to equal cryptograms
  - information leakage
  - concern of replay attacks at unwrapping end
- Solution is using *nonces* (Number used only ONCE)
  - impose that the AD is a nonce for the given key K
  - often presented as a separate field N
  - wrapping engine shall ensure (K, N) is unique
    - 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 or $\perp$
  - processing:
    - if $(N \in \mathcal{N})$ return $\perp$
    - else add N to $\mathcal{N}$ and return $C \leftarrow \text{Wrap}[K](N, AD, P)$
- Unwrapping:
  - state: K
  - input: $M_c = (N, AD, C)$
  - output: P or $\perp$
  - processing:
    - return $\text{Unwrap}[K](N, AD, C)$: P if valid and $\perp$ otherwise

Sessions

- Session: 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 view:
  - splits a long cryptogram in shorter ones
  - intermediate tags

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

Functional behaviour, with sessions

- Initialization of stateful session object D
  - state: past nonces $\mathcal{N}$ (may be omitted for unwrapping)
  - input: key K, nonce N
  - processing:
    - if $(N \in \mathcal{N})$ return $\perp$
    - else add N to $\mathcal{N}$ and create D with $\mathrm{D.S} \leftarrow \mathrm{Init}(K, N)$
  - D.S will be updated during the session
- Wrapping
  - return $C^{(i)} \leftarrow \mathrm{D.Wrap}(AD^{(i)}, P^{(i)})$
  - this updates D.S
- Unwrapping
  - return $\mathrm{D.Unwrap}(AD^{(i)}, C^{(i)})$: $P^{(i)}$ or $\perp$
  - in case of no error, this updates D.S
  - session may be aborted after specific number of errors

Outline

An ideal AE scheme

- Separate fixed-length tag, so $M_c = (N, AD, C, T)$
- Functional components: random oracle $\mathcal{RO}$
  - variable output length, 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 and P

Ideal AE scheme, now supporting sessions

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

Security of our ideal AE scheme

- Attack model: adversary can adaptively query:
  - Init, 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 at unwrap: send 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 probability for n key guesses $\approx n2^{-|K|}$
  - multi-target: success probability for n key guesses $\leq (m+1)n2^{-|K|}$
  - Countermeasure against multi-target security erosion: global nonce
- Summary:
  - 1 out of m keys 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

Outline

Instantiating our ideal AE scheme

- Replace $\mathcal{RO}$ by a sponge function like KECCAK
- Thanks to $\mathcal{RO}$-differentiating bound of sponge [KT, EC 2008]:
  - key recovery: $\min(2^{|K|-\log_2 m}, 2^{c/2})$ offline calls to KECCAK-f
  - privacy break/forgery: $\min(2^{|T|}, 2^{c/2})$ online calls to KECCAK-f
  - ... assuming KECCAK-f has no exploitable properties
  - tighter bounds in [Andreeva, Daemen, Mennink, Van Assche, FSE 2015]
- Practical scheme?
  - D.S buffers all previous messages
  - Input to our sponge includes all messages
- Practical scheme!
  - sponge operates sequentially on a b-bit state S
  - update this state S on the fly
  - instantiations: our designs KEYAK and KETJE

Instantiating our ideal AE scheme

- Replace $\mathcal{RO}$ by a sponge function like KECCAK
- Thanks to $\mathcal{RO}$-differentiating bound of sponge [KT, EC 2008]:
  - key recovery: $\min(2^{|K|-\log_2 m}, 2^{c/2})$ offline calls to KECCAK-f
  - privacy break/forgery: $\min(2^{|T|}, 2^{c/2})$ online calls to KECCAK-f
  - ... assuming KECCAK-f has no exploitable properties
  - tighter bounds in [Andreeva, Daemen, Mennink, Van Assche, FSE 2015]
- Practical scheme?
  - D.S buffers all previous messages
  - Input to our sponge includes all messages
- Practical scheme!
  - sponge operates sequentially on a b-bit state S
  - update this state S on the fly
  - instantiations: our designs KEYAK and KETJE

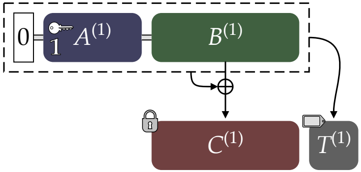
KEYAK [KECCAK team + Ronny Van Keer]

