Symmetric cryptanalysis: from primitives to modes

Rachelle Heim Boissier

Under the supervision of Christina Boura, Henri Gilbert, Yann Rotella

UVSQ, ANSSI

Journées du GDR SI, 23 juin 2025

Outline

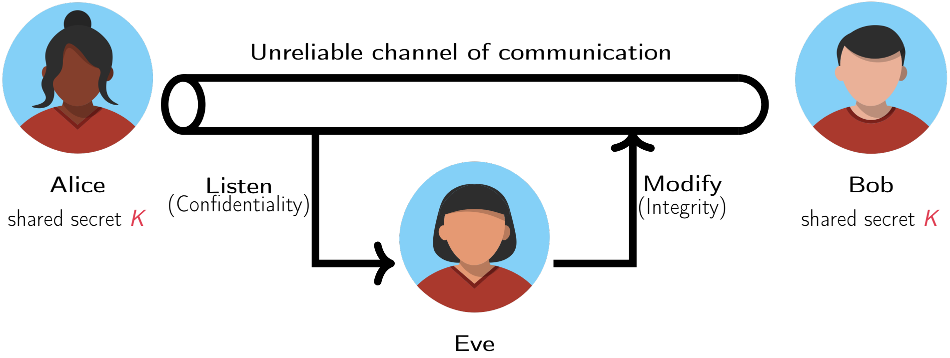
1 Symmetric cryptology

2 The key recovery step in differential attacks

3 Generic attacks on duplex-based AEAD modes

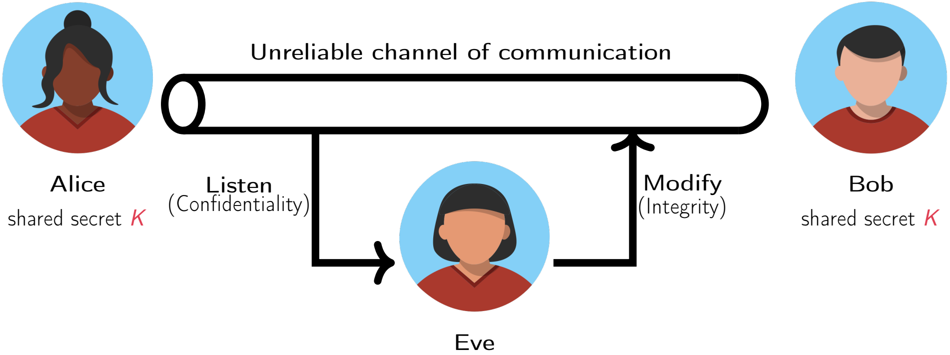
Symmetric cryptology

Efficient protection of information systems.



Symmetric cryptology

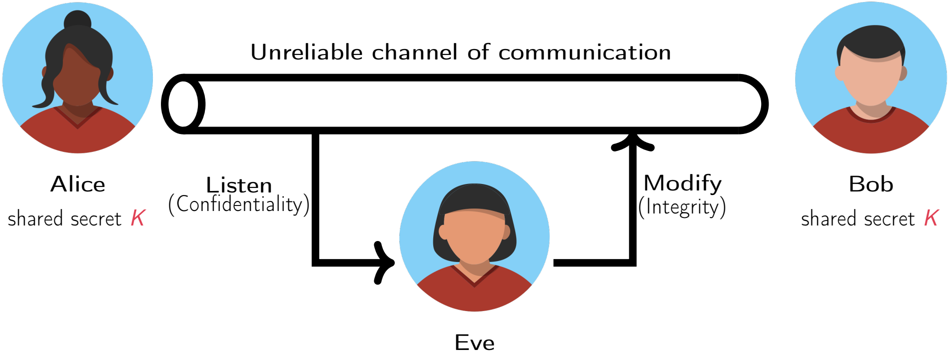
Efficient protection of information systems.



Confidentiality: The data exchanged is unintelligible (i.e. looks random) to Eve.

Symmetric cryptology

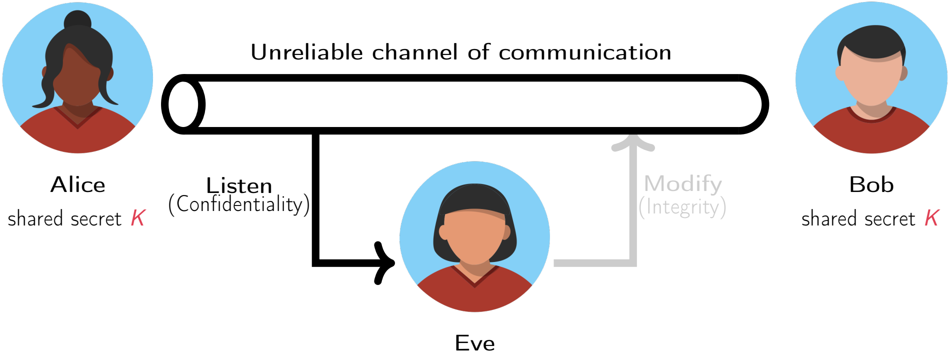
Efficient protection of information systems.



