Radboud University

Authenticated encryption: state-of-the-art and beyond

Joan Daemen, Radboud University, NL

Summer School on real-world crypto and privacy,
Monday June 3, 2024, Vodice, Croatia

ESCADA

What do we want to do?

Protect the communication between Alice and Bob over an open channel that is controlled by a potentially very resourceful adversary Eve

- Confidentiality of the content of messages
- Authentication of the communication
  - Alice can check that any received message was effectively sent by Bob
  - ... and that it is not a *replay* of an old message by Eve
  - ... and that Bob sent it recently: *freshness*

We assume that Alice and Bob share a secret key K that Eve does not know

Confidentiality: encryption with the one-time pad (OTP)

Bob enciphers: $C \leftarrow M \oplus K$

$$
\begin{array}{rcccccccccccccccc}
M = & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & \\
K = & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & \oplus \\
\hline
C = & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 &
\end{array}
$$

Alice deciphers: $M \leftarrow C \oplus K$

$$
\begin{array}{rcccccccccccccccc}
C = & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & \\
K = & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & \oplus \\
\hline
M = & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 &
\end{array}
$$

Unconditional security of the one-time pad

Claude Shannon

Encryption is 100 % secure if ciphertext C gives no information on content of plaintext P

$$\forall p : \mathbf{Pr}(P = p|C) = \mathbf{Pr}(P = p)$$

Claude Shannon proved the security of OTP

- Unconditional: $\forall$ Eve
- Problem: key K must be as long as total amount of plaintext

From one-time pad to practical stream encryption

Stream encryption: bitwise add to the plaintext P a keystream of same length Z

$$C \leftarrow P + Z$$

Stream encryption made practical: generate Z from a short key K and a *diversifier* D

$$Z \leftarrow \mathrm{SC}_K(D) \quad \text{or for short} \quad C \leftarrow P + \mathrm{SC}_K(D)$$

Diversifier is there to generate multiple keystreams Z from a single key K

Stream encryption is secure if D is unique per message and stream cipher is secure

Stream encryption: beware of diversifier re-use!

- Say we use same diversifier D to encipher M_1 and M_2
- This gives same keystream $Z = \mathrm{SC}_K(D)$ for both messages, so:

$$C_1 = M_1 \oplus Z \text{ and } C_2 = M_2 \oplus Z$$

- Eve can compute difference between M_1 and M_2 from ciphertexts:

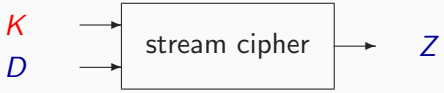
$$C_1 \oplus C_2 = M_1 \oplus Z \oplus M_2 \oplus Z = M_1 \oplus M_2$$

- (partial) knowledge of M_1 gives (partial) knowledge of M_2

Nonce

Diversifier D must be unique per message. We call this a *nonce*: number used once

When is a stream cipher secure?



- For an adversary that knows the algorithm but not the key K
  - $Z = \mathrm{SC}_K(D)$ must be hard to distinguish from a random sequence
  - for values of D freely chosen by the adversary
- You cannot prove this is satisfied
  - you can try to demonstrate this is *not* satisfied using *cryptanalysis*
  - security assurance of a function grows with amount of cryptanalysis it had
  - . . . as long as this does not result in a break

Concrete design requirements: Diffusion and Confusion (C. Shannon)

All bits of $Z = \mathrm{SC}_K(D)$ must be a **complicated** function of **all** bits of D en K

Authenticating a message using a MAC function

Encryption does not guarantee that Alice can detect message tampering by Eve

This can be done by appending an *authentication code* $T \leftarrow \mathrm{MAC}_K(M)$

- Bob computes over the message M an authentication code T (aka *tag*)
  - M can be long, T is short, e.g., 16 bytes
  - computation of T requires a secret key K
  - with key it is easy to compute T, without key not
- Bob sends (M, T) to Alice
  - Alice uses her key K to compute T' from M
  - if $T = T'$, she accepts the message as coming from Bob
- A *forgery* is a pair (M, T) not coming from Bob, accepted by Alice

Often Alice wants to authenticate more, requiring *active checks* from her part

Authenticating freshness

Alice wants to authenticate that the message from Bob is recent

	Bob		Alice
have:	K		K
		$\xrightarrow{\text{Bob}}$	$R \xleftarrow{\$} \{0,1\}^n$
		$\xleftarrow{R}$	
	$T \leftarrow \mathrm{MAC}_K(R; M)$	$\xrightarrow{M,T}$	$T \stackrel{?}{=} \mathrm{MAC}_K(R; M)$