- Four instances, all with 128 bits of security strength
- Architecture in multiple layers
  - permutation: reduced-round KECCAK-$f[1600]$ or KECCAK-$f[800]$
  - duplex construction: alternating input with output
  - DuplexWrap mode: unique decodability and domain separation
  - (optional) KeyakLines mode: for parallelizable instances
- Generic security thanks to a combination of results:
  - keyed sponge distinguishing bounds [Andreeva, Daemen, Mennink, Van Assche, FSE 2015]
  - security equivalence of sponge and duplex [KT, SAC 2011]
  - SPONGEWRAP generic security [KT, SAC 2011], adapted to DUPLEXWRAP
  - sound tree hashing modes [KT, IJIS 2013] for parallelized modes

DuplexWrap layer

DuplexWrap

- nonce-based authenticated encryption mode
- works on sequences of header-body pairs

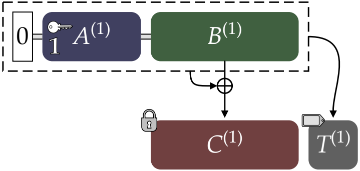


- $A^{(1)}$ contains the key and must be unique, e.g.,
  - $A^{(1)}$ contains a session key used only once
  - $A^{(1)}$ contains a key and a nonce
  - in general: $A^{(1)} = K||N||AD^{(1)}$
- $B^{(i)} = P^{(i)}$ and for $i > 1 : A^{(i)} = AD^{(i)}$

DuplexWrap layer

DuplexWrap

- nonce-based authenticated encryption mode
- works on sequences of header-body pairs

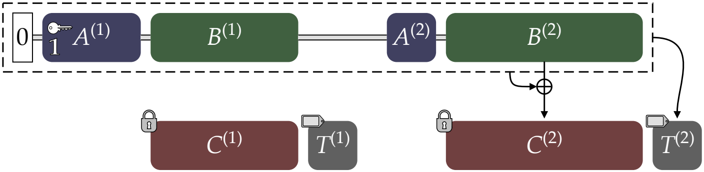


- $A^{(1)}$ contains the key and must be unique, e.g.,
  - $A^{(1)}$ contains a session key used only once
  - $A^{(1)}$ contains a key and a nonce
  - in general: $A^{(1)} = K||N||AD^{(1)}$
- $B^{(i)} = P^{(i)}$ and for $i > 1 : A^{(i)} = AD^{(i)}$

DuplexWrap layer

DuplexWrap

- nonce-based authenticated encryption mode
- works on sequences of header-body pairs

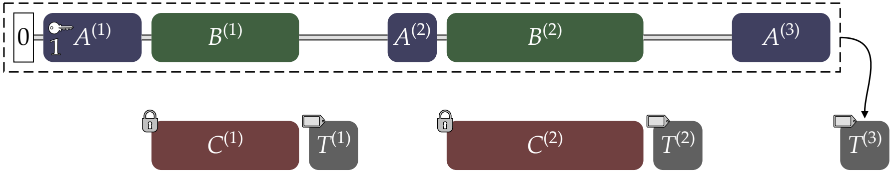


- $A^{(1)}$ contains the key and must be unique, e.g.,
  - $A^{(1)}$ contains a session key used only once
  - $A^{(1)}$ contains a key and a nonce
  - in general: $A^{(1)} = K||N||AD^{(1)}$
- $B^{(i)} = P^{(i)}$ and for $i > 1 : A^{(i)} = AD^{(i)}$

DuplexWrap layer

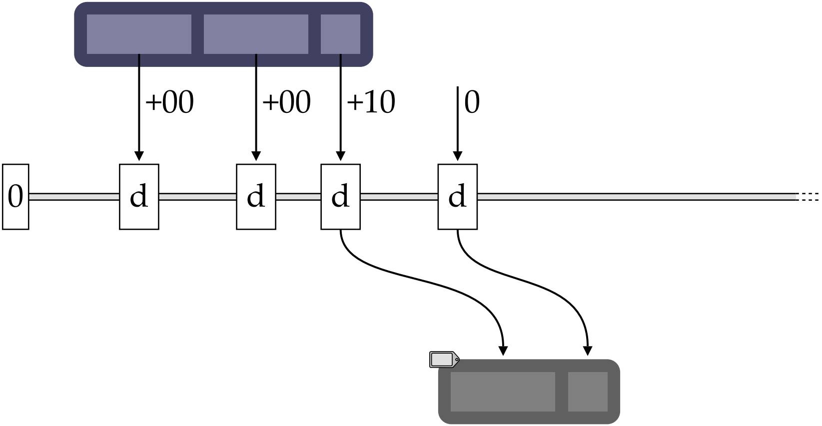
DuplexWrap

- nonce-based authenticated encryption mode
- works on sequences of header-body pairs