Integrity: If Eve modifies the data sent by Alice, Bob will realise.

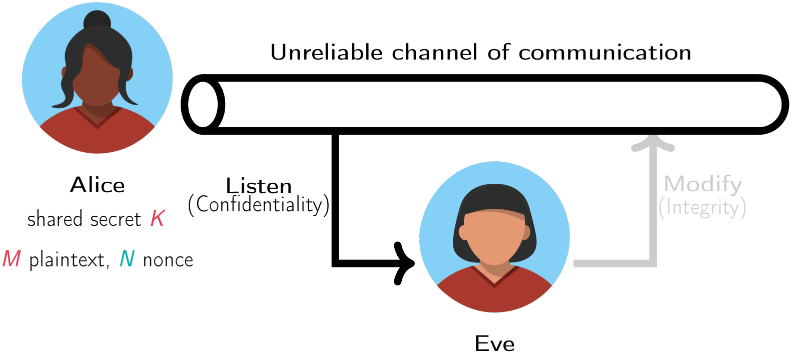
Symmetric cryptology

Efficient protection of information systems.



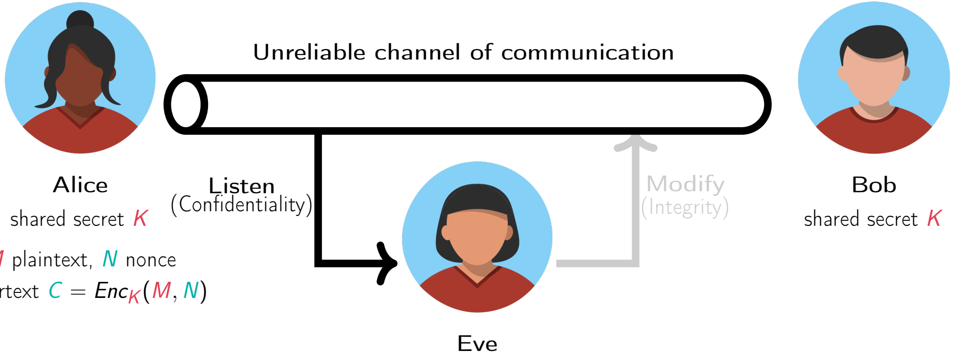
Symmetric cryptology

Efficient protection of information systems.



Symmetric cryptology

Efficient protection of information systems.

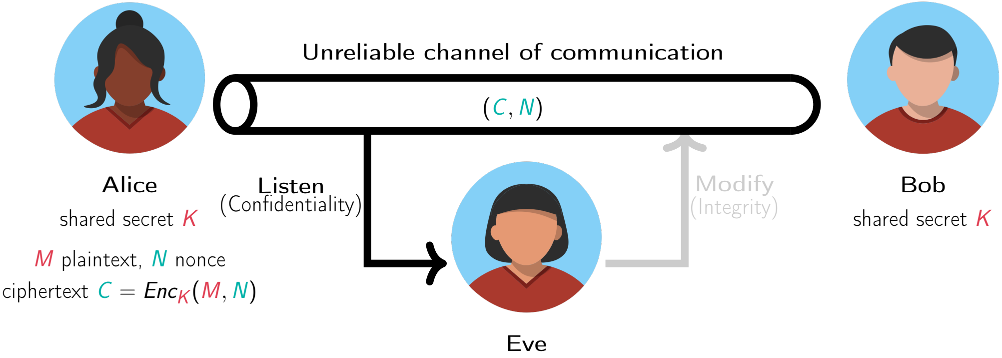


M plaintext, N nonce

ciphertext $C = Enc_K(M, N)$

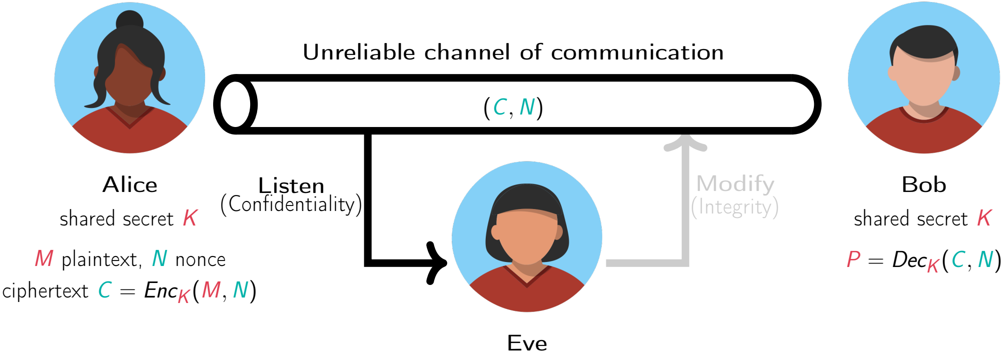
Symmetric cryptology

Efficient protection of information systems.