Requires 3 passes:

- Bob announces he has a message
- Alice provides an unpredictable challenge R
- Bob computes tag and sends (M, T)

Protecting against replay

	Bob		Alice
have:	K, N_B		K, N_A
	$N_B \leftarrow N_B + 1$		
	$T \leftarrow \mathrm{MAC}_K(N_B; M)$	$\xrightarrow{\text{Bob}, N_B, M, T}$	$N_B \stackrel{?}{>} N_A$
			$T \stackrel{?}{=} \mathrm{MAC}_K(N_B; M)$
			$N_A \leftarrow N_B$

N_X: counter keeping track of the number of protocol runs started by Bob

- Requires only a single pass
- Stateful: Bob and must keep track of counters N
- Prevents replay but not *stalling* of messages by Eve

Note: this protocol allows *holes* in the counter sequence, variants exist

Intermezzo: finite field basics

A field is a set S with two operations: addition $+$ and multiplication $\cdot$

- $(S, +)$ forms a group
  - Associative, neutral element 0 and each element a has an inverse $-a$
- $(S \setminus \{0\}, \cdot)$ forms a group
  - Associative, neutral element 1 and each element a has an inverse a^{-1}
- Distributive: $a(b + c) = ab + ac$ and $(a + b)c = ac + bc$
- If S is finite we speak of a finite field
- There exists one finite field for any $\#S = p^n$ with p a prime. Special cases:
  - Prime fields $\mathbb{F}_p$: addition and multiplication modulo p
  - Binary fields $\mathbb{F}_{2^n}$: XOR and "carryless" modular multiplication

A polynomial of degree n in a finite field has at most n roots

Polynomial authentication codes

This type of code requires encoding message and keys as elements of finite field

- Message encoded as a sequence of elements of a finite field S: $M = m_1, m_2, \ldots m_\ell$
- Key K consists of a pair of subkeys, also in the finite field S: $K = z, k$.

Tag computation:

$$T \leftarrow z + \sum_{i=1}^{\ell} m_i k^i$$

This is

- secure if a fresh random key z, k is used for every message
- not secure if a key z, k is used for more than a single message

Re-use of keys for polynomial authentication codes

Consider two messages with length 1, m and m', and their tags T and T'

$$T = z + mk \quad \text{and} \quad T' = z + m'k$$

Subtracting gives

$$T - T' = (m - m')k$$

Solving for k:

$$k = \frac{T - T'}{m - m'}$$

And then determine $z = T - mk$

Re-use leaks the key allowing arbitrary forgery!

Security in case of unique key per message

Say Eve observes a valid pair (M, T)

If fresh keys z, k for each message, probability of successful forgery is $\leq \ell/\#S$

- successful forgery (M', T') would satisfy $T' = z + \sum_{i=1}^{\ell} m'_i k^i$
- subtraction from $T = z + \sum_{i=1}^{\ell} m_i k^i$ gives

$$T - T' + \sum_{i=1}^{\ell} (m'_i - m'_i) k^i = 0$$

Polynomial equation of degree ℓ in the unknown k, so at most ℓ solutions

For any choice of (M', T') the forgery is successful for at most ℓ out of $\#S$ keys k

MAC functions made practical

Problem: each message consumes 2 finite field elements

Solution 1: Carter-Wegman (example for $\mathbb{F}_{2^n}$)

- Generate z with a stream cipher: $z \leftarrow 0^n + \mathrm{SC}_K(D)$
- Requires D to be a nonce per message
- Re-use of D leaks k
- Secure if D is a nonce and stream cipher is secure

Solution 2: Dedicated MAC function $T \leftarrow \mathrm{MAC}_K(M)$

When is a MAC function secure?

- For an adversary that does not know the key K
  - $T = \mathrm{MAC}_K(M)$ must be hard to distinguish from a random sequence
  - for values of M freely chosen by the adversary
- You cannot prove this is satisfied
- But you can *claim* security, hope it is attacked and pray it is not broken

Concrete design requirements: Diffusion and Confusion (C. Shannon)

All bits of $T = \mathrm{MAC}_K(M)$ must be a **complicated** function of **all** bits of M en K