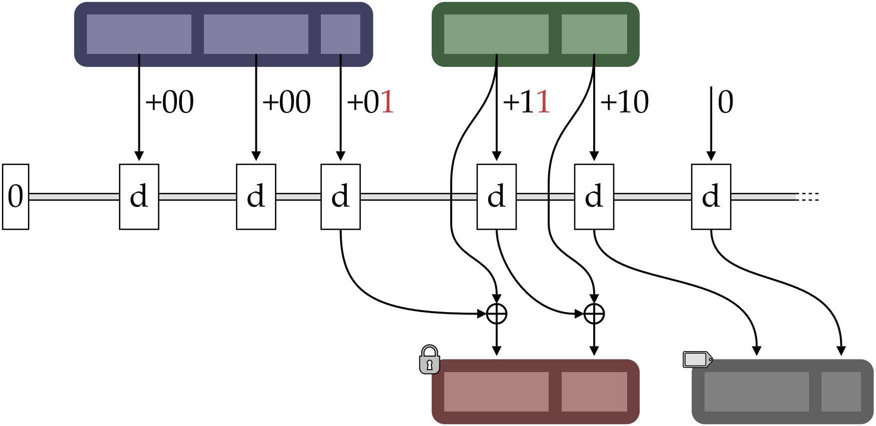


- $A^{(1)}$ contains the key and must be unique, e.g.,
  - $A^{(1)}$ contains a session key used only once
  - $A^{(1)}$ contains a key and a nonce
  - in general: $A^{(1)} = K||N||AD^{(1)}$
- $B^{(i)} = P^{(i)}$ and for $i > 1 : A^{(i)} = AD^{(i)}$

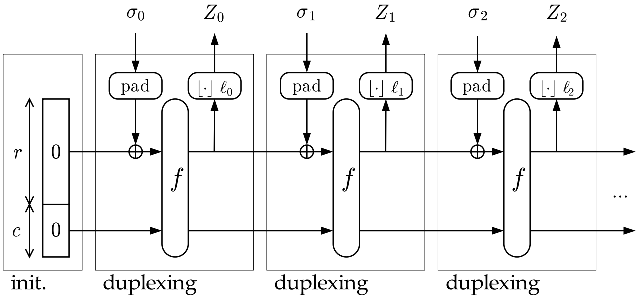
Inside DuplexWrap



Inside DuplexWrap



Duplex layer



$f = \text{KECCAK-}p[1600, n_r = 12]$ or $f = \text{KECCAK-}p[800, n_r = 12]$

- σ_i: a block of header, a block of body or an empty block
- Z_i: a block of keystream, a block of tag or nothing
- blocks are up to $\rho = b - c - 4$ bits long

KEYAK instances

Name	Width b	Parallelism P
RIVER KEYAK	800	1
LAKE KEYAK	1600	1
SEA KEYAK	1600	2
OCEAN KEYAK	1600	4

- 252-bit capacity: 128-bit security if data $< 2^{123}$ blocks [FSE 2015]
  - RIVER KEYAK: block length up to 68 bytes
  - other : block length up to 168 bytes
- Processing for LAKE KEYAK
  - long messages: about 50 % of SHAKE128
  - short messages: 24 rounds
- Working memory footprint
  - reasonable on high- and middle-end platforms
  - not ideal on constrained platforms

KETJE [KECCAK team + Ronny Van Keer]

- 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 duplex
  - 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**	
initialization	per session	12 rounds	12 rounds
wrapping	per block	1 round	1 round
8-byte tag comp.	per message	9 rounds	7 rounds

Outline

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
  - in most models unwrap never returns unverified plaintext
  - two ways to tackle this problem
- Tolerating Release of Unverified Plaintext (RUP)
  - generates additional security notions and attacks [Andreeva, Bogdanov, Luykx, Mennink, Mouha, and Yasuda, ASIACRYPT 2014]
  - try to satisfy (some of) these: costly
  - catastrophic fragmentation attack [Albrecht, Paterson, Watson, IEEE S&P 2009]
- Session approach:
  - split long cryptogram into short ones, each with tag
  - cryptograms short enough to fit the unwrap buffer

Wish for surviving sloppy nonce management

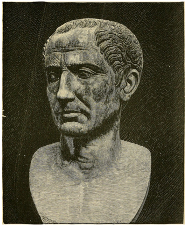
- Our assumption: K, N is unique per call to Init for wrapping
  - users/implementers do not always respect this
  - wish to limit consequences of nonce violation
- All online AE schemes leak in case of nonce violation
  - equality of first messages of session leaks in any case
  - if stream encryption: re-use of keystream
  - if block encryption: just equality of block(s) leaks
  - low entropy plaintexts become an issue
  - successful active attacks for quasi all proposed schemes
- I think there is consensus among experts on the following:
  - hard to give an understandable security definition
  - user shall be warned to not allow nonce violation
  - calling an AE scheme nonce-misuse resistant gives wrong message
- Question: may nonce violation lead to full security breakdown?