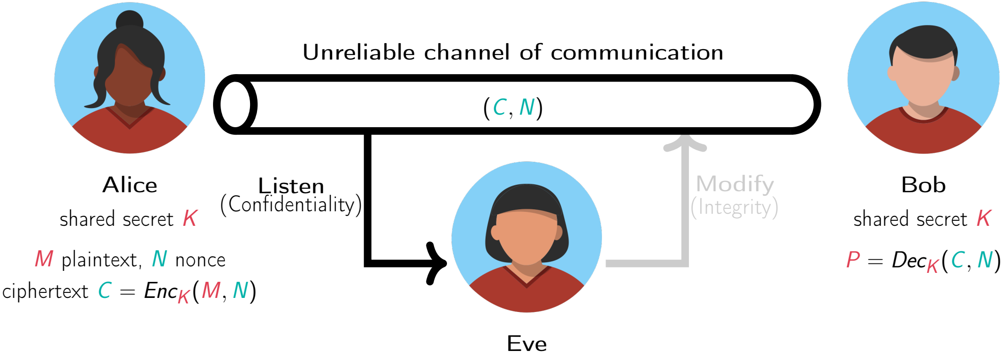
Symmetric cryptology

Efficient protection of information systems.



Symmetric cryptology

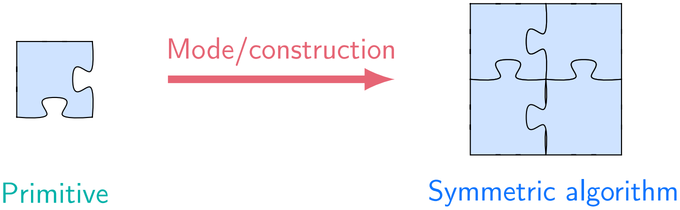
Efficient protection of information systems.



- Key must be **shared**: asymmetric/public-key cryptography.

Building symmetric algorithms

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



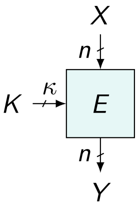
- The notion of primitive is *relative*.
- Most primitives do not provide a standalone cryptographic mechanism on their own.

Symmetric primitives

- A block cipher is a function

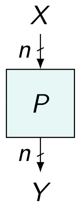
$$\begin{array}{rcl} E : \{0,1\}^{\kappa} \times \{0,1\}^{n} & \longrightarrow & \{0,1\}^{n} \\ (K, X) & \longmapsto & E(K, X) \end{array}$$

such that for any key K, $E_K(\cdot) := E(K, \cdot)$ is invertible.



E.g. NIST standard AES used in the protocol TLS for web navigation.

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



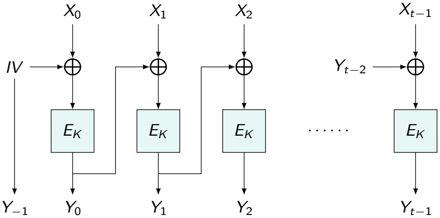
E.g. The NIST standard for lightweight applications ASCON is permutation-based.

Modes/constructions

If each pixel is encrypted independently by a block cipher:

- **Block cipher-based mode**

Ex: the encryption mode CBC.

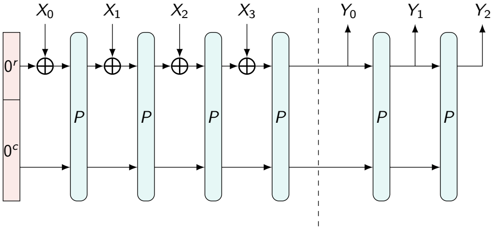


Modes/constructions

If each pixel is encrypted independently by a block cipher:

- **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 this 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 should be global, continuous and comprehensive.

Security in cryptography (2/2)

Primitive security

- can only be guaranteed through cryptanalysis.
- Security assumption ≈ must look random.

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.

Ex: generic key recovery attack on a block cipher E given X and $Y = E_K(X)$.

- Exhaustively try the 2^κ possible secret keys.