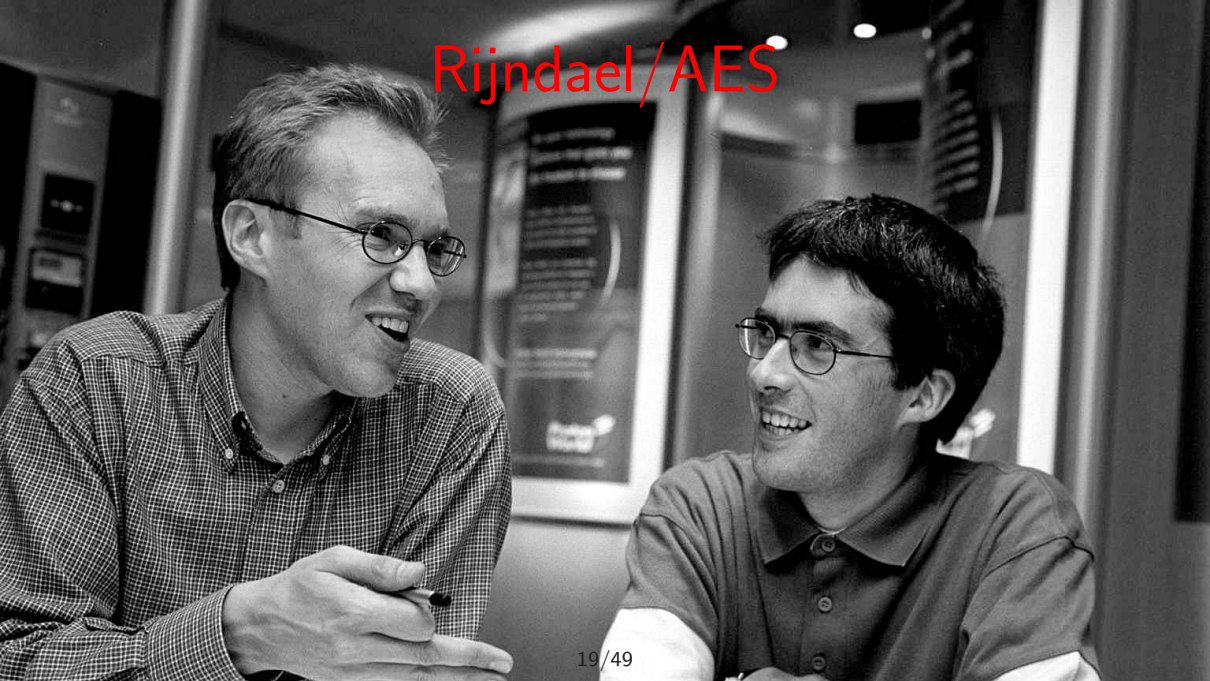
Example of encryption combined with message sequence authentication

Bob		Alice
K, N		K, N'
$N \leftarrow N + 1$		
$C_N \leftarrow M_N \oplus \mathrm{SC}_K(N)$		
$T \leftarrow \mathrm{MAC}_K(N; C_N)$	$\xrightarrow{\text{Bob}, N, C_N, T}$	$N \stackrel{?}{=} N' + 1$
		$T \stackrel{?}{=} \mathrm{MAC}_K(N; C_N)$
		$N' \leftarrow N$
		$M_N \leftarrow C_N \oplus \mathrm{SC}_K(N)$

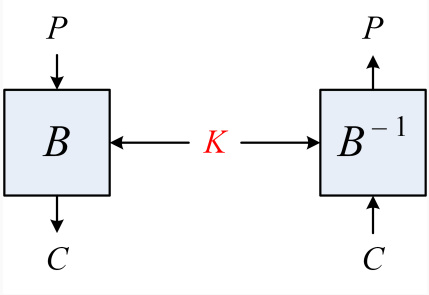
- N serves two purposes
  - encryption nonce incremented by Bob for each message
  - replay/omission prevention checked by Alice per message
- This protocol: no freshness and tolerates no message loss

But how do we build a secure stream cipher or MAC function?