Wish for parallelism

- AES is the official NIST and de facto world standard block cipher
- 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
- Filling the pipeline requires parallelism
- Also non-AES based schemes can benefit from parallelism
  - exploiting SIMD instructions
  - exploiting multi-core

Outline

The CAESAR competition



- Public competition for *authenticated ciphers*
  - consortium from academia and industry
  - aims for portfolio instead of single winner
  - CAESAR committee (secretary Dan Bernstein)
- Timeline
  - submission deadline: March 15, 2014
  - 57 submissions
    - many block cipher modes
    - about a dozen sponge-based,
    - including our submissions: KETJE and KEYAK
  - 3 rounds foreseen
    - goal of round 1: reduction to 25 or so candidates
    - we are experiencing some delay ...
  - target end date: December 2017

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

CAESAR candidate statistics (approximate numbers)

- Usage of primitives
  - 12 permutations, 10 new
  - 7 block ciphers, 1 new
  - 6 tweakable block ciphers, all new
  - about 20 submissions use AES
- Modes
  - 16 block encryption modes, 12 new
  - 30 stream encryption modes, 25 new
  - popular modes:
    - sponge-like
    - Even-Mansour
    - OCB
    - COPA
- 9 out of 57 submissions already withdrawn and 1 more broken

Permutation-based modes

- Mostly in two categories: sponge and Even-Mansour
- Sponge: $b = r + c$
  - one (or more) serial data paths
  - stream encryption
  - no permutation inverse needed (except in APE of PRIMATEs)
  - sub-type: non-hermetic approach
    - full security breakdown under nonce violation
    - AES-round (AEGIS, Tiaoxin) and KECCAK-f round (Ketje)
- Even-Mansour: $b = r$
  - permutation to build (tweakable) block cipher
  - parallelizable modes as OCB, COPA, PMAC, CTR, OTR
  - need for permutation inverse (except OTR)

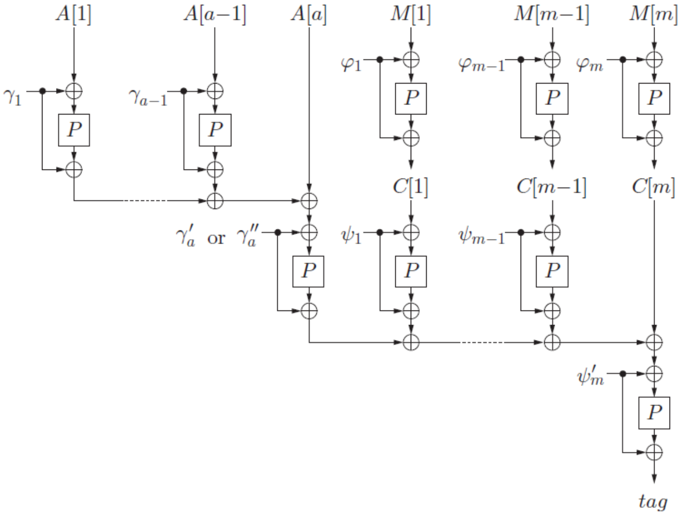
Blockcipher-based modes

- Those that require inverse
  - aiming at nonce-misuse resistance and parallelism
- Those that don't
  - mostly counter mode encryption
  - some sponge-like
  - OTR: block encryption without block cipher inverse!
- Often complex treatment of last block
  - to avoid message expansion due to encryption
  - to reduce the number of block cipher calls for certain message lengths

CAESAR submission Minalpher [Sasaki, Todo, Aoki, Naito, Sugawara, Murakami, Matsui and Hirose]

- Permutation-based mode
- Aims for lightweight
- Primitive: dedicated 256-bit permutation
  - security strength: 128 bits
  - due to birthday bound
- Mode
  - Very parallelizable
  - Permutation used in tweakable Even-Mansour construction
  - One permutation call per 256-bit *AD* block
  - Two permutation calls per 256-bit *P* block