Symmetric cryptanalysis: from primitives to modes

Primitive cryptanalysis

Mode cryptanalysis

Differential cryptanalysis

Algebraic cryptanalysis

Generic attacks

Symmetric cryptanalysis: from primitives to modes

Primitive cryptanalysis | **Mode cryptanalysis**

Differential cryptanalysis

Cryptanalysis of Speedy
Eurocrypt 2023

Algebraic cryptanalysis

Generic attacks

Symmetric cryptanalysis: from primitives to modes

Primitive cryptanalysis

Mode cryptanalysis

Differential cryptanalysis

Cryptanalysis of Speedy
Eurocrypt 2023

Generic algorithm for key recovery
Eurocrypt 2024

Algebraic cryptanalysis

Generic attacks

Symmetric cryptanalysis: from primitives to modes

Primitive cryptanalysis

Mode cryptanalysis

Differential cryptanalysis

Cryptanalysis of Speedy
Eurocrypt 2023

Generic algorithm for key recovery
Eurocrypt 2024

Algebraic cryptanalysis

Cryptanalysis of Keccak (SHA-3)
FSE 2021

Generic attacks

Symmetric cryptanalysis: from primitives to modes

Primitive cryptanalysis

Differential cryptanalysis

Cryptanalysis of Speedy
Eurocrypt 2023

Generic algorithm for key recovery
Eurocrypt 2024

Algebraic cryptanalysis

Cryptanalysis of Keccak (SHA-3)
FSE 2021

Full break of `Elisabeth-4`
Asiacrypt 2023

Mode cryptanalysis

Generic attacks

Symmetric cryptanalysis: from primitives to modes

Primitive cryptanalysis

Differential cryptanalysis

Cryptanalysis of Speedy
Eurocrypt 2023

Generic algorithm for key recovery
Eurocrypt 2024

Algebraic cryptanalysis

Cryptanalysis of Keccak (SHA-3)
FSE 2021

Full break of `Elisabeth-4`
Asiacrypt 2023

Mode cryptanalysis

Generic attacks

Attack on duplex-based modes
+
Attack on full Xoodyak
Eurocrypt 2023

Symmetric cryptanalysis: from primitives to modes

Primitive cryptanalysis

Differential cryptanalysis

Cryptanalysis of Speedy
Eurocrypt 2023

Generic algorithm for key recovery
Eurocrypt 2024

Algebraic cryptanalysis

Cryptanalysis of Keccak (SHA-3)
FSE 2021

Full break of `Elisabeth-4`
Asiacrypt 2023

Mode cryptanalysis

Generic attacks

Attack on duplex-based modes
+
Attack on full Xoodyak
Eurocrypt 2023

Improved attack on duplex-based modes
Crypto 2024

Symmetric cryptanalysis: from primitives to modes

Primitive cryptanalysis

Mode cryptanalysis

Differential cryptanalysis

Cryptanalysis of Speedy
Eurocrypt 2023

Generic algorithm for key recovery
Eurocrypt 2024

Algebraic cryptanalysis

Cryptanalysis of Keccak (SHA-3)
FSE 2021

Full break of `Elisabeth-4`
Asiacrypt 2023

Generic attacks

Attack on duplex-based modes
+
Attack on full Xoodyak
Eurocrypt 2023

Improved attack on duplex-based modes
Crypto 2024

Practical break of Panther
Africacrypt 2022

Outline

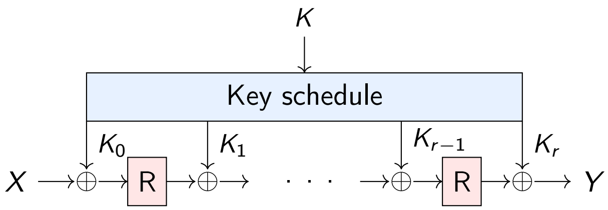
1 Symmetric cryptology

2 The key recovery step in differential attacks

3 Generic attacks on duplex-based AEAD modes

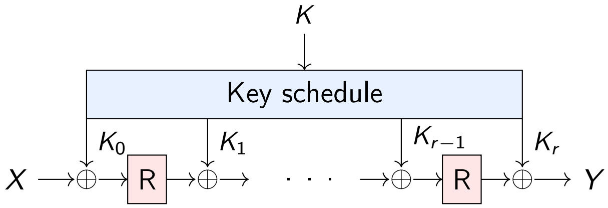
Key recovery attacks against block ciphers

General structure of an iterated block cipher

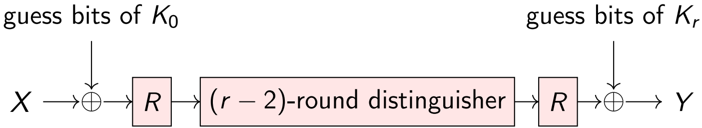


Key recovery attacks against block ciphers

General structure of an iterated block cipher

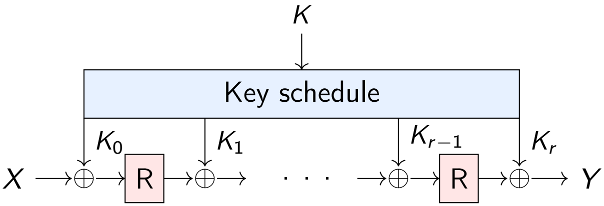


Key recovery attacks

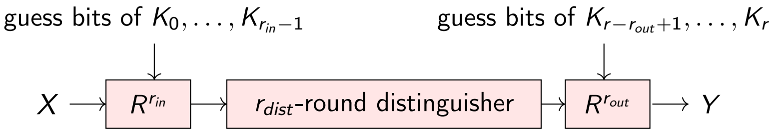


Key recovery attacks against block ciphers

General structure of an iterated block cipher



Key recovery attacks



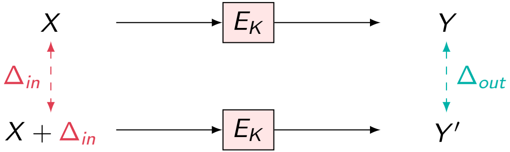
Differential cryptanalysis [BS91]

For a block cipher E, a differential is a pair of input/output differences $(\Delta_{in}, \Delta_{out}) \neq (0, 0)$.

