Generic attacks using random functions statistics

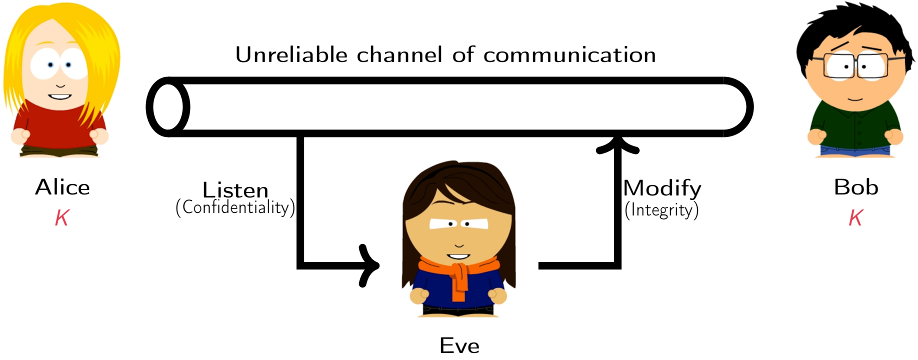
Rachelle Heim Boissier

Université Catholique de Louvain

March, 2025

Symmetric cryptology

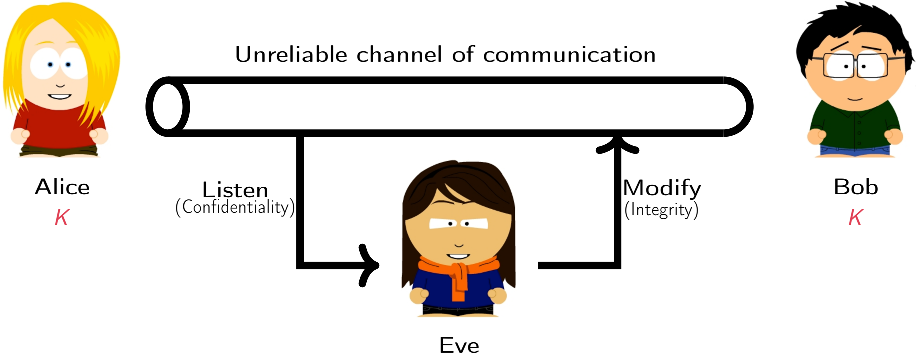
Symmetric cryptology studies algorithms allowing two entities that share a common secret, the key K, to communicate in a secure manner*



Symmetric cryptology

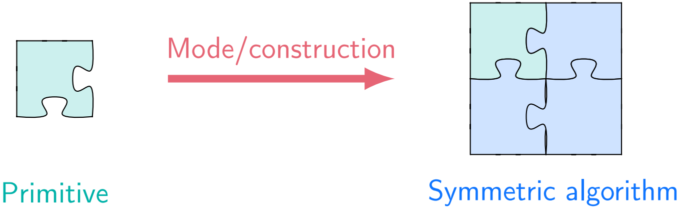
Symmetric cryptology studies algorithms allowing two entities that share a common secret, the key K, to communicate in a secure manner*

*… as well as some 'keyless' algorithms such as hash functions.



Building symmetric algorithms

Cryptography relies on building blocks called *primitives* used within *modes of operation* or *constructions* to build more complex algorithms.



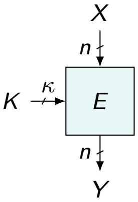
- The notion of primitive is *relative*.
- Most primitives do not provide a standalone cryptographic mechanism on their own.

Primitives

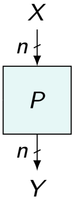
- A block cipher of key size κ bits and block size n bits is a function

$$E : \begin{array}{ccc} \mathbb{F}_2^\kappa \times \mathbb{F}_2^n & \longrightarrow & \mathbb{F}_2^n \\ (K, X) & \longmapsto & E(K, X) \end{array}$$

such that for any key K, $E_K(\cdot) := E(K, \cdot)$ is a permutation of $\mathbb{F}_2^n$.



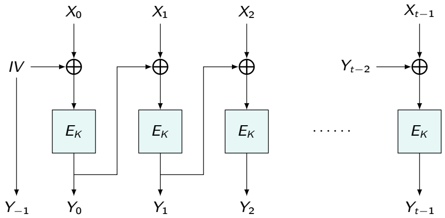
- A public permutation P over $\mathbb{F}_2^n$ does not depend on a key.



Modes/constructions

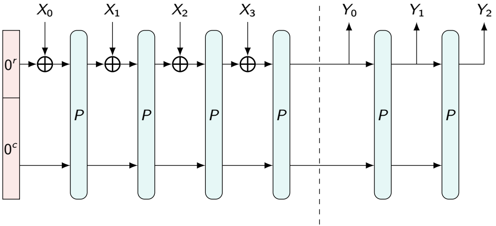
- **Block cipher-based**

Ex: the encryption mode CBC.



- **Permutation-based**

Ex: the sponge construction for hashing.



Security in cryptography (1/2)

Two main approaches:

- Provable security: reducing the security of a scheme to some ‘reasonable’ assumption.
  - How do we assess the reasonability of our assumption?
- Cryptanalysis: security analysis effort.
  - If the international cryptographic community cannot break it, then, hopefully, noone else can.
  - International standardisation competitions organised by the NIST.
  - The cryptanalysis effort be global, continuous and comprehensive.

Security in cryptography (2/2)

Primitive security

- can only be guaranteed through cryptanalysis.
- More or less well-defined security assumptions.

Mode/construction security

- Proved under the assumption that the primitive is secure.
- Proofs provide a partial information on the security level.
- Cryptanalysis, and in particular generic attacks, provides a complementary point of view.

A generic attack assumes an ideal behaviour of the underlying primitive.

Elementary ex: generic key recovery attack on E given X and $Y = E_K(X)$.

- Exhaustively try the 2^{κ} possible secret keys.

This talk

- Symmetric cryptanalysis.
- Generic attacks against a variety of iterated constructions:
  - Hash functions;
  - Message Authentication Codes (MAC) modes;
  - Authenticated encryption (AE) modes.
- Our main tool: random functions graphs statistics.

Outline

1. Random function statistics
2. Memory-negligible collision search
3. State recovery attack against `HMAC`
4. Generic attack against AE modes
5. Conclusion

Random functions

$\mathfrak{F}_N$ is the set of functions which map a finite set of size $N \in \mathbb{N}^*$ to itself.

Our main focus:

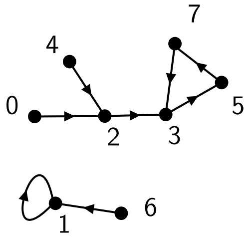
The graph of f, denoted by $G(f)$, is a directed graph such that an edge goes from node i to node j if and only if $f(i) = j$.

Properties and statistics of functional graphs are used in generic attacks.

Functional graphs: an example

The graph of f, denoted by $G(f)$, is a directed graph such that an edge goes from node i to node j if and only if $f(i) = j$.

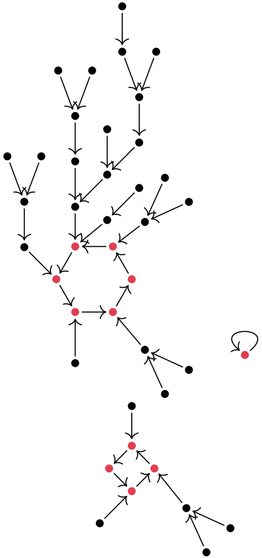
$$f \quad : \quad [\![0;7]\!] \longrightarrow [\![0;7]\!]$$
$$\left\{\begin{array}{lll} 0 & \longmapsto & 2 \\ 1 & \longmapsto & 1 \\ 2 & \longmapsto & 3 \\ 3 & \longmapsto & 5 \\ 4 & \longmapsto & 2 \\ 5 & \longmapsto & 7 \\ 6 & \longmapsto & 1 \\ 7 & \longmapsto & 3 \end{array}\right.$$



Functional graphs (1)

Definitions.

- The graph of f can be seen as a set of connected components.
- Each connected component has a unique cycle.
- Each cyclic node is the root of a tree.



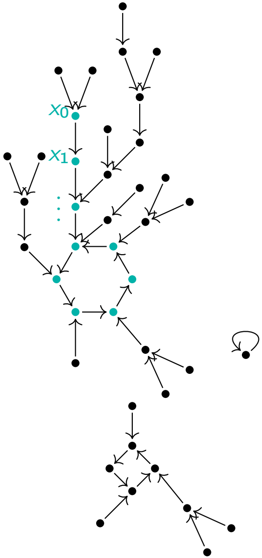
Statistics (e.g. [FO89]).

- Expected size of f's largest component: $0.76N$
- Expected size of f's largest tree: $0.48N$

Functional graphs (2)

For any $x_0 \in G(f)$

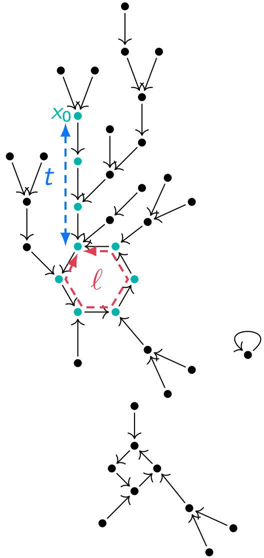
- $(x_i := f^i(x_0))_{i \in \mathbb{N}}$ is eventually periodic.
- $(x_i)_{i \in \mathbb{N}}$ graphically corresponds to a path linked to a cycle.



Functional graphs (2)

For any $x_0 \in G(f)$

- $(x_i := f^i(x_0))_{i\in\mathbb{N}}$ is eventually periodic.
- $(x_i)_{i\in\mathbb{N}}$ graphically corresponds to a path linked to a cycle.



Definitions.

- Tail length $t(x_0)$: smallest i s.t. x_i is in the cycle.
- Cycle length $\ell(x_0)$: number of nodes in the cycle.

Statistics. For x a random node:

- Expected value of its tail length $t(x)$: $\sqrt{\pi N/8}$.
- Expected value of its cycle length $\ell(x)$: $\sqrt{\pi N/8}$.

Outline

Cryptographic hash functions

Definition. A cryptographic hash function is a function $H : \mathbb{F}_2^* \to \mathbb{F}_2^n$ such that

- Preimage resistance. Given $D \in \mathbb{F}_2^n$, it is difficult to find $M \in \mathbb{F}_2^*$ s.t. $H(M) = D$;
- Second preimage resistance. Given M, it is difficult to find $M' \neq M$ s.t. $H(M') = H(M)$;
- Collision resistance. It is difficult to find (M, M'), $M \neq M'$ such that $H(M) = H(M')$.

Cryptographic hash functions

Definition. A cryptographic hash function is a function $H : \mathbb{F}_2^* \to \mathbb{F}_2^n$ such that

- Preimage resistance. Given $D \in \mathbb{F}_2^n$, it is difficult to find $M \in \mathbb{F}_2^*$ s.t. $H(M) = D$;
- Second preimage resistance. Given M, it is difficult to find $M' \neq M$ s.t. $H(M') = H(M)$;
- Collision resistance. It is difficult to find (M, M'), $M \neq M'$ such that $H(M) = H(M')$.

Generic collision attack: Compute $H(M)$ for $O(2^{n/2})$ messages M, store M at the address $H(M)$.

Cryptographic hash functions

Definition. A cryptographic hash function is a function $H : \mathbb{F}_2^* \to \mathbb{F}_2^n$ such that

- Preimage resistance. Given $D \in \mathbb{F}_2^n$, it is difficult to find $M \in \mathbb{F}_2^*$ s.t. $H(M) = D$;
- Second preimage resistance. Given M, it is difficult to find $M' \neq M$ s.t. $H(M') = H(M)$;
- Collision resistance. It is difficult to find (M, M'), $M \neq M'$ such that $H(M) = H(M')$.

Generic collision attack: Compute $H(M)$ for $O(2^{n/2})$ messages M, store M at the address $H(M)$.

Memory complexity is also a $O(2^{n/2})$.