Rijndael/AES

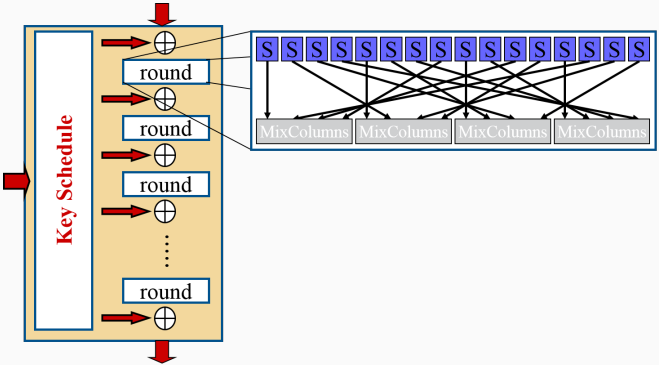


AES is a block cipher



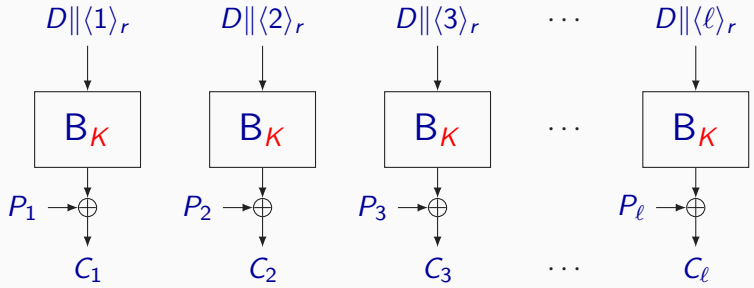
- B_K: encryption of a 16-byte plaintext P to an 16-byte ciphertext C
- Computation of $C = \mathrm{B}_K(P)$ of $P = \mathrm{B}_K^{-1}(C)$ must be
  - efficient for someone who knows the key K
  - . . . infeasible otherwise: B_K *must look like a random permutation*
- With some tinkering this can be made into a stream cipher and a MAC function: *modes of use*

AES in a nutshell



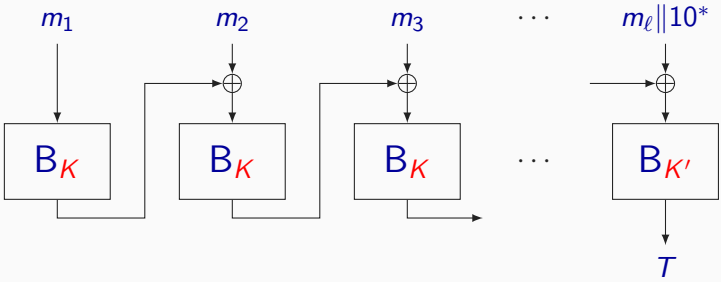
- 10 rounds for 128-bit key, 12 for 192-bit key and 14 for 256-bit key
- Selling point: table-lookup implementatie costs only 16 TLU en XORs per round.
- Best attack breaks reduced-round version of 7 rounds (out of 10)

Stream encryption with a block cipher: Counter mode



- You can prove this is reasonably secure if B_K behaves like a random permutation
- The diversifier D must be a nonce: unique per message
- Security breaks down after $2^{n/2}$ blocks: *birthday bound*

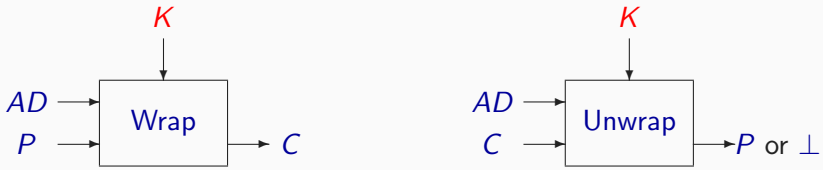
MAC function with a block cipher: CBC-MAC(-like)



- You can prove this is reasonably secure if B_K behaves like a random permutation
- Also here security breaks at the birthday bound: after $2^{n/2}$ blocks

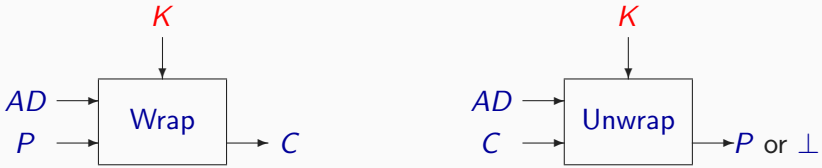
Authenticated encryption

Authenticated encryption (modern definition)