The probability of $(\Delta_{in}, \Delta_{out})$ is the probability p that

$$E_K(X) + E_K(X + \Delta_{in}) = \Delta_{out} ,$$

for a key K and an X both chosen uniformly at random.



If $p \gg 2^{-n}$, where n is the block size, then we have a differential distinguisher on E.

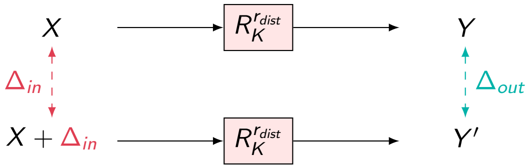
Differential cryptanalysis [BS91]

For a block cipher E, a differential is a pair of input/output differences $(\Delta_{in}, \Delta_{out}) \neq (0, 0)$.

The probability of $(\Delta_{in}, \Delta_{out})$ is the probability p that

$$R_K^{r_{dist}}(X) + R_K^{r_{dist}}(X + \Delta_{in}) = \Delta_{out} \ ,$$

for a key K and an X both chosen uniformly at random.



If $p \gg 2^{-n}$, where n is the block size, then we have a differential distinguisher on $R^{r_{dist}}$.

Differential key recovery attacks

A differential distinguisher can be used to mount a key recovery attack.

- New primitives should come with arguments of resistance by design against this technique.
- Most of the arguments used rely on showing that differential distinguishers of high probability do not exist after a certain number of rounds.
- Not always enough: A deep understanding of how the key recovery works is necessary to claim resistance against these attacks.

The example of SPEEDY

SPEEDY-7-192 (Leander, Moos, Moradi, Rasoolzadeh, TCHES 21) is a 7-round block cipher.

Designers claim:

- 'The probability of any differential characteristic over **6 rounds** is $\leq 2^{-192}$.'
- 'Not possible to add more than one key recovery round to any differential distinguisher.'

Better Steady than Speedy: Full Break of SPEEDY-7-192. Boura, David, Heim Boissier, Naya-Plasencia. **EUROCRYPT 2023**

- Distinguisher over 5.5 rounds ($\rightarrow$ of proba 0 [BN24], corrected in [BDGHN25,BN25]).
- Key recovery on 1.5 rounds.
- This work motivated us to work more specifically on the key recovery step.

The example of SPEEDY

SPEEDY-7-192 (Leander, Moos, Moradi, Rasoolzadeh, TCHES 21) is a 7-round block cipher.

Designers claim:

- ‘The probability of any differential characteristic over **6 rounds** is $\leq 2^{-192}$.’
- ‘Not possible to add more than one key recovery round to any differential distinguisher.’ **(False)**

Better Steady than Speedy: Full Break of SPEEDY-7-192. Boura, David, Heim Boissier, Naya-Plasencia. **EUROCRYPT 2023**

- Distinguisher over 5.5 rounds ($\rightarrow$ of proba 0 [BN24], corrected in [BDGHN25,BN25]).
- Key recovery on 1.5 rounds.
- This work motivated us to work more specifically on the key recovery step.

In previous works

The key recovery step is often done

- either in a 'naive' and non-efficient way;
- or using a tedious and error-prone procedure.

Emergence of new tools for cryptanalysis

- Most tools focus on the search for a differential distinguisher;
- the key recovery step is often considered using heuristics (e.g. [DF16]).

Our contribution: KYRYDI

A Generic Algorithm for Efficient Key Recovery in Differential Attacks - and its Associated Tool.
Boura, David, Derbez, Heim Boissier, Naya-Plasencia. **EUROCRYPT 2024**

Automatic key recovery for SPN block ciphers with

- a bit-permutation as linear layer;
- an (almost) linear key schedule.

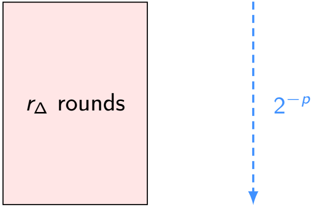
Link to our tool **KYRYDI**:

`https://gitlab.inria.fr/capsule/kyrydi`

Differential key recovery attacks

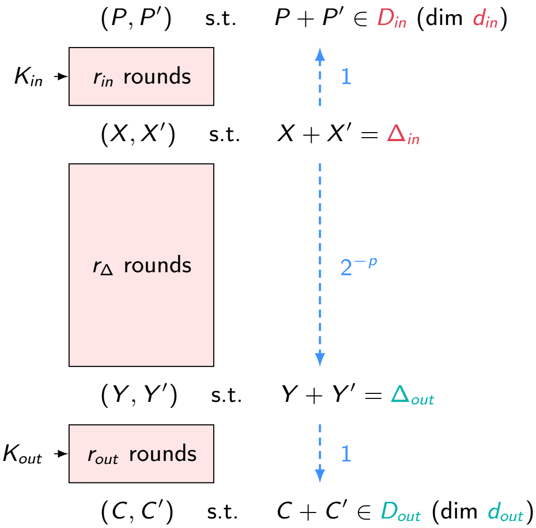
Differential distinguisher

(X, X') s.t. $X + X' = \Delta_{in}$



(Y, Y') s.t. $Y + Y' = \Delta_{out}$

Differential key recovery attacks



Differential key recovery attacks

(P, P') s.t. $P + P' \in \color{red}{D_{in}}$ (dim d_{in})