Cryptographic hash functions

Definition. A cryptographic hash function is a function $H : \mathbb{F}_2^* \to \mathbb{F}_2^n$ such that

- Preimage resistance. Given $D \in \mathbb{F}_2^n$, it is difficult to find $M \in \mathbb{F}_2^*$ s.t. $H(M) = D$;
- Second preimage resistance. Given M, it is difficult to find $M' \neq M$ s.t. $H(M') = H(M)$;
- Collision resistance. It is difficult to find (M, M'), $M \neq M'$ such that $H(M) = H(M')$.

Generic collision attack: Compute $H(M)$ for $O(2^{n/2})$ messages M, store M at the address $H(M)$.

Memory complexity is also a $O(2^{n/2})$.

Solution: a generic memory-negligible collision attack using functional graphs.

A memory-negligible collision attack on H

Let $f \in \mathfrak{F}_{2^n}$ be defined as

$$\begin{array}{rccc} f & : & \mathbb{F}_2^n & \longrightarrow & \mathbb{F}_2^n \\ & & x & \longmapsto & H(x)\,. \end{array}$$

Step 1. A cycle finding algorithm allows to recover a cyclic node x_c

- in time $O(2^{n/2})$;
- using a negligible amount of memory.

Step 2. Using x_c, one can

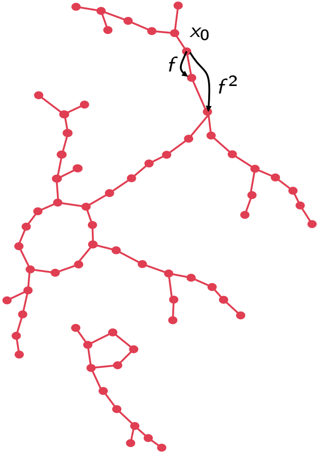
- recover the cycle length $\ell(x_c)$,
- find a collision on f, and thus on H,

in time $O(2^{n/2})$ and with negligible memory.

Ex: Floyd's cycle finding algorithm

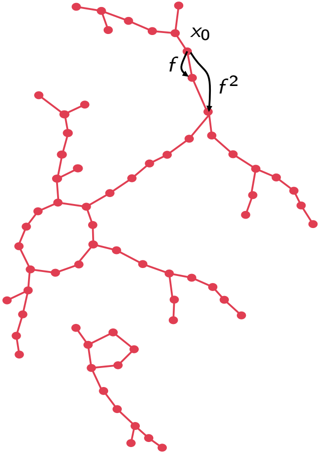
parameters: $f \in \mathfrak{F}_{2^n}$

$x_0 \leftarrow_R \mathbb{F}_2^n$
$turtle, hare \leftarrow x_0, x_0$
for $i = 1$ to $2^n - 1$ **do**
    $turtle \leftarrow f(turtle)$
    $hare \leftarrow f^2(hare)$
    if $turtle = hare$ **then**
        return $turtle$
    end if
end for



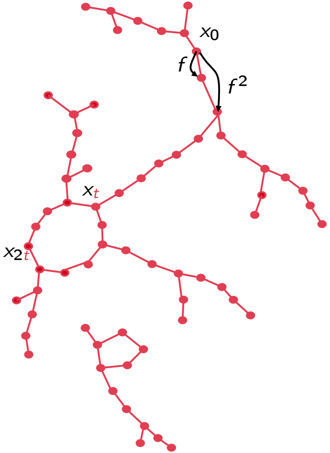
Ex: Floyd's cycle finding algorithm

The tail length $t = t(x_0)$ is the smallest j s.t. x_j in the cycle.



Ex: Floyd's cycle finding algorithm

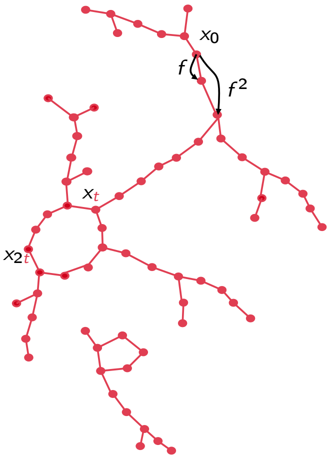
The tail length $t = t(x_0)$ is the smallest j s.t. x_j in the cycle.



Ex: Floyd's cycle finding algorithm

The tail length $t = t(x_0)$ is the smallest j s.t. x_j in the cycle.

Let $d_t = dist(x_t, x_{2t})$.



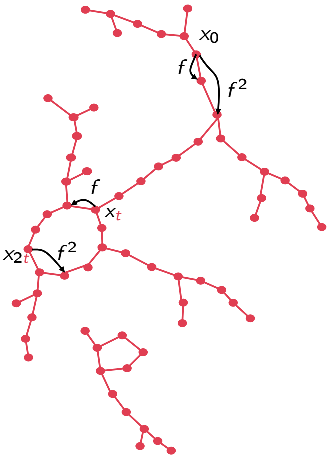
Ex: Floyd's cycle finding algorithm

The tail length $t = t(x_0)$ is the smallest j s.t. x_j in the cycle.

Let $d_t = dist(x_t, x_{2t})$.

Then

$$dist(f(x_t), f^2(x_{2t})) = d_t + 1 \mod \ell\,.$$



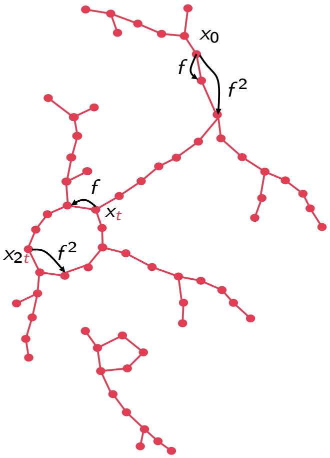
Ex: Floyd's cycle finding algorithm

The tail length $t = t(x_0)$ is the smallest j s.t. x_j in the cycle.

Let $d_t = dist(x_t, x_{2t})$.

Then

$$dist(x_{t+k}, x_{2(t+k)}) = d_t + k \mod \ell\,.$$



Ex: Floyd's cycle finding algorithm

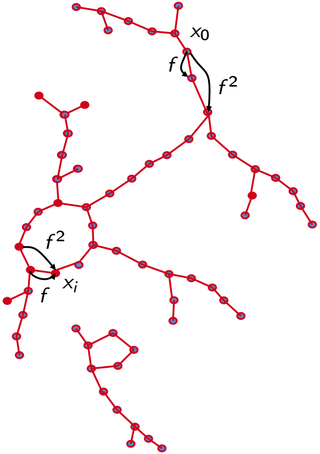
The tail length $t = t(x_0)$ is the smallest j s.t. x_j in the cycle.

Let $d_t = dist(x_t, x_{2t})$.

Then

$$dist(x_{t+k}, x_{2(t+k)}) = d_t + k \mod \ell .$$

After at most $t + \ell$ tries, the algorithm finds i s.t. $x_i = x_{2i}$, and x_i is in the cycle.



d

Ex: Floyd's cycle finding algorithm

The tail length $t = t(x_0)$ is the smallest j s.t. x_j in the cycle.

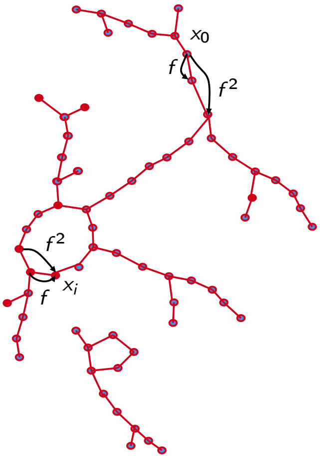
Let $d_t = dist(x_t, x_{2t})$.

Then

$$dist(x_{t+k}, x_{2(t+k)}) = d_t + k \mod \ell\,.$$

After at most $t + \ell$ tries, the algorithm finds i s.t. $x_i = x_{2i}$, and x_i is in the cycle.

If f behaves like a random function, if x_0 is drawn at random, we expect $t = \ell = \sqrt{\pi/8} \cdot 2^{n/2}$.



d

Ex: Floyd's cycle finding algorithm

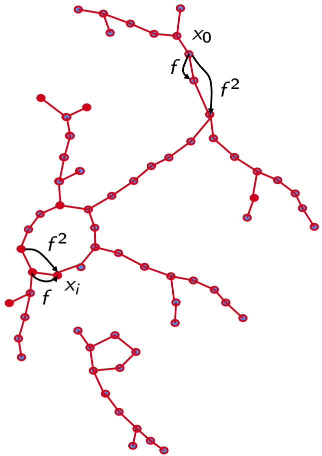
The tail length $t = t(x_0)$ is the smallest j s.t. x_j in the cycle.

Let $d_t = dist(x_t, x_{2t})$.

Then

$$dist(x_{t+k}, x_{2(t+k)}) = d_t + k \mod \ell\,.$$

After at most $t + \ell$ tries, the algorithm finds i s.t. $x_i = x_{2i}$, and x_i is in the cycle.



If f behaves like a random function, if x_0 is drawn at random, we expect $t = \ell = \sqrt{\pi/8} \cdot 2^{n/2}$.

Floyd's time complexity: $O(2^{n/2})$ evaluations of f, memory complexity is negligible.

d

A memory-negligible collision attack on H

Let $f \in \mathfrak{F}_{2^n}$ be defined as

$$\begin{array}{cccccc} f & : & \mathbb{F}_2^n & \longrightarrow & \mathbb{F}_2^n \\ & & x & \longmapsto & H(x)\,. \end{array}$$

Step 1. A cycle finding algorithm allows to recover a cyclic node x_c

- in time $O(2^{n/2})$;
- using a negligible amount of memory.

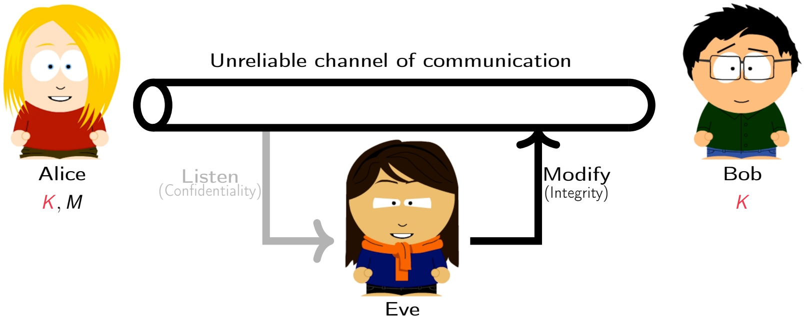
Step 2. Using x_c, one can

- recover the cycle length $\ell(x_c)$,
- find a collision on f, and thus on H,

in time $O(2^{n/2})$ and with negligible memory.

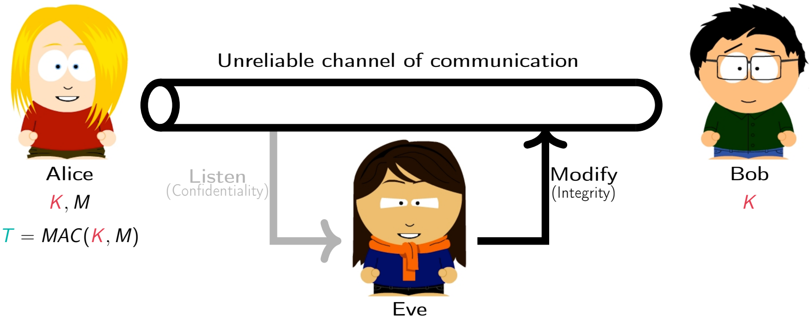
Outline

Message Authentication Code (MAC) algorithms



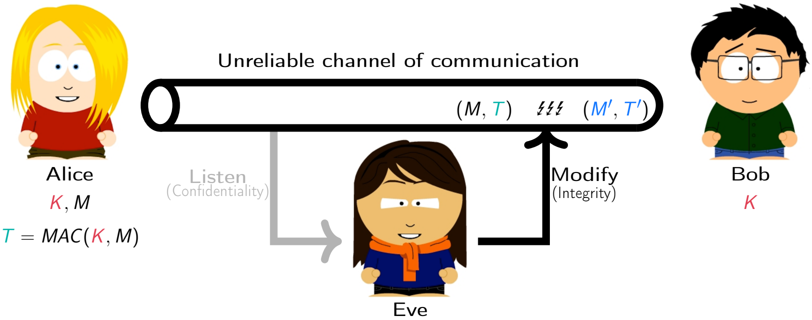
A Message Authentication Code algorithm (MAC) produces a fixed length tag that guarantees the integrity of the message.