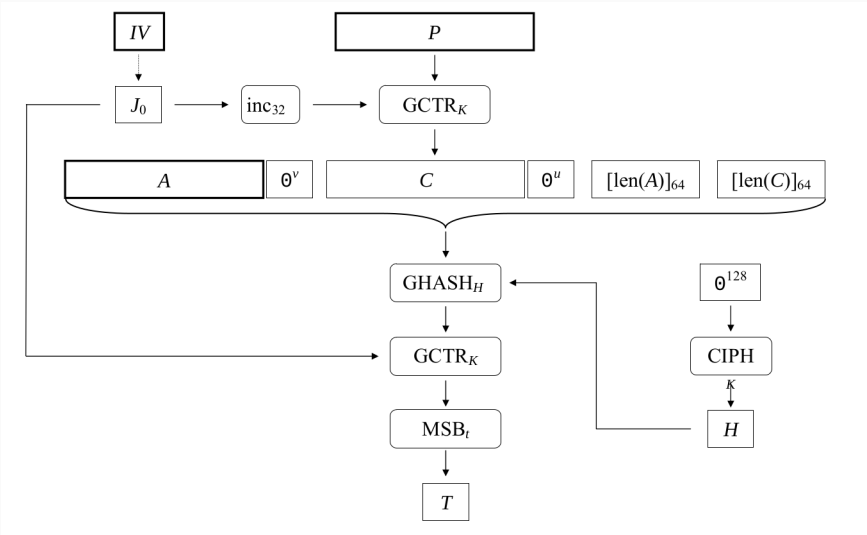
- Wrap: encryption of (var. length) plaintext P to ciphertext C under parameters:
  - (variable-length) associated data AD
  - secret key K
- Unwrap: decryption of C to P includes *authentication*
  - returns error $\perp$ if C is not a valid cryptogram
  - C contains redundancy: $|C| > |P|$
- Wrap and unwrap are deterministic

Authenticated encryption: security goal



- A scheme is secure if it is hard to distinguish from an ideal scheme
- Ideal scheme: the Jammin' cipher [Bacuieti et al., Asiacrypt '22]
  - ciphertexts C are random and independent per input (K, AD, P)
  - unwrap rejects all invalid ciphertexts C
- Limitation: encryption is deterministic
  - leakage: equal inputs (AD, P) give equal ciphertexts C
  - countermeasure: include nonce in AD
- Distinguishability may be conditional on AD uniqueness

Dominant AE standard: AES in Galois Counter Mode (GCM) [NIST SP800-38D]



Issues of AES-GCM

Combination of counter mode AES with Carter-Wegman polynomial MAC over $\mathbb{F}_{2^{128}}$

- In case the diversifier (called IV) repeats:
  - difference of plaintexts leaks
  - authentication key H leaks, so forgery becomes easy
- Diversifier is limited to 96 bits
- Taking a short tag may eventually leak authentication key H
- AES software implementations are vulnerable to **cache attacks**!
  - efficient AES code makes use of table-lookups (TLU)
  - presence/absence of table in cache makes TLU variable-time
  - can be used to recover the key [Page 2002], etc. etc ...

Later Intel and others added AES and GHASH instructions in their CPUs

ChaCha20-Poly1305 by Dan Bernstein

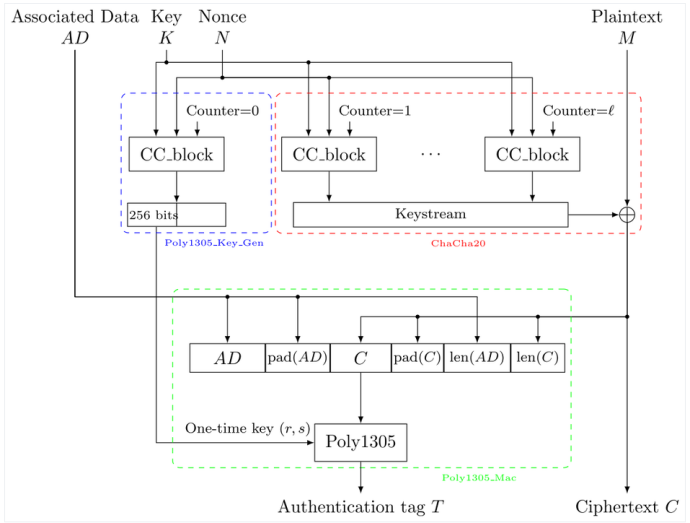


figure by Morz25

ChaCha20 explained

Initial state:

C	C	C	C
K	K	K	K
K	K	K	K
Ctr	D	D	D

Quarter round:

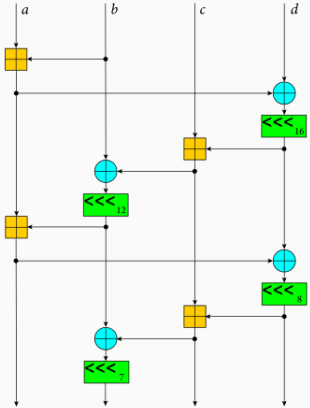


figure T. Arcieri

Odd round:

a	a	a	a
b	b	b	b
c	c	c	c
d	d	d	d

Even round:

a	a	a	a
b	b	b	b
c	c	c	c
d	d	d	d

Best attack breaks reduced-round version of 5/6 rounds (out of 20)