Minalpher Illustrated

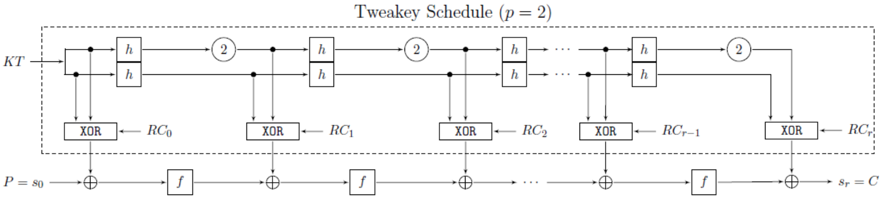


Courtesy Sasaki, Todo, Aoki, Naito, Sugawara, Murakami, Matsui and Hirose

CAESAR submission Deoxys [Jean, Nikolic and Peyrin]

- 2 different modes calling a tweakable block cipher
- Tweakable block cipher Deoxys-BC
  - AES Round function
  - Key schedule replaced by key-and-tweak schedule
  - Tweakey method [Jean, Nikolic and Peyrin 2014]
- ΘCB3 [Rogaway and Krovetz, 2011]
  - fully parallelizable
  - one block cipher call per *AD* or *P* block
- COPA [Andreeva, Bogdanov, Luykx, Mennink and Yasuda, 2013]
  - very parallelizable
  - two block cipher calls per *P* block
  - better behaviour under nonce violation

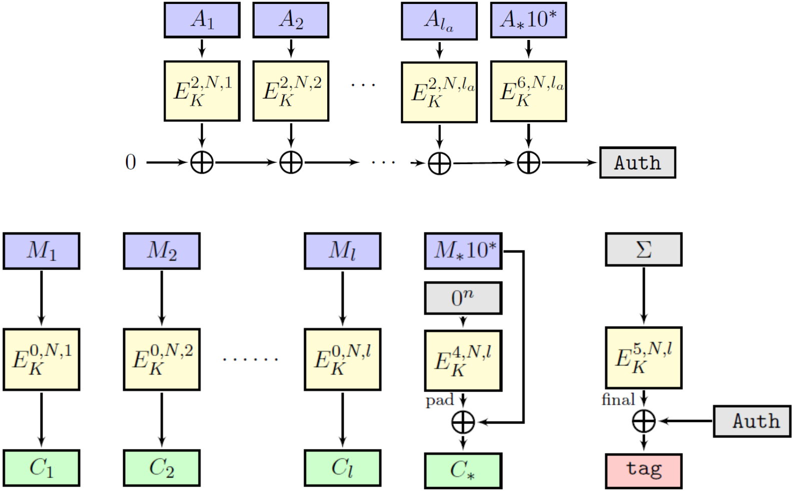
Tweakey



Courtesy Jean, Nikolic and Peyrin

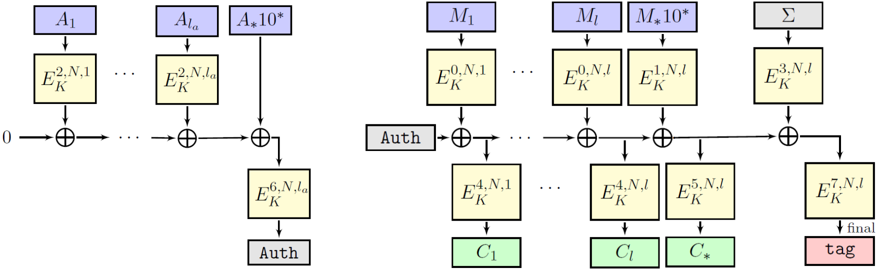
- Idea: integrate tweak in key schedule
  - allows having 128-bit generic security with AES
- Applied to AES
  - *h*: byte transposition
  - 2: multiplication by x in GF(2^8)
  - *KT*: key (top thread) and tweak (bottom thread)
  - proven bounds in chosen-tweak scenario

ΘCB3 illustrated



Courtesy Jean, Nikolic and Peyrin

COPA illustrated



Courtesy Jean, Nikolic and Peyrin

Conclusions

- CAESAR submissions cover a wide range of AE schemes
  - parallel vs compact
  - high throughput vs lightweight
  - software vs hardware oriented
  - side-channel aware or not
  - different levels of robustness against improper usage
  - go see for yourself!
- Interesting ongoing discussions
- In any case:
  - don’t repeat nonces
  - don’t release unverified plaintext

Thanks for your attention!