Message Authentication Code (MAC) algorithms



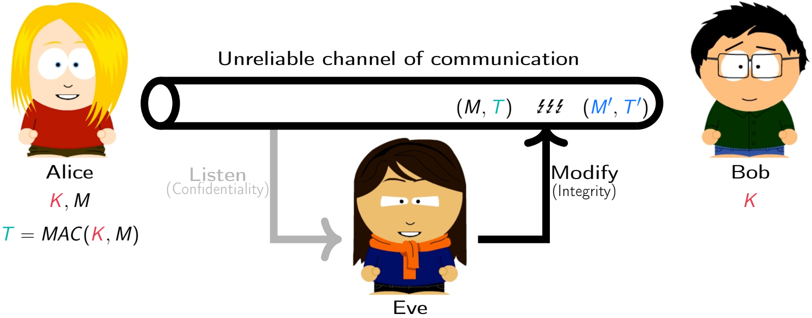
A Message Authentication Code algorithm (MAC) produces a fixed length tag that guarantees the integrity of the message.

Message Authentication Code (MAC) algorithms



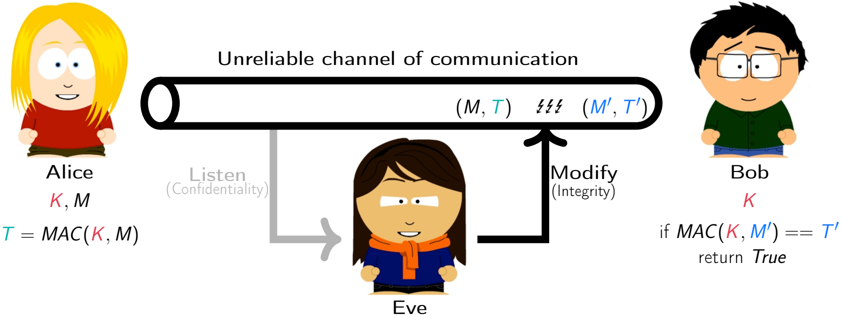
A Message Authentication Code algorithm (MAC) produces a fixed length tag that guarantees the integrity of the message.

Message Authentication Code (MAC) algorithms



A Message Authentication Code algorithm (MAC) produces a fixed length tag that guarantees the integrity of the message.

Message Authentication Code (MAC) algorithms



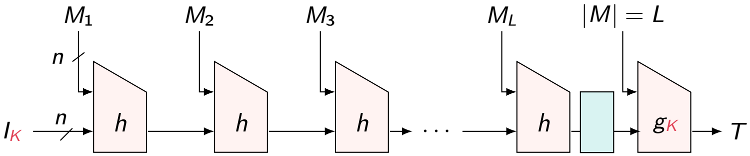
A Message Authentication Code algorithm (MAC) produces a fixed length tag that guarantees the integrity of the message.

Hash-based MACs

Hash functions can be used to build MACs.

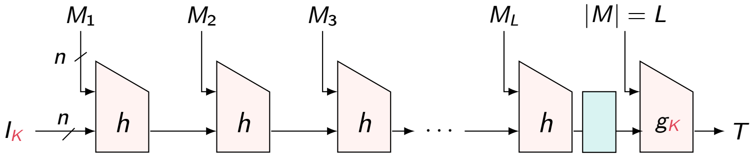
- It is easy to build a secure MAC with an ideal hash function, i.e. a random oracle.
- With a real hash function, it is essential to study generic attacks.
- Several papers analyse the generic security of HMACs.

We present a 2013 state recovery attack by Leurent, Peyrin and Wang on `HMAC` [BCK96].



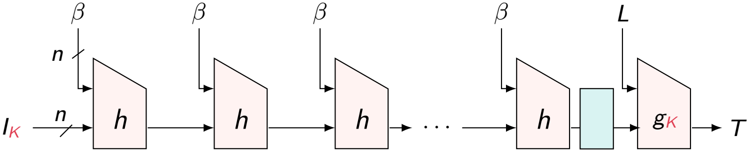
State-recovery attack on HMAC [LPW13]

$$M = M_1 \mid\mid \cdots \mid\mid M_L$$



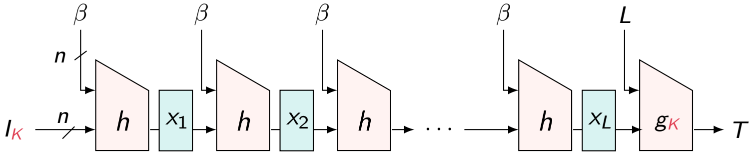
State-recovery attack on HMAC [LPW13]

$$M = \beta \,||\cdots||\beta = \beta^L$$



State-recovery attack on HMAC [LPW13]

$M = \beta \,||\cdots||\beta = \beta^L$

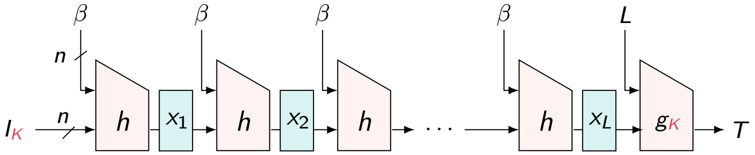


The tag generation iterates the function

$$\begin{aligned} h_\beta : \mathbb{F}_2^n &\longrightarrow \mathbb{F}_2^n \\ x &\longmapsto h(\beta, x)\,. \end{aligned}$$

State-recovery attack on HMAC [LPW13]

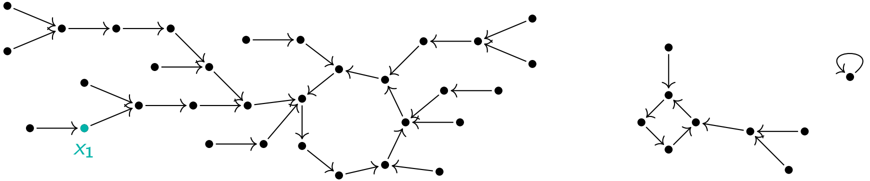
$M = \beta \,||\cdots||\beta = \beta^L$



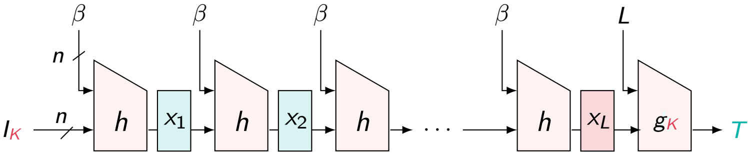
- For a random β, we expect h_β to behave as a function drawn at random in $\mathfrak{F}_{2^n}$.
  - Giant component with about 76% of the nodes.
- We expect x_1 to behave as a point drawn at random in the graph of h_β.
  - With proba 0.76, x_1 is in the giant component.
  - $t(x_1) = \ell(x_1) = \sqrt{\pi/8} \cdot 2^{n/2}$.

State-recovery attack on HMAC [LPW13]

- Setting $L = cst \cdot 2^{n/2}$:

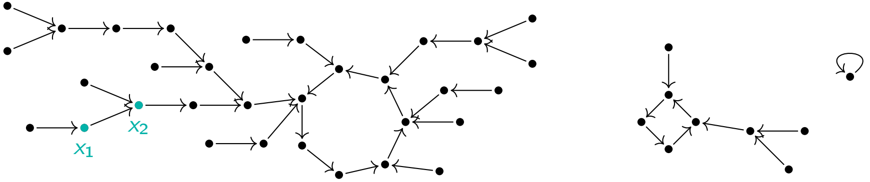


Graph of h_β

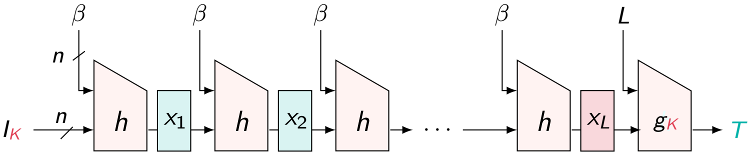


State-recovery attack on HMAC [LPW13]

- Setting $L = cst \cdot 2^{n/2}$:

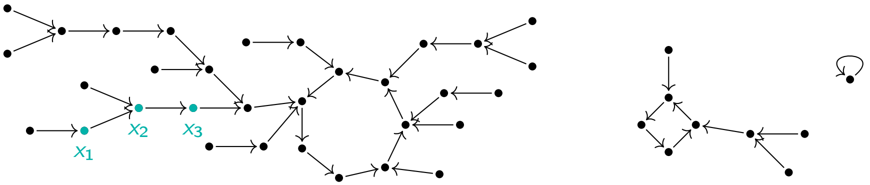


Graph of h_β

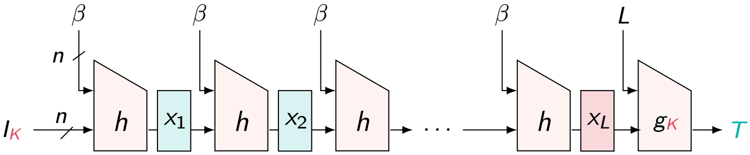


State-recovery attack on HMAC [LPW13]

- Setting $L = cst \cdot 2^{n/2}$:

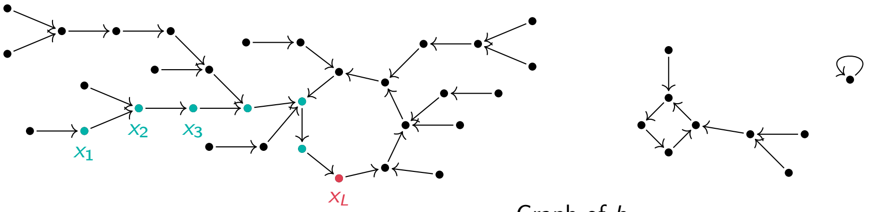


Graph of h_β

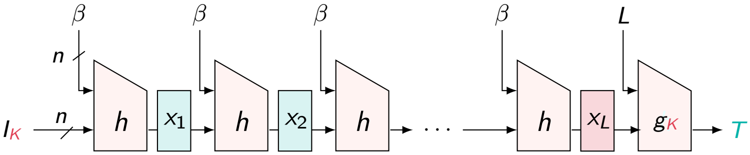


State-recovery attack on HMAC [LPW13]

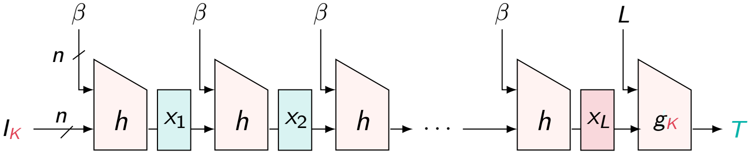
- Setting $L = cst \cdot 2^{n/2}$:



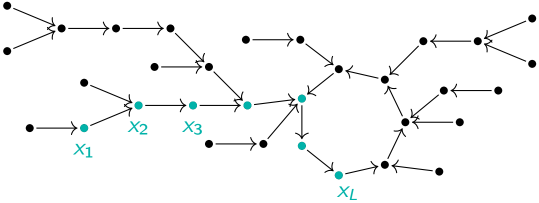
Graph of h_β



Idea 1: Building two messages who reach the same state



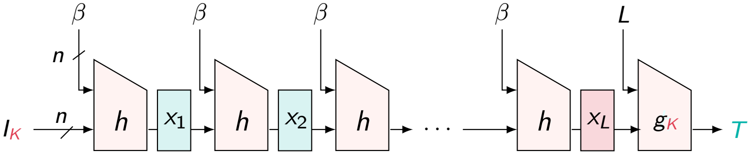
$M_1 = \beta^L$



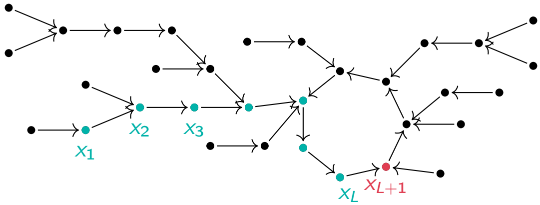
$M_1 = \beta^L$ and $M_2 = \beta^{L+\ell}$ reach the same final state.