On ChaCha20-Poly1305

- No table-lookups so not vulnerable to cache attacks
  - ChaCha20 is an Addition-Rotation-XOR (ARX) cipher
  - security is based on the mess these operations create when combined
  - very efficient in software, less suited for hardware implementation
  - hard to protect against side channel attacks exploiting measurement of power consumption or electromagnetic emanations
- Poly1305 is a polynomial MAC function in $\mathbb{F}_p$ with $p = 2^{130} - 5$
- In case the diversifier (called N) repeats:
  - difference of plaintexts leaks
  - authentication key r, s leaks, so forgery with same N becomes easy
- Diversifier is limited to 96 bits

ChaCha20-Poly1305 is embraced by Big Tech and the cool crowd!

Deck functions and modes

Deck function based crypto

Refactoring of symmetric crypto, with a three-layer approach:

1. Build permutation f
2. Construct a *new kind of primitive* F_K on top of it
   - input and output of variable length (preferably arbitrary length)
   - output of $F_K(X)$ should look as a sequence of random bits
   - assurance: based on public scrutiny by cryptanalysts
3. Build modes on top of this function that are proven secure if F_K is secure

Definition of a deck function

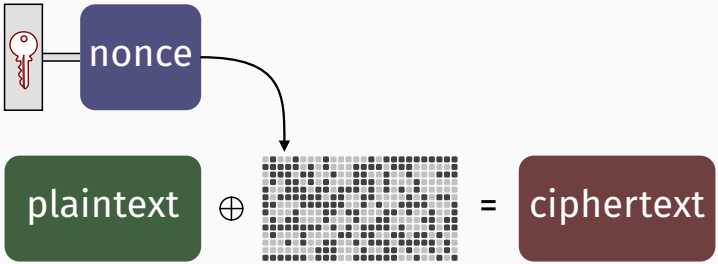
A deck function F_K

$$Z = 0^n + F_K\left(X^{(1)}; \ldots; X^{(m)}\right) \ll q$$

- Input: sequence of strings $X^{(1)}; \ldots; X^{(m)}$
- Output: potentially infinite output
  - **pseudo-random function of the input**
  - taking n bits starting from offset q

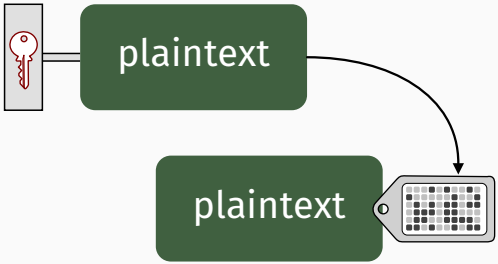
doubly extendable cryptographic keyed function

Stream encryption: short input, long output



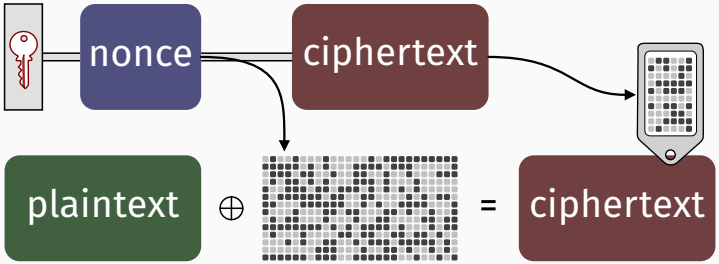
$$C \leftarrow P + F_K(N)$$

MAC computation: long input, short output



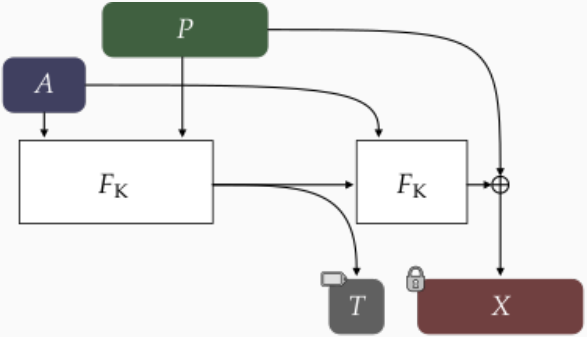
$$T \leftarrow 0^t + F_K(P)$$

Nonce-based authenticated encryption



$$X \leftarrow P + F_K(AD)$$
$$T \leftarrow 0^t + F_K(AD; X)$$
$$C \leftarrow X || T$$