K_{in} → r_{in} rounds ↑ 1

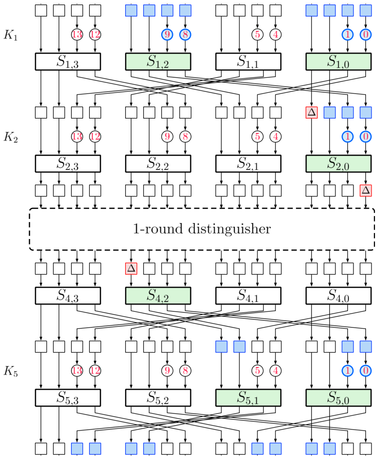
(X, X') s.t. $X + X' = \Delta_{in}$

r_Δ rounds ↓ 2^{-p}

(Y, Y') s.t. $Y + Y' = \Delta_{out}$

K_{out} → r_{out} rounds ↓ 1

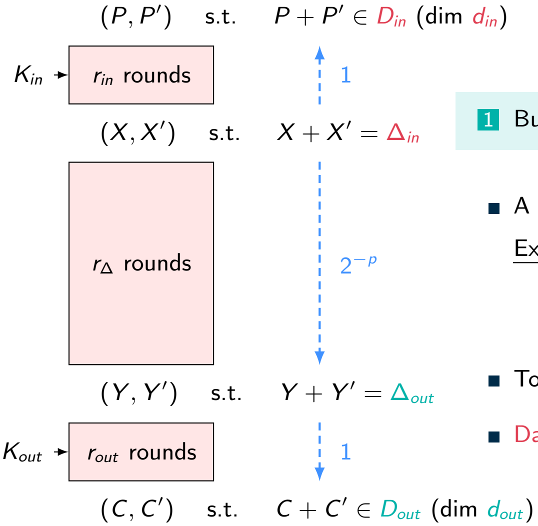
(C, C') s.t. $C + C' \in D_{out}$ (dim d_{out})



Ex: $D_{in} = \{0\}^4 \times \mathbb{F}_2^4 \times \{0\}^4 \times \mathbb{F}_2^4$, $d_{in} = 8$.

$D_{out} = \{0\}^8 \times \mathbb{F}_2^8$ $d_{out} = 8$.

Differential key recovery attacks (1/3)



1. Build enough pairs for at least one to satisfy the differential.

- A structure of size $2^{d_{in}}$ allows to build $2^{2d_{in}-1}$ pairs.

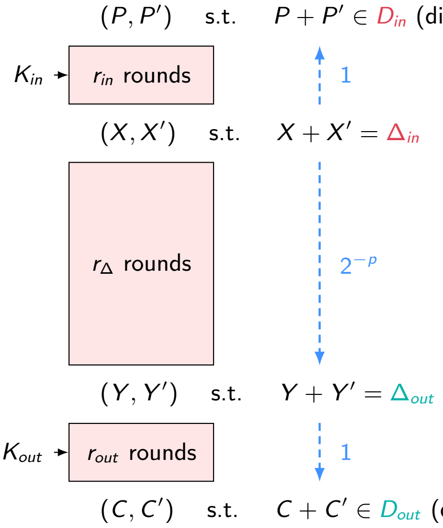
  Ex: $D_{in} = \{0\}^4 \times \mathbb{F}_2^4 \times \{0\}^4 \times \mathbb{F}_2^4$, $d_{in} = 8$.

  - Structures of the form $\{c_1\} \times \mathbb{F}_2^4 \times \{c_2\} \times \mathbb{F}_2^4$ where $c_1, c_2 \in \mathbb{F}_2^4$.

- To build enough pairs, one needs $2^{p-d_{in}+1}$ such structures.

- Data complexity: 2^{p+1} plaintexts/ciphertext pairs.

Differential key recovery attacks (2/3)



2 Filter out pairs that cannot follow the differential.

i.e. only retain the fraction $2^{d_{out}-n}$ of pairs s.t. $C + C' \in D_{out}$.