Two issues: $\neq$ message lengths + the state is not recovered.

Idea 1: Building two messages who reach the same state



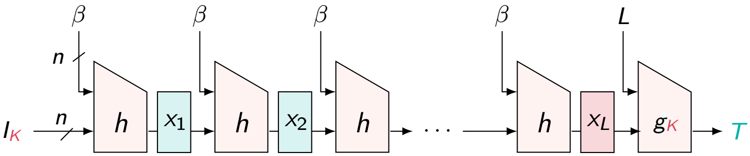
$M = \beta^{L+1}$



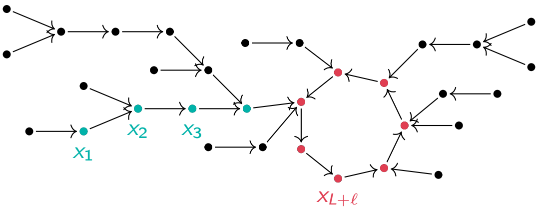
$M_1 = \beta^L$ and $M_2 = \beta^{L+\ell}$ reach the same final state.

Two issues: $\neq$ message lengths + the state is not recovered.

Idea 1: Building two messages who reach the same state



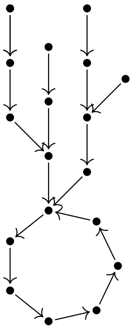
$M_2 = \beta^{L+\ell}$



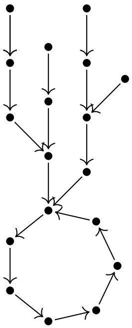
$M_1 = \beta^L$ and $M_2 = \beta^{L+\ell}$ reach the same final state.

Two issues: $\neq$ message lengths + the state is not recovered.

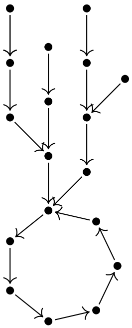
Idea 2: reach the cycle twice



$M_0 = \beta^L || \gamma || \beta^L$



$M_1 = \beta^{L+\ell} || \gamma || \beta^L$

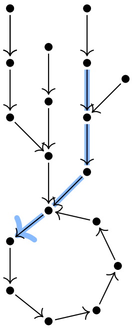


$M_2 = \beta^L || \gamma || \beta^{L+\ell}$

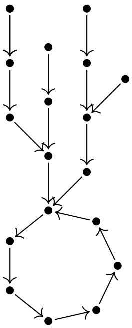
M_1 and M_2 reach the same state with constant probability and $|M_1| = |M_2|$.

Still no state recovery.

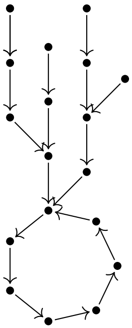
Idea 2: reach the cycle twice



$M_0 = \beta^{L} || \gamma || \beta^{L}$



$M_1 = \beta^{L+\ell} || \gamma || \beta^{L}$

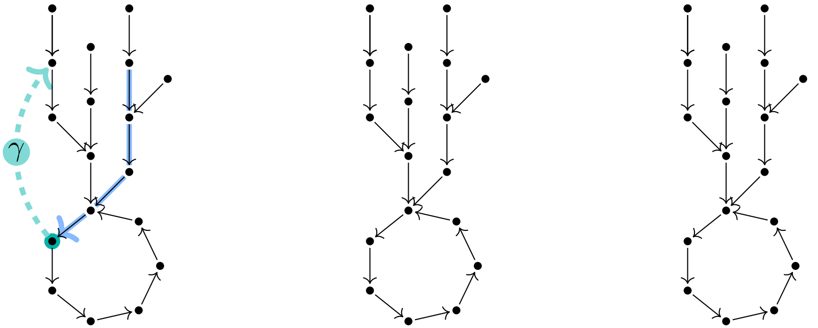


$M_2 = \beta^{L} || \gamma || \beta^{L+\ell}$

M_1 and M_2 reach the same state with constant probability and $|M_1| = |M_2|$.

Still no state recovery.

Idea 2: reach the cycle twice

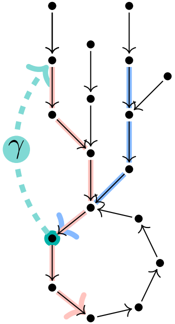


$M_0 = \beta^L || \gamma || \beta^L$ $\quad$ $M_1 = \beta^{L+\ell} || \gamma || \beta^L$ $\quad$ $M_2 = \beta^L || \gamma || \beta^{L+\ell}$

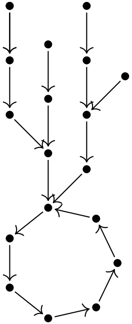
M_1 and M_2 reach the same state with constant probability and $|M_1| = |M_2|$.

Still no state recovery.

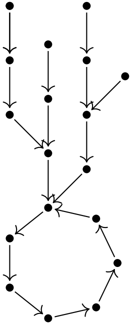
Idea 2: reach the cycle twice



$M_0 = \beta^L || \gamma || \beta^L$



$M_1 = \beta^{L+\ell} || \gamma || \beta^L$

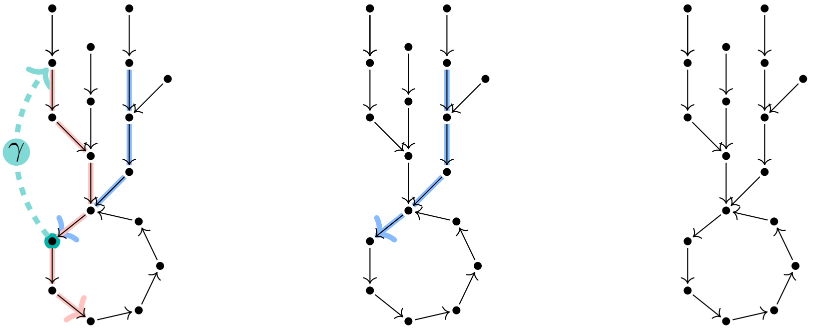


$M_2 = \beta^L || \gamma || \beta^{L+\ell}$

M_1 and M_2 reach the same state with constant probability and $|M_1| = |M_2|$.

Still no state recovery.

Idea 2: reach the cycle twice

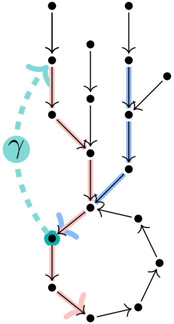


$M_0 = \beta^L || \gamma || \beta^L$ $\quad$ $M_1 = \beta^{L+\ell} || \gamma || \beta^L$ $\quad$ $M_2 = \beta^L || \gamma || \beta^{L+\ell}$

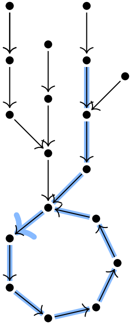
M_1 and M_2 reach the same state with constant probability and $|M_1| = |M_2|$.

Still no state recovery.

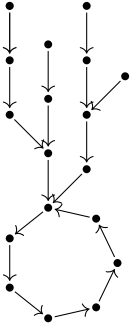
Idea 2: reach the cycle twice



$M_0 = \beta^L || \gamma || \beta^L$



$M_1 = \beta^{L+\ell} || \gamma || \beta^L$

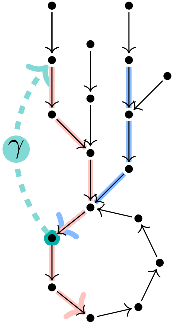


$M_2 = \beta^L || \gamma || \beta^{L+\ell}$

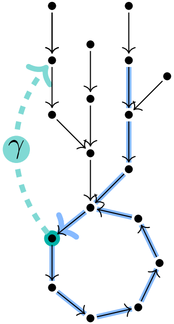
M_1 and M_2 reach the same state with constant probability and $|M_1| = |M_2|$.

Still no state recovery.

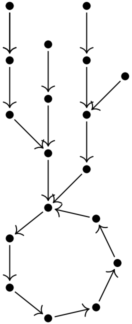
Idea 2: reach the cycle twice



$M_0 = \beta^{L}||\gamma||\beta^{L}$



$M_1 = \beta^{L+\ell}||\gamma||\beta^{L}$

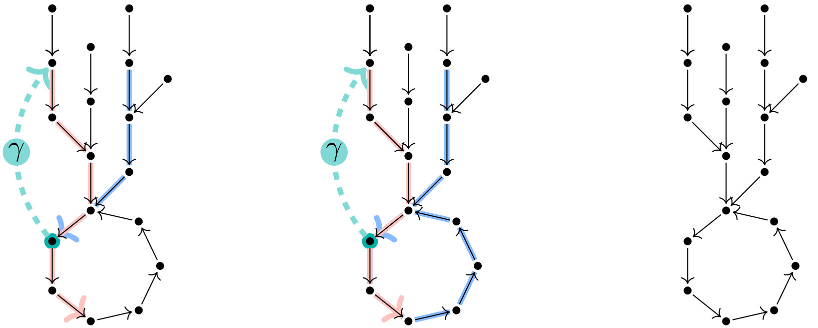


$M_2 = \beta^{L}||\gamma||\beta^{L+\ell}$

M_1 and M_2 reach the same state with constant probability and $|M_1| = |M_2|$.

Still no state recovery.

Idea 2: reach the cycle twice

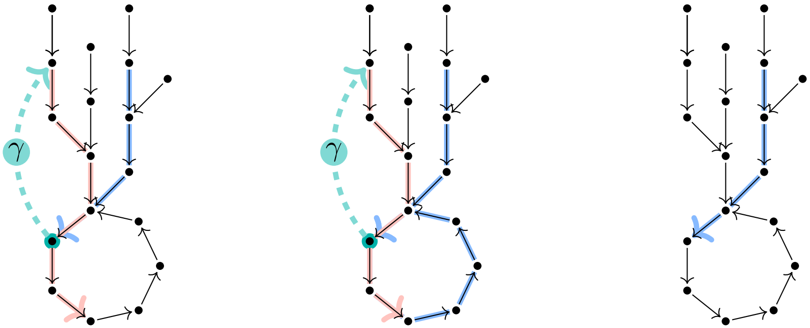


$M_0 = \beta^L || \gamma || \beta^L$ $\quad$ $M_1 = \beta^{L+\ell} || \gamma || \beta^L$ $\quad$ $M_2 = \beta^L || \gamma || \beta^{L+\ell}$

M_1 and M_2 reach the same state with constant probability and $|M_1| = |M_2|$.

Still no state recovery.

Idea 2: reach the cycle twice

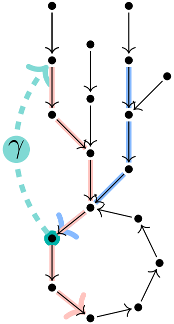


$M_0 = \beta^L || \gamma || \beta^L$ $\quad$ $M_1 = \beta^{L+\ell} || \gamma || \beta^L$ $\quad$ $M_2 = \beta^L || \gamma || \beta^{L+\ell}$

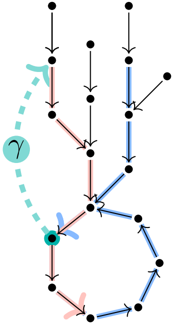
M_1 and M_2 reach the same state with constant probability and $|M_1| = |M_2|$.

Still no state recovery.

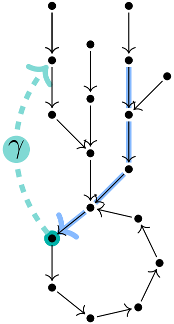
Idea 2: reach the cycle twice



$M_0 = \beta^L || \gamma || \beta^L$



$M_1 = \beta^{L+\ell} || \gamma || \beta^L$

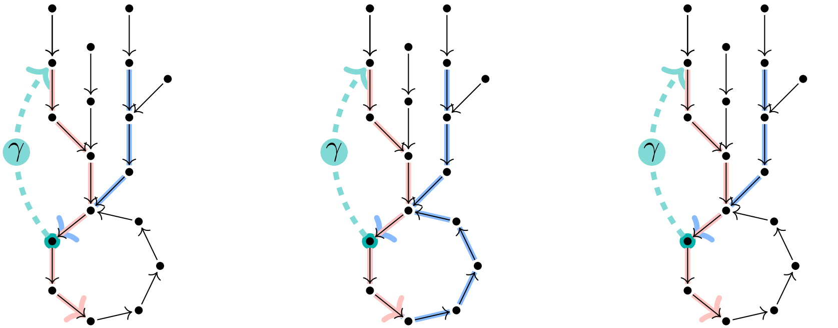


$M_2 = \beta^L || \gamma || \beta^{L+\ell}$

M_1 and M_2 reach the same state with constant probability and $|M_1| = |M_2|$.