AE without nonce requirement: variant of SIV [Rogaway, Shrimpton Eurocrypt '06]



$$T \leftarrow 0^t + F_K(AD; P||0)$$

$$X \leftarrow P + F_K(AD; T||1)$$

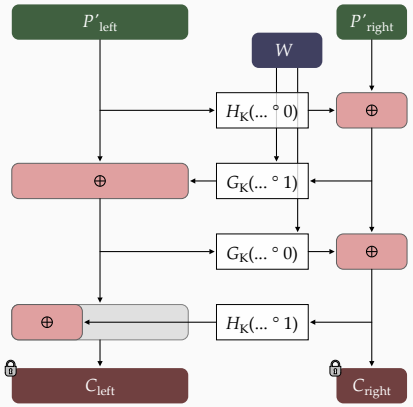
$$C \leftarrow X||T$$

Wide block cipher with double-decker, as Deck-WBC in [KT, eprint 2016/1188]

Encipher $P \in \mathbb{Z}_2^*$ with K and tweak $W \in \mathbb{Z}_2^*$

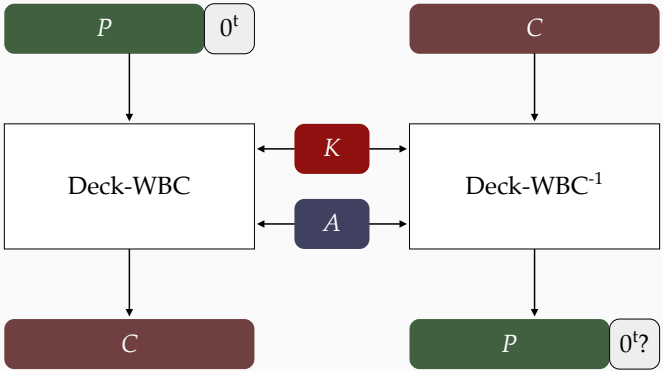
$$
\begin{array}{llll}
(L, R) & \leftarrow & & \mathrm{split}(P) \\
R_0 & \leftarrow R_0 & + & H_K(L||0) \\
L & \leftarrow L & + & G_K(W; R||1) \\
R & \leftarrow R & + & G_K(W; L||0) \\
L_0 & \leftarrow L_0 & + & H_K(R||1) \\
C & \leftarrow L & || & R
\end{array}
$$

return ciphertext C of length $|P|$



Security bound proven in [Gunsing et al, ToSC 2019/4]

Deck-WBC-AE: robust AE [Hoang, Krovetz and Rogaway, Eurocrypt '15]



. . . and much more

- Support for sessions
  - allow intermediate tags for *online* operation
  - allows integrated authentication of bidirectional communication
- Trade-off of efficiency vs robustness against misuse
- Support for *committing* authenticated encryption
- . . . and all their combinations

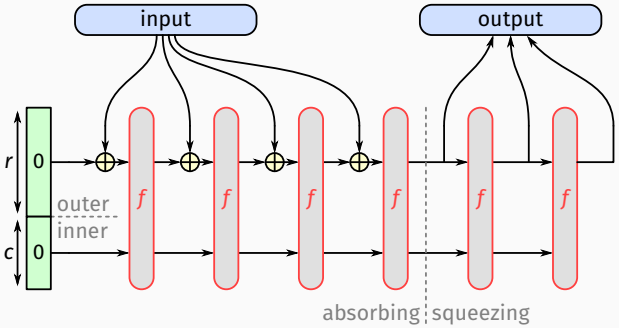
See

"Jammin' on the deck" [Bacuieti et al., Asiacrypt '22]

"Committing authenticated encryption based on SHAKE" [Eprint '23/1494]

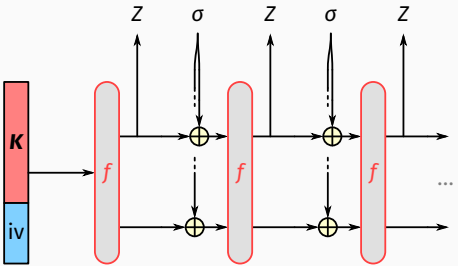
Building deck functions, serially

Sponge [Keccak Team, Ecrypt 2008]