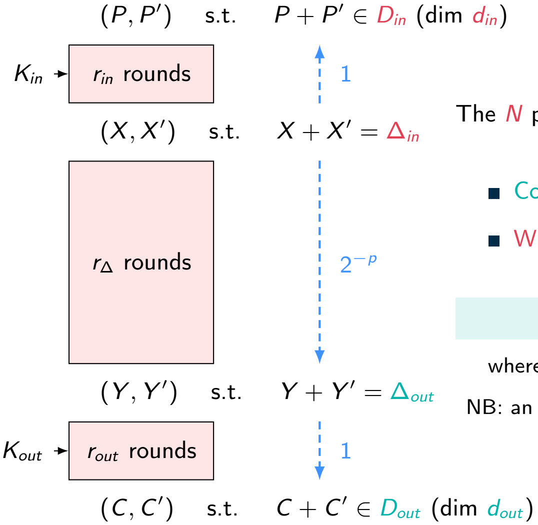
Ex: $D_{out} = \{0\}^8 \times \mathbb{F}_2^8$, $d_{out} = 8 \rightarrow$ filter 2^{-8}.

- Can be done with hash tables at a cost at most 2^{p+1} i.e. the data complexity.

Number of pairs to consider in the key recovery step:

$$N = 2^{p+d_{in}+d_{out}-n}.$$

Differential key recovery attacks



The N pairs provide a test for each guess on the involved external key material:

- Correct key guess: one pair satisfies the differential.
- Wrong key guess: on average, $2^{p-n} \ll 1$ 'false alarm(s)'.

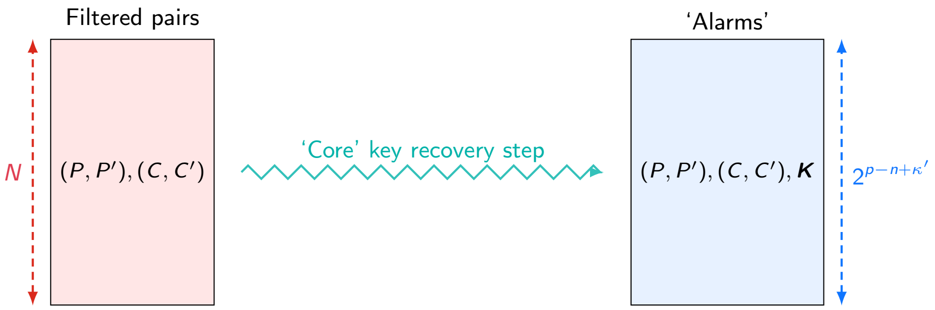
Remaining candidates: $2^{p-n+\kappa'} \ll 2^{\kappa'}$.

where κ' is the number of bits involved in the external key material.

NB: an exhaustive search on the remaining unknown key bits is required.

3. Core key recovery step

Procedure that allows to enumerate the alarms $((P, P'), (C, C'), \boldsymbol{K})$ as efficiently as possible.

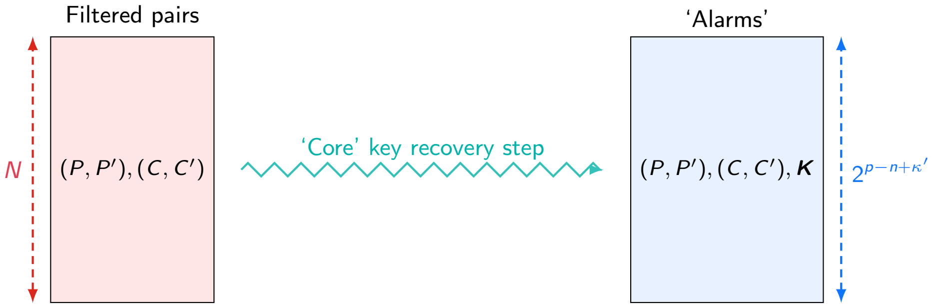


$P + P' \in D_{in}$
$C + C' \in D_{out}$

$\boldsymbol{K} \in \mathbb{F}_2^{\kappa'}$ partially encrypts/decrypts (P, P') to Δ_{in}, (C, C') to Δ_{out}

3. Core key recovery step

Procedure that allows to enumerate the alarms $((P, P'), (C, C'), \boldsymbol{K})$ as efficiently as possible.



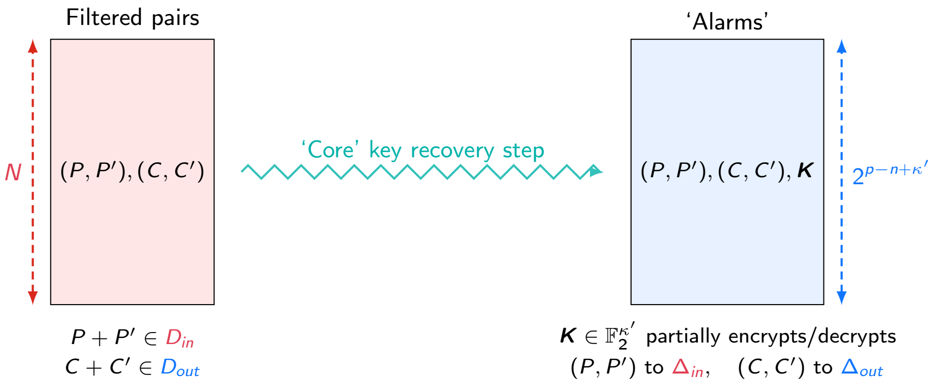
$P + P' \in D_{in}$
$C + C' \in D_{out}$

$\boldsymbol{K} \in \mathbb{F}_2^{\kappa'}$ partially encrypts/decrypts
(P, P') to Δ_{in}, (C, C') to Δ_{out}

What is the complexity of this procedure?