Still no state recovery.

Idea 2: reach the cycle twice

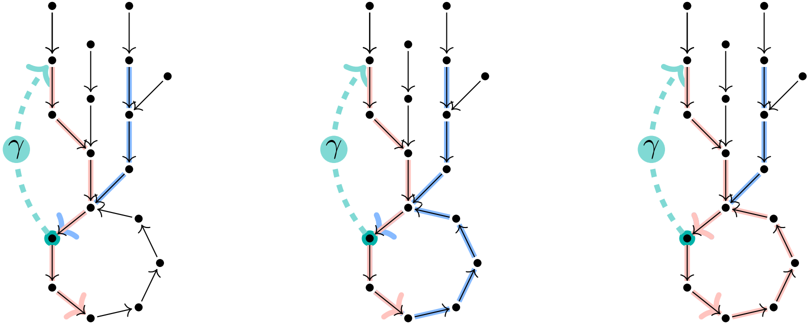


$M_0 = \beta^L || \gamma || \beta^L$ $\qquad$ $M_1 = \beta^{L+\ell} || \gamma || \beta^L$ $\qquad$ $M_2 = \beta^L || \gamma || \beta^{L+\ell}$

M_1 and M_2 reach the same state with constant probability and $|M_1| = |M_2|$.

Still no state recovery.

Idea 2: reach the cycle twice

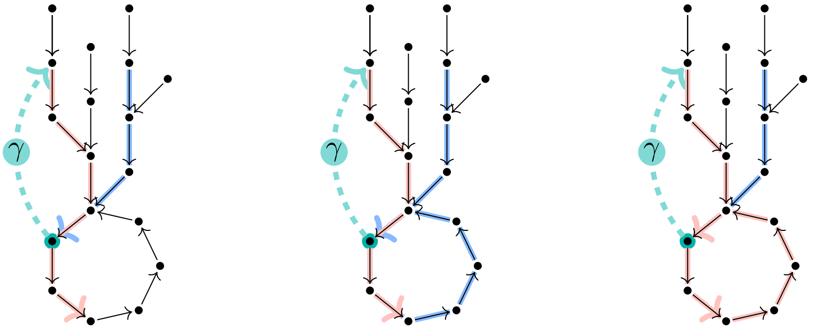


$M_0 = \beta^L || \gamma || \beta^L$ $\qquad$ $M_1 = \beta^{L+\ell} || \gamma || \beta^L$ $\qquad$ $M_2 = \beta^L || \gamma || \beta^{L+\ell}$

M_1 and M_2 reach the same state with constant probability and $|M_1| = |M_2|$.

Still no state recovery.

Idea 2: reach the cycle twice



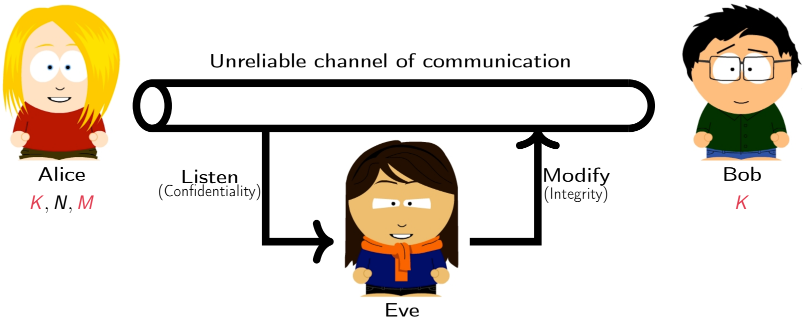
$M_0 = \beta^L || \gamma || \beta^L$ $\quad$ $M_1 = \beta^{L+\ell} || \gamma || \beta^L$ $\quad$ $M_2 = \beta^L || \gamma || \beta^{L+\ell}$

M_1 and M_2 reach the same state with constant probability and $|M_1| = |M_2|$.

Still no state recovery. Idea 3: use the root of the main tree α.

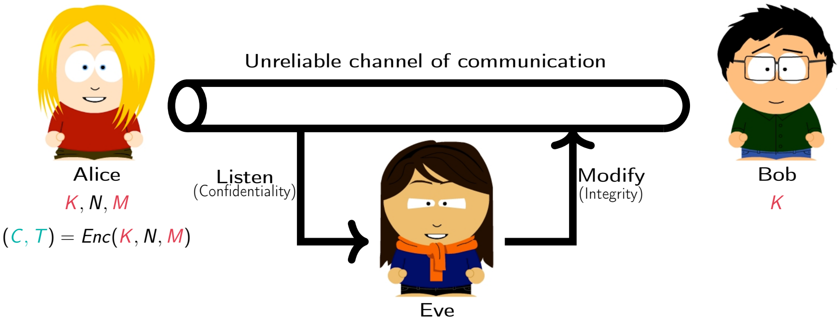
Outline

Authenticated Encryption



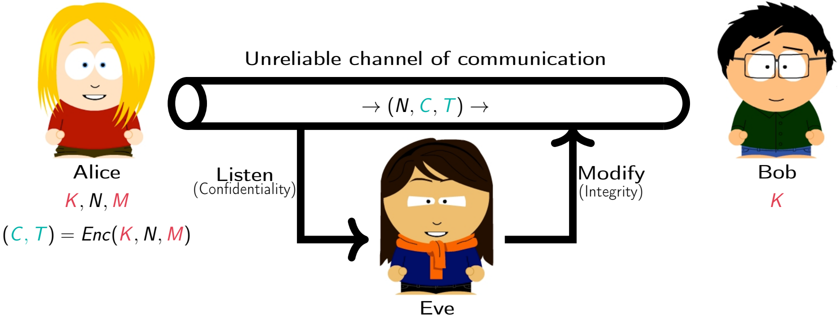
An authenticated encryption scheme ensures both the confidentiality and integrity of communications.

Authenticated Encryption



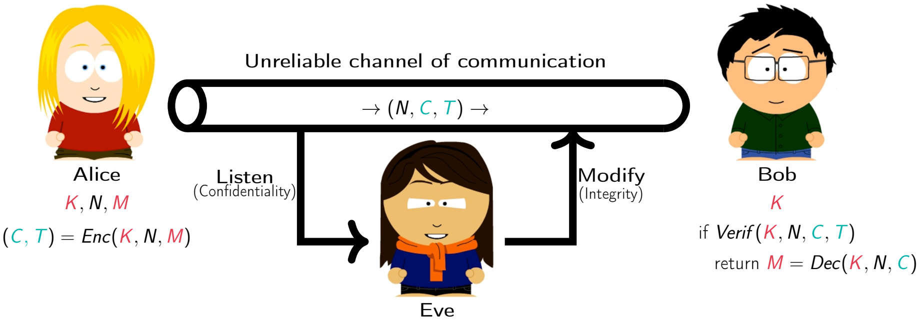
An authenticated encryption scheme ensures both the confidentiality and integrity of communications.

Authenticated Encryption



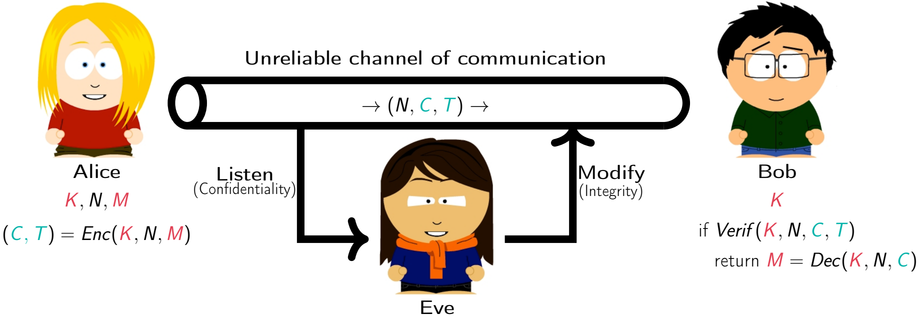
An authenticated encryption scheme ensures both the confidentiality and integrity of communications.

Authenticated Encryption



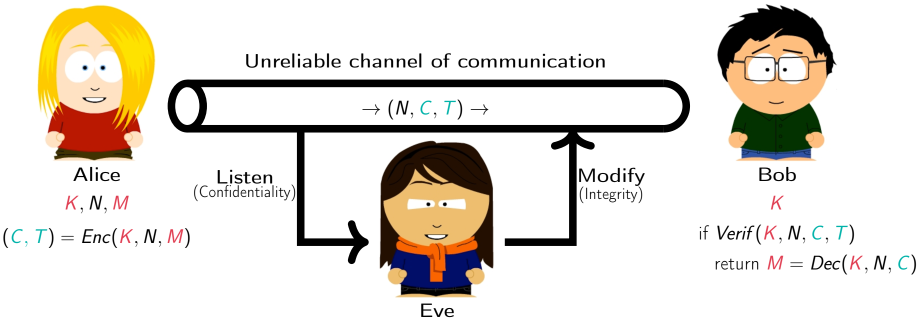
An authenticated encryption scheme ensures both the confidentiality and integrity of communications.

Authenticated Encryption



Forgery attack: find a decryption query (N, C, T) s.t. the tag verification succeeds.

Authenticated Encryption



Forgery attack: find a decryption query (N, C, T) s.t. the tag verification succeeds.

- Assuming a nonce-respecting adversary
- and no release of unverified plaintext.

Duplex-based AE modes

Authenticated Encryption

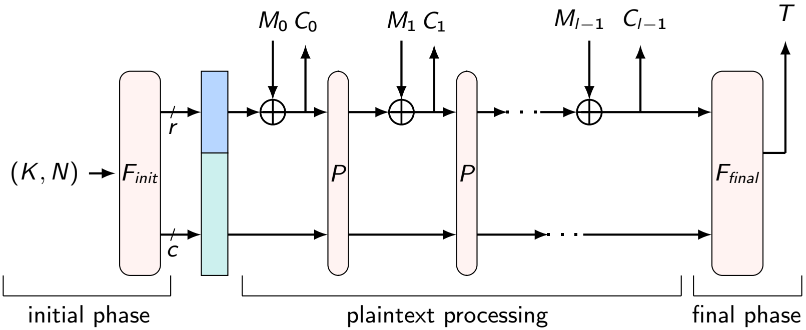
- (Historically) block-cipher based: (tweakable) block cipher + mode
- (More recently) permutation-based: public permutation + keyed mode

Permutation-based modes of operation [BDPVA11]

- Many candidates at the NIST lightweight competition (2018-2023), including the winner ASCON.
- Modes are proven secure when instantiated with a random permutation.
- It is difficult to assess this ‘assumption’ in practice $\rightarrow$ cryptanalysis.

Duplex-based AE modes [BDPVA11,DMV17]

Encryption

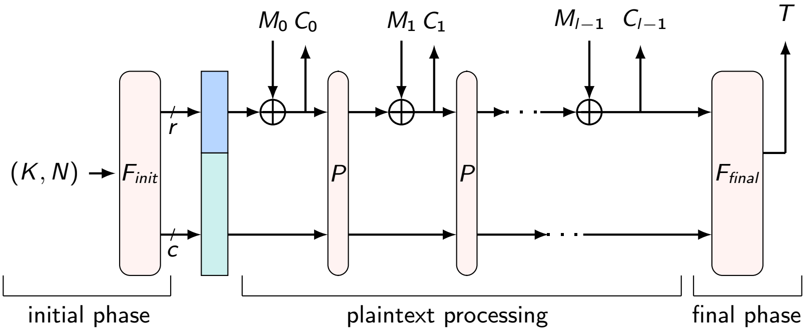


- Permutation P operates on a state of length $b = r + c$ bits, r is the rate, c the capacity.
- First r bits: the outer state
- Next c bits: the inner state

Ex: Cyclist (Xoodyak)
$r = 192$, $c = 192$

Duplex-based AE modes [BDPVA11,DMV17]

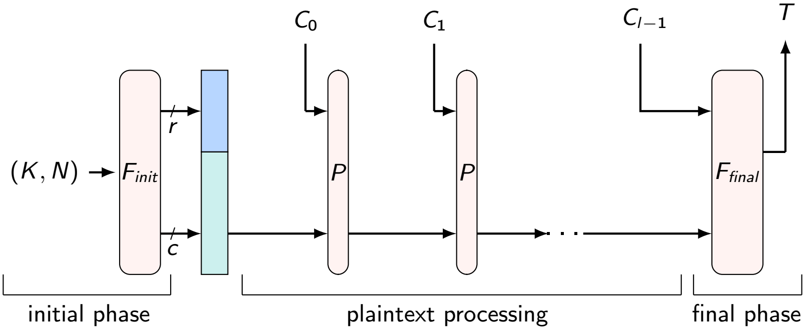
Encryption



Forgery attack: find a decryption query (N, C, T) s.t. the tag verification succeeds.