Taking K as first part of input gives something close to a deck function

Keyed duplex object



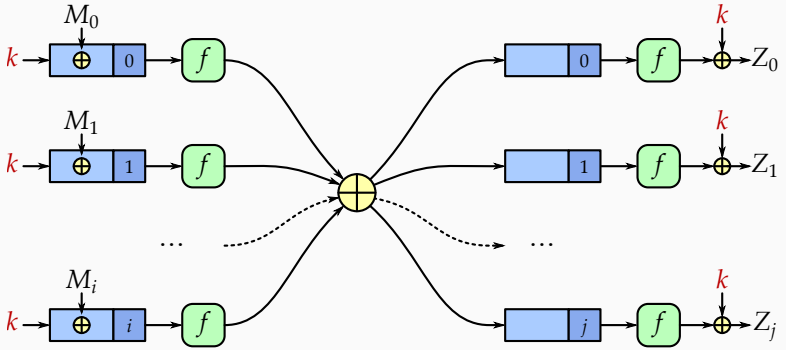
E.g., Strobe [Hamburg 2017], Xoodyak, Subterranean, ASCON [NIST lwc]

- Deck function with length-restricted input and output strings
- Restrictions can be circumvented with a thin input-encoding layer on top
- Cryptanalysis: plug in (round-reduced) permutation and try to break

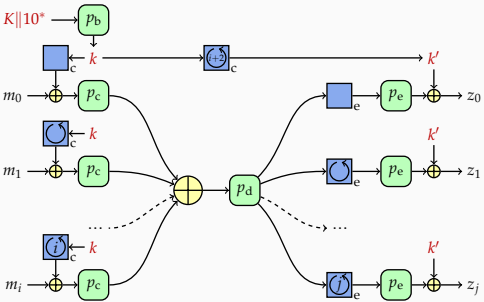
Parallel deck functions

How to build a parallelizable deck function?

Farfalle: early attempt [KT 2014-2016]



Farfalle as published in [Keccak Team + Seth Hoffert, ToSC 2018]



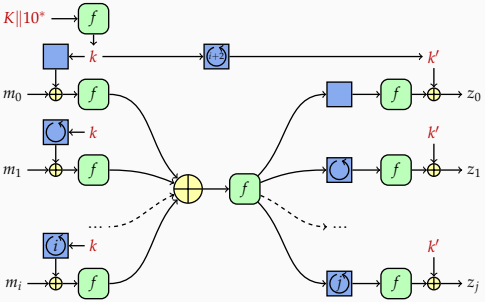
- Derivation of mask k from user key K using p_b
- Linear input mask rolling and p_c *to prevent collisions in state at input of* p_d
- Non-linear state rolling, p_e and mask against state retrieval from output
- Input-output attacks have to deal with $p_e \circ p_d \circ p_c$

Xoodoo and Xoofff

The permutation Xoodoo [Keccak team with Seth Hoffert, ToSC 2018]

- Inspired by 384-bit permutation Gimli [Bernstein et al., CHES 2017]
  - compact on low-end: fits registers of ARM Cortex M3/M4
  - fast on high-end: suitable for SIMD
  - not suited for Farfalle
- Xoodoo
  - 384-bit permutation Keccak *philosophy ported to Gimli shape*
  - main purpose: usage in Farfalle: Xoofff
  - efficient on wide range of platforms

Xoodoo + Farfalle = Xoofff



- $f = \textsc{Xoodoo}$[6]
- Input mask rolling with LFSR, state rolling with NLFSR

Xoofff performance

	ARM Cortex		Intel		
	M0	**M3**	**Skylake**	**SkylakeX**	
mask derivation	1985	781	168	74	cycles
less than 48 bytes	5658	2568	504	358	cycles
MAC computation use case:					
long inputs	26.0	8.8	0.90	0.40	cycles/byte
Stream encryption use case:					
long outputs	25.1	8.1	0.94	0.51	cycles/byte
AES-128 counter mode	121.4	33.2	0.65	0.65	cycles/byte

Conclusions

- There are many AE schemes
- In classical schemes
  - encryption usually done with block ciphers
  - tag computation usually done with polynomial hashes
  - AES-GCM and ChaCha20-Poly1305 dominate
- Secure deck functions are very powerful primitives
  - nonce-based (session) AE
  - SIV-based (session) AE
  - Wide block encryption
- Deck functions can be built from permutations
  - compact: keyed duplex
  - computationally efficient: Farfalle
- Using Xoodoo already gives very competitive deck function Xoofff

Thanks for your attention!