3. Core key recovery step

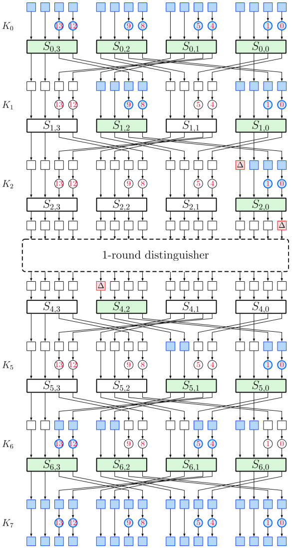
Procedure that allows to enumerate the alarms $((P, P'), (C, C'), \boldsymbol{K})$ as efficiently as possible.

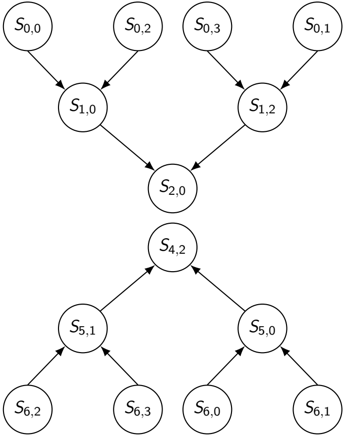


What is the complexity of this procedure?

- Upper bound: $\min(2^{\kappa}, N \cdot 2^{\kappa'})$
- Lower bound: $N + 2^{p-n+\kappa'}$

The key recovery problem as a graph

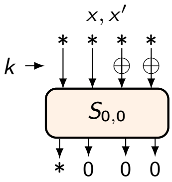




'Solving' an active S-box: For a given pair, finding the guesses on the key material that allow it to respect the differential constraints.

'Solving' S-boxes : the example of $S_{0,0}$

A solution to S is any tuple (x, x', k) s.t. $x + x' \in \nu_{in}$ and $S(x+k) + S(x'+k) \in \nu_{out}$.

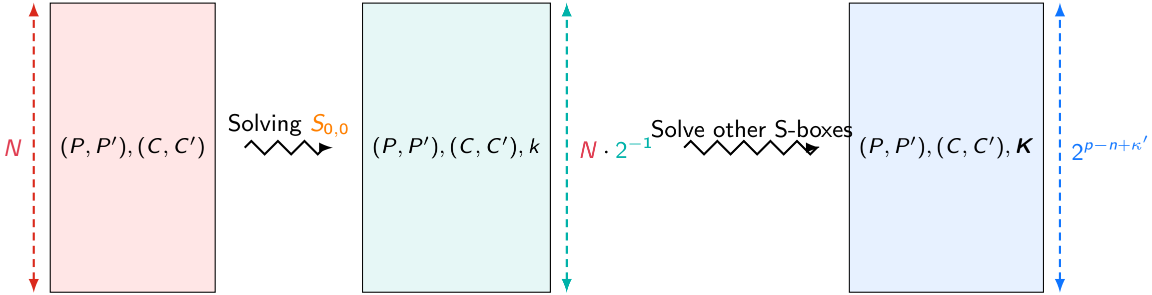


- Number of solutions (x, x', k) to $S_{0,0}$: $2^{4+1+2} = 2^7$.
- $S_{0,0}$ is an S-box of the first round:
  On any of the N pairs, the plaintext pair determines the value of (x, x').
- Probability to match a solution is $c_i = 2^7 \cdot 2^{-8} = 2^{-1}$.

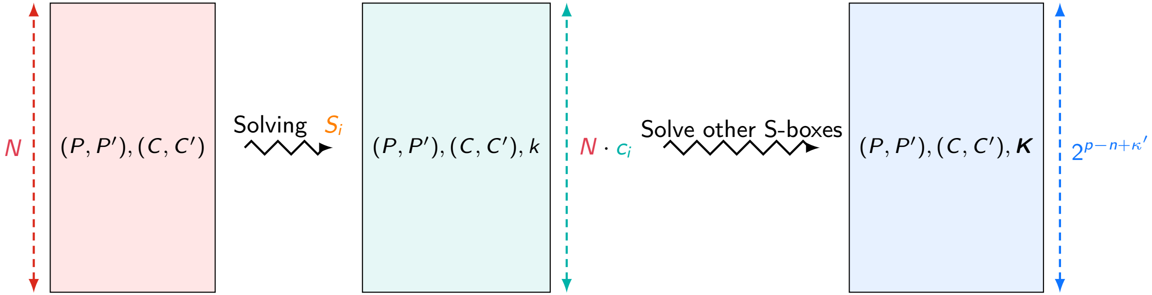
Solving $S_{0,0}$ filters $N \cdot 2