Duplex-based AE modes [BDPVA11,DMV17]

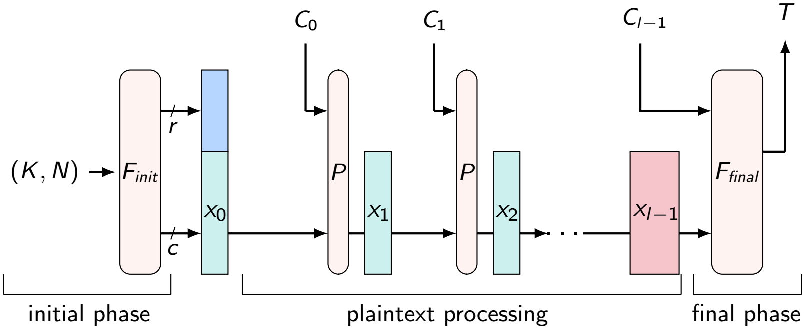
Decryption/verification



Forgery attack: find a decryption query (N, C, T) s.t. the tag verification succeeds.

Duplex-based AE modes [BDPVA11,DMV17]

Decryption/verification



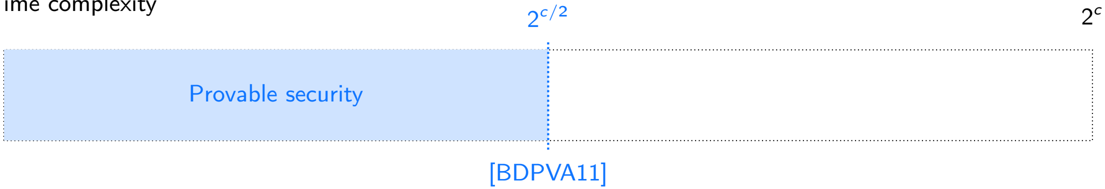
Guessing x_{l-1} allows to build a forgery.

Disclaimer: this is simplified

Security of duplex-based modes

Assuming a sufficiently large key/tag/state length:

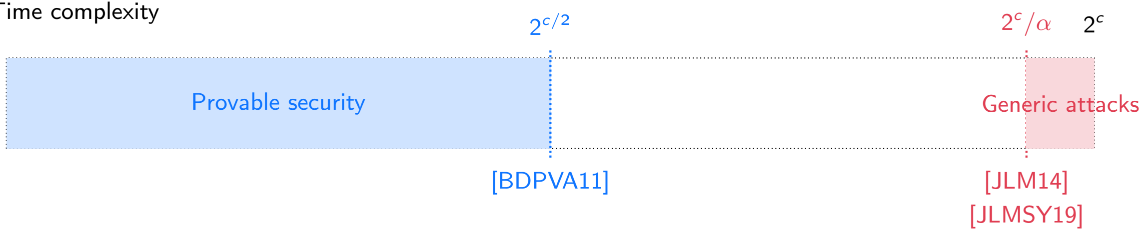
Time complexity



Security of duplex-based modes

Assuming a sufficiently large key/tag/state length:

Time complexity

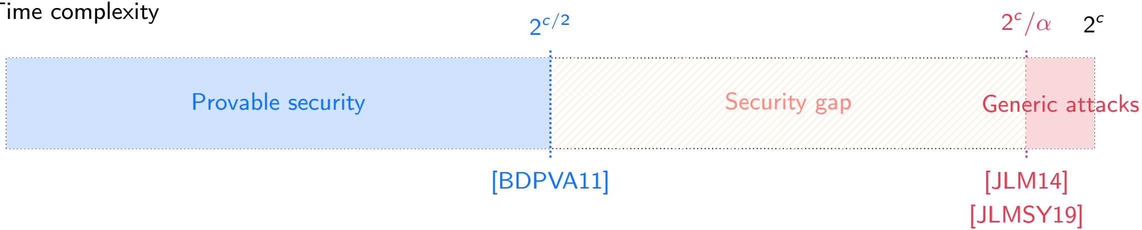


α: small constant

Security of duplex-based modes

Assuming a sufficiently large key/tag/state length:

Time complexity



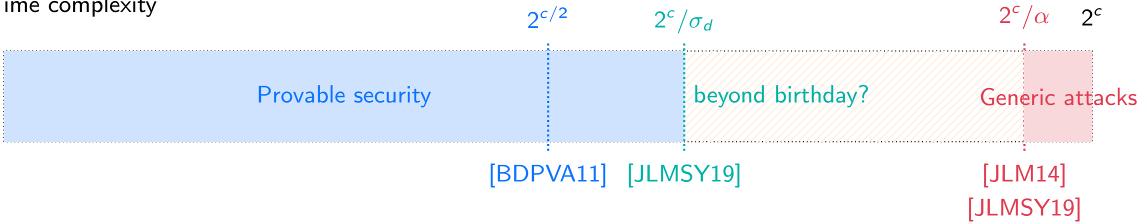
α: small constant

Disclaimer: this is simplified

Security of duplex-based modes

Assuming a sufficiently large key/tag/state length:

Time complexity



α: small constant

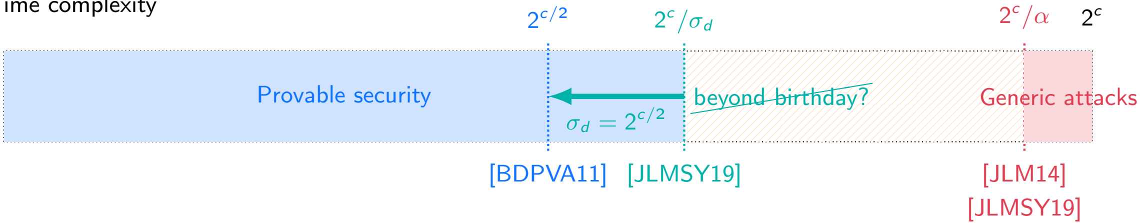
σ_d: number of online calls to P caused by forgery attempts

Disclaimer: this is simplified

Security of duplex-based modes

Assuming a sufficiently large key/tag/state length:

Time complexity

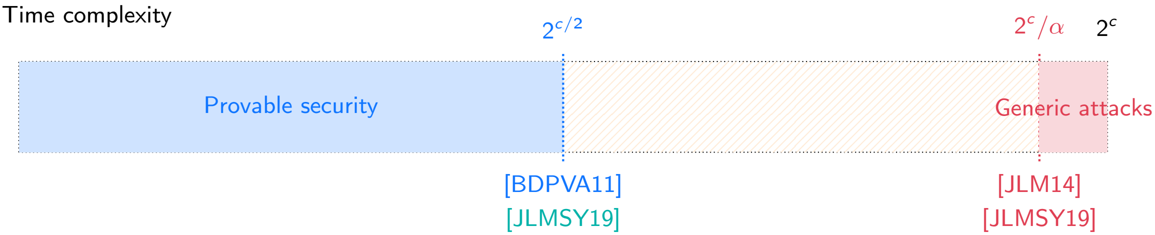


α: small constant

σ_d: number of online calls to P caused by forgery attempts

Security of duplex-based modes

Assuming a sufficiently large key/tag/state length:



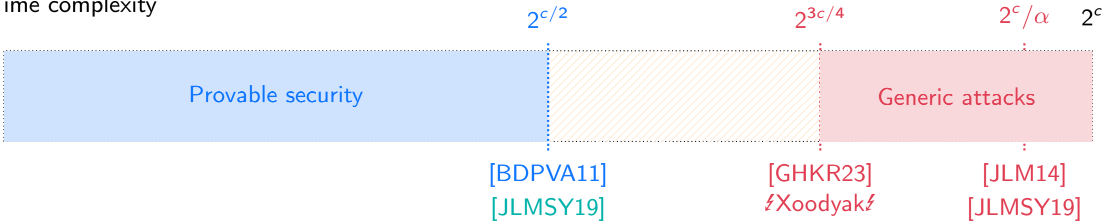
α: small constant

σ_d: number of online calls to P caused by forgery attempts

Security of duplex-based modes

Assuming a sufficiently large key/tag/state length:

Time complexity



α: small constant

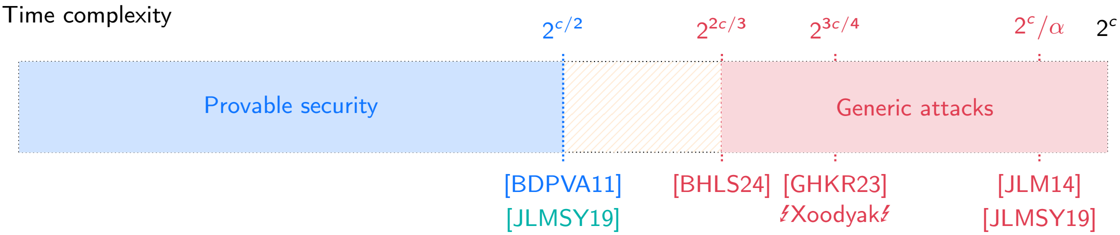
σ_d: number of online calls to P caused by forgery attempts

Generic Attack on Duplex-Based AEAD Modes Using Random Function Statistics. Gilbert, Heim Boissier, Khati, Rotella. **EUROCRYPT 2023**

Disclaimer: this is simplified

Security of duplex-based modes

Assuming a sufficiently large key/tag/state length:



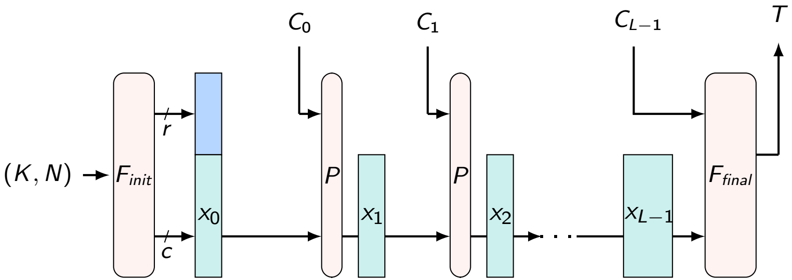
α: small constant

σ_d: number of online calls to P caused by forgery attempts

Improving Generic Attacks Using Exceptional Functions. Bonnetain, Heim Boissier, Leurent, Schrottenloher.
CRYPTO 2024

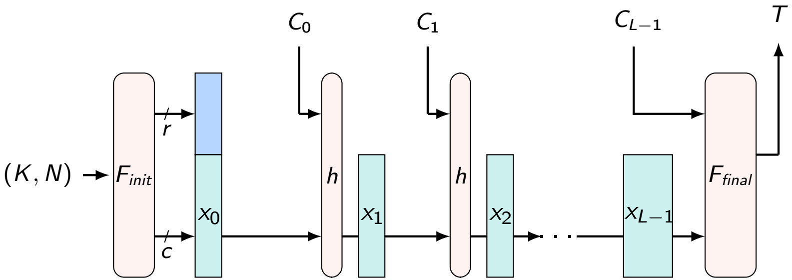
Main observation (1/2)

Verification $(C = C_0 \mid\mid \cdots \mid\mid C_{L-1}, T)$



Main observation (1/2)

Verification $(C = C_0 \mid\mid \cdots \mid\mid C_{L-1}, T)$

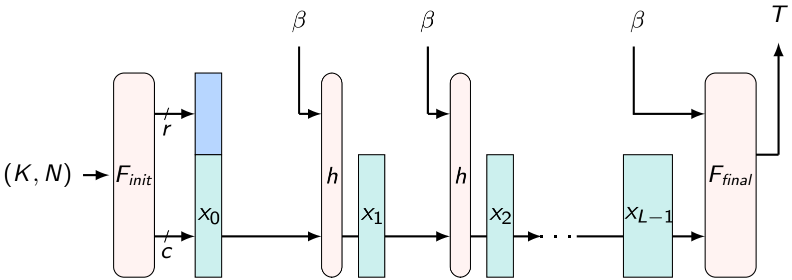


We define a compression function h induced by P:

$$\begin{aligned} h : \mathbb{F}_2^b &\longrightarrow \mathbb{F}_2^c \\ x &\longmapsto \lfloor P(x) \rfloor_c \, . \end{aligned}$$

Main observation (1/2)

Verification (β^L, T)

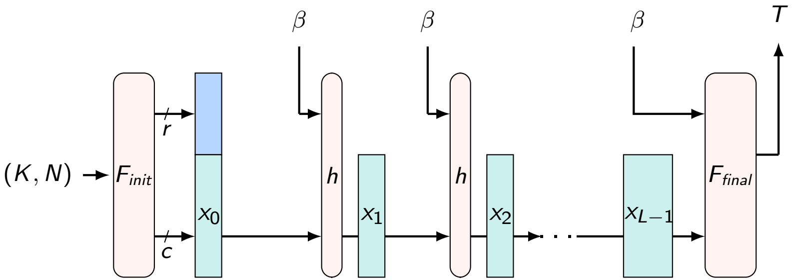


We define a compression function h induced by P:

$$\begin{aligned} h : \mathbb{F}_2^b &\longrightarrow \mathbb{F}_2^c \\ x &\longmapsto \lfloor P(x) \rfloor_c \,. \end{aligned}$$

Main observation (1/2)

Verification (β^L, T)

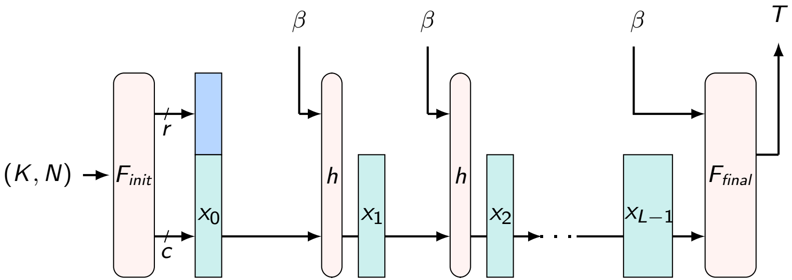


The tag verification iterates the function

$$\begin{aligned} h_\beta : \mathbb{F}_2^c &\longrightarrow \mathbb{F}_2^c \\ x &\longmapsto h(\beta, x)\,. \end{aligned}$$

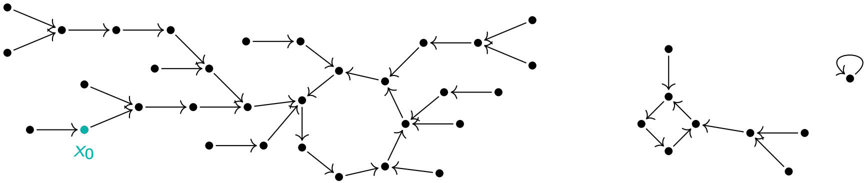
Main observation (1/2)

Verification (β^L, T)

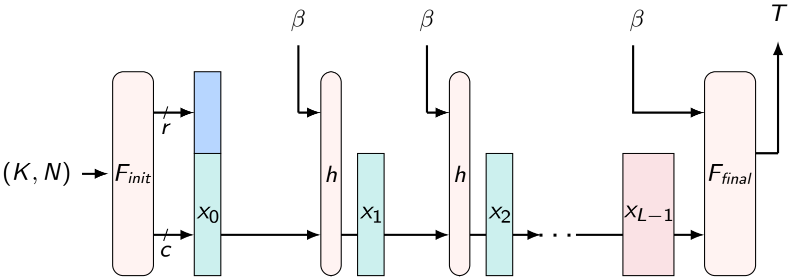


- For a random β, we expect h_β to behave as a random function drawn in $\mathfrak{F}_{2^c}$.
- For each nonce, we expect x_0 to behave as a random point drawn in the graph of h_β.

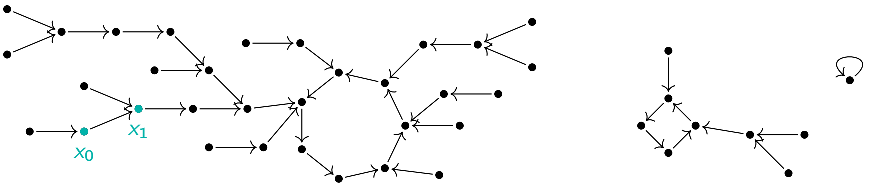
Main observation (2/2)



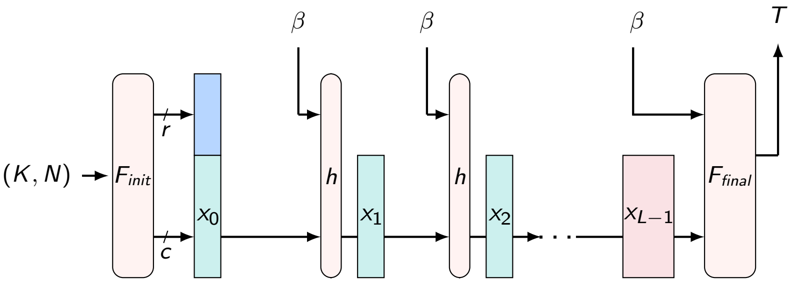
Graph of h_β



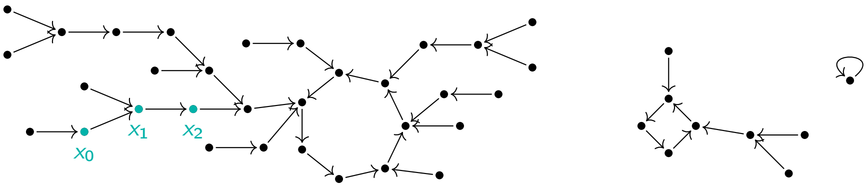
Main observation (2/2)



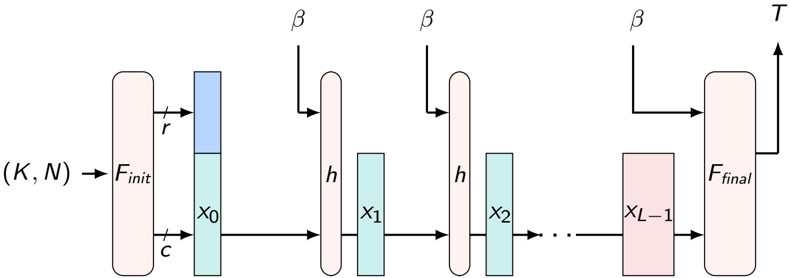
Graph of h_β



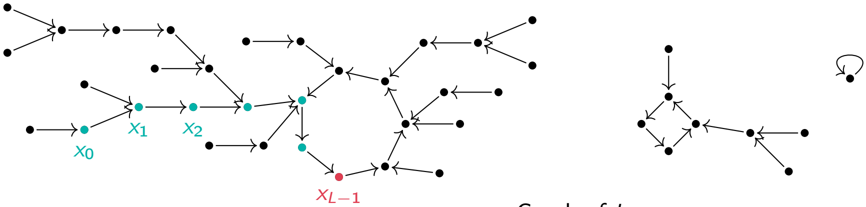
Main observation (2/2)



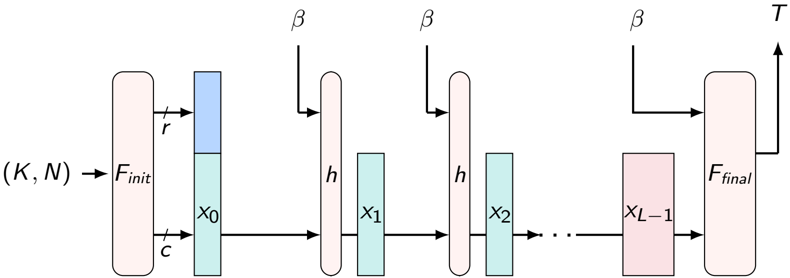
Graph of h_β



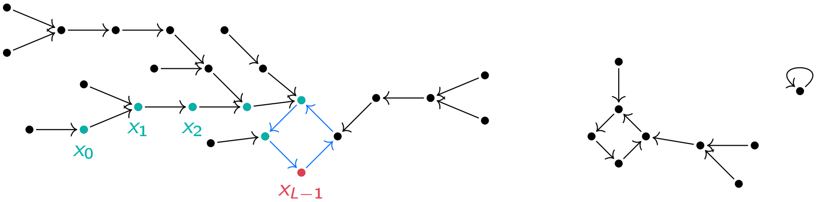
Main observation (2/2)



Graph of h_β



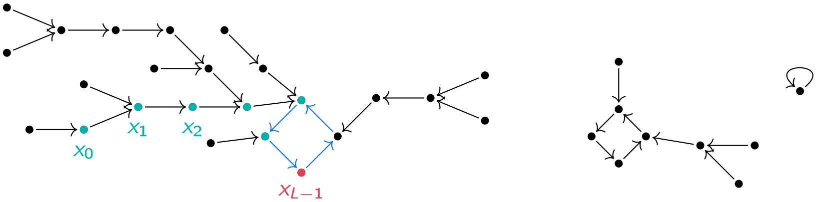
Exceptional functions



Graph of an exceptional h_β

If one finds β s.t. h_β has a reasonably large component (say $\geq 0.65 \cdot 2^c$) with an exceptionnally small cycle (say $\leq 2^{\frac{c}{4}}$)...

Exceptional functions

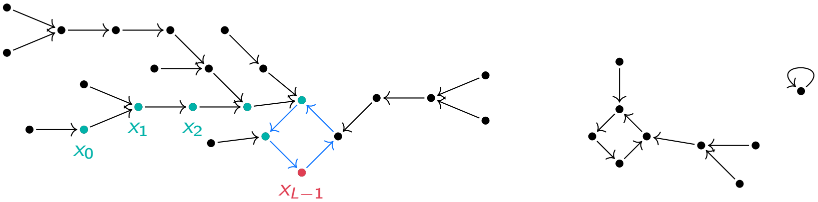


Graph of an exceptional h_β

If one finds β s.t. h_β has a reasonably large component (say $\geq 0.65 \cdot 2^c$) with an exceptionnally small cycle (say $\leq 2^{\frac{c}{4}}$)...

- x_0 belongs to the large component with good probability (≥ 0.65).

Exceptional functions

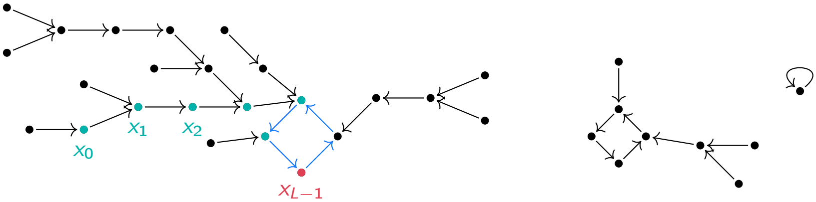


Graph of an exceptional h_β

If one finds β s.t. h_β has a reasonably large component (say $\geq 0.65 \cdot 2^c$) with an exceptionnally small cycle (say $\leq 2^{\frac{c}{4}}$)...

- x_0 belongs to the large component with good probability (≥ 0.65).
- If so, if L is 'large enough' ($L = cst \cdot 2^{\frac{c}{2}}$), x_{L-1} is in the small cycle with good probability.

Exceptional functions

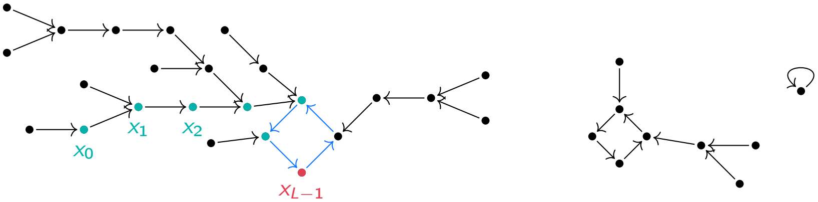


Graph of an exceptional h_β

If one finds β s.t. h_β has a reasonably large component (say $\geq 0.65 \cdot 2^c$) with an exceptionnally small cycle (say $\leq 2^{\frac{c}{4}}$)...

- x_0 belongs to the large component with good probability (≥ 0.65).
- If so, if L is 'large enough' ($L = cst \cdot 2^{\frac{c}{2}}$), x_{L-1} is in the small cycle with good probability.
- If so, there are at most $2^{\frac{c}{4}}$ possible values for x_{L-1}; i.e., at most $2^{\frac{c}{4}}$ possible tags.

Exceptional functions



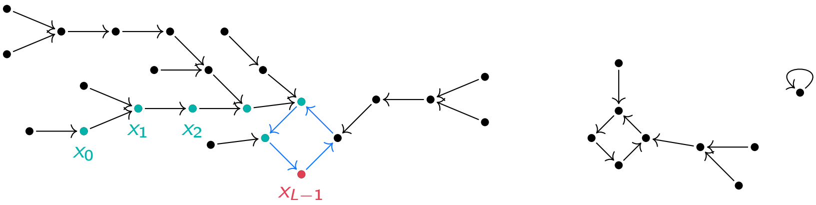
Graph of an exceptional h_β

If one finds β s.t. h_β has a reasonably large component (say $\geq 0.65 \cdot 2^c$) with an exceptionnally small cycle (say $\leq 2^{\frac{c}{4}}$)...

- x_0 belongs to the large component with good probability (≥ 0.65).
- If so, if L is 'large enough' ($L = cst \cdot 2^{\frac{c}{2}}$), x_{L-1} is in the small cycle with good probability.
- If so, there are at most $2^{\frac{c}{4}}$ possible values for x_{L-1}; i.e., at most $2^{\frac{c}{4}}$ possible tags.

Resulting forgery attack: (1) precompute an exceptional h_β and (2) try the $\leq 2^{\frac{c}{4}}$ possible values for T.

A new statistic



Graph of an exceptional h_β

[DeLaurentis87]: Probability that a h_β has a component s.t.

- (exceptionally small cycle) $\ell \leq 2^\mu$ (e.g. $\ell \leq 2^{-c/4}$);
- (reasonably large size) of size $\geq 2^c \cdot s$ (e.g. size $\geq 0.65 \cdot 2^c$):

$$p_{s,\mu} \approx \sqrt{2(1-s)/\pi s} \cdot 2^{\mu - \frac{c}{2}} \qquad \text{(e.g. } 0.6 \cdot 2^{-\frac{c}{4}}\text{)}.$$

Forgery attack [GKHR23]

1 **Precomputation phase**: Find β s.t. h_β has a large component ($\geq 0.65 \cdot 2^c$) with an exceptionally small cycle ($\leq 2^\mu$) and recover this cycle.

- For random β's,
  - Recover the cycle length using Brent's algorithm.

candidates for $\beta \approx 1/p_{s,\mu} \approx 2^{\frac{c}{2}-\mu}$

complexity $\approx 2^{\frac{c}{2}}$ applications of h

Total complexity: $\approx 2^{c-\mu}$ applications of h.

2 **Online phase**: Submit $(N, C = \beta^L, T)$ queries where $T = F_{final}(\beta||x)$, x in the cycle.

- Set $L = 3 \cdot 2^{\frac{c}{2}}$ so that x_{L-1} in the cycle with good probability
- At most 2^μ possible values for T.

Total complexity: $\approx 2^{\frac{c}{2}+\mu}$ applications of h.

Balanced complexity: $2^{\frac{3c}{4}}$

Summary of our results

Beyond an asymptotic result

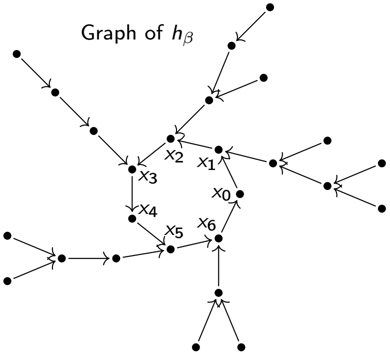
- Total time complexity: $\leq 21 \cdot 2^{\frac{3c}{4}}$.
- Probability of success: $\geq 95\%$.
- NB: almost always a key recovery (since forgery $\rightarrow$ state recovery $\rightarrow$ key recovery).

Applications

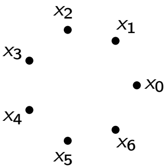
- Modes of `Norx` v2, Ketje, KNOT and Keyak;
- Attack of complexity 2^{148} on Xoodyak
  - Breaks a 184-bit security claim (corrected since).

Define a nested function $g_{\beta,\gamma}$ from the cycle $\mathscr{C}$ of h_β to itself.

Graph of h_β



Graph of $g_{\beta,\gamma}$

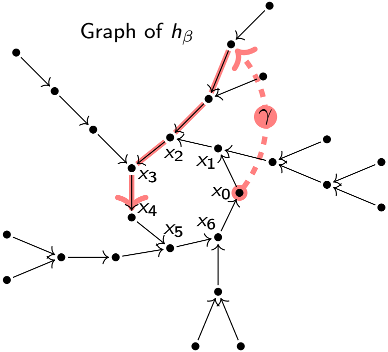


$$g_{\beta,\gamma} = h_\beta^L \circ h_\gamma : x \in \mathscr{C} \longmapsto x' \in \mathscr{C} \text{ with good probability.}$$

A new improvement [BHLS24]

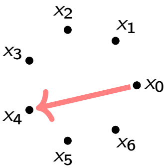
Define a nested function $g_{\beta,\gamma}$ from the cycle $\mathscr{C}$ of h_β to itself.

Graph of h_β



$g_{\beta,\gamma}(x_0) = x_4$

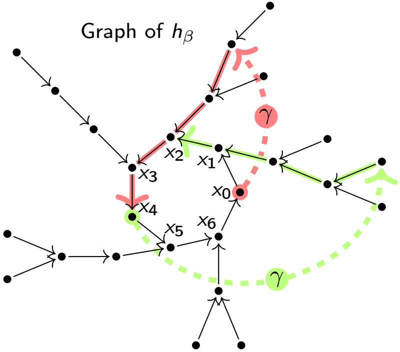
Graph of $g_{\beta,\gamma}$



$$g_{\beta,\gamma} = h_\beta^L \circ h_\gamma : x \in \mathscr{C} \longmapsto x' \in \mathscr{C} \text{ with good probability.}$$

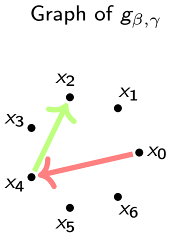
A new improvement [BHLS24]

Define a nested function $g_{\beta,\gamma}$ from the cycle $\mathscr{C}$ of h_β to itself.



$g_{\beta,\gamma}(x_0) = x_4$

$g_{\beta,\gamma}(x_4) = x_2$

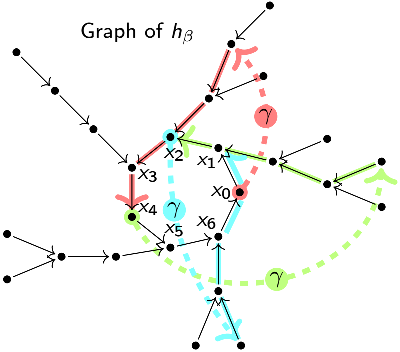


$g_{\beta,\gamma} = h_\beta^L \circ h_\gamma : x \in \mathscr{C} \longmapsto x' \in \mathscr{C}$ with good probability.

A new improvement [BHLS24]

Define a nested function $g_{\beta,\gamma}$ from the cycle $\mathscr{C}$ of h_β to itself.

Graph of h_β

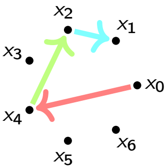


$g_{\beta,\gamma}(x_0) = x_4$

$g_{\beta,\gamma}(x_4) = x_2$

$g_{\beta,\gamma}(x_2) = x_1$

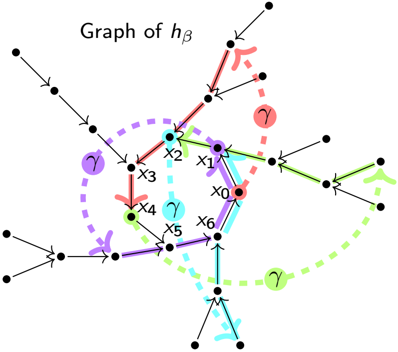
Graph of $g_{\beta,\gamma}$



$g_{\beta,\gamma} = h_\beta^L \circ h_\gamma : x \in \mathscr{C} \longmapsto x' \in \mathscr{C}$ with good probability.

A new improvement [BHLS24]

Define a nested function $g_{\beta,\gamma}$ from the cycle $\mathscr{C}$ of h_β to itself.



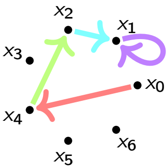
$g_{\beta,\gamma}(x_0) = x_4$

$g_{\beta,\gamma}(x_4) = x_2$

$g_{\beta,\gamma}(x_2) = x_1$

$g_{\beta,\gamma}(x_1) = x_1$

Graph of $g_{\beta,\gamma}$



$$g_{\beta,\gamma} = h_\beta^L \circ h_\gamma : x \in \mathscr{C} \longmapsto x' \in \mathscr{C} \text{ with good probability.}$$

Nesting exceptional functions [BHLS24]

1. Find β s.t. h_β is exceptional.
   - Let $2^\mu \ll 2^{c/2}$ be the cycle length of h_β.
2. Find γ s.t. $g_{\beta,\gamma}$ is exceptional.
   - For a *random* γ, $g_{\beta,\gamma}$ has cycle length $2^{\mu/2} \ll 2^{c/4}$.
   - Let $2^\nu (\ll 2^{c/4})$ be the cycle length of the exceptional $g_{\beta,\gamma}$.
3. One must only try 2^ν tags, but the ciphertexts are a lot longer.

For $\mu = 2c/7$ and $\nu = c/14$, the balanced total complexity is $2^{5c/7} < 2^{3c/4}$.

Our best attack against duplex-based modes has complexity $2^{2c/3}$.

- It uses precomputations in the graph of h_β.

Other contributions [BHLS24]

- Generic attacks against hash combiners using (nested) exceptional functions.
- Historically, those attacks use a bunch of cryptanalytic tools.
  - Joux's multi-collisions, Diamond structure, Expandable messages,...
- Using (classical) exceptional functions, we improve the best existing attacks against
  - XOR Combiner. $M \mapsto H_1(M) \oplus H_2(M)$ (preimage);
  - Zipper Hash. $M \mapsto H_2(H_1(\mathrm{IV}, M), \overleftarrow{M})$ (second preimage);
  - Hash-Twice. $M \mapsto H_2(H_1(\mathrm{IV}, M), M)$ (second preimage, second preimage quantum).

Outline

Key take-aways

Functional graphs have many applications in generic cryptanalysis.

Our contribution [GKHR23,BHLS24]

- Showing the applicability of functional graph techniques to AE modes.
- First use of exceptional behaviour of random functions.
- Bridging the gap between provable security and practical attacks.
  - A variant of our attack w/ computational complexity $O(2^c)$ is 'tight'. [Lef24]
- Beyond asymptotic results: break of a security assumption of Xoodyak.
- Improving a long series of attacks on hash combiners.

Perspectives and fun follow-up questions

Fully specified primitives

- Finding exceptional functions on real-life permutations using their specification.
- Building a backdoor permutation that 'looks' secure, but with a known exceptional function.

Overall goal: Bridging the gap between provable security and cryptanalysis.

- What about the quantum setting?

Removing residual heuristics

- Heuristic assumptions on the distribution of $t(x_0)$ for x_0 in an exceptional component.
- Experimentally verified.

Thank you for your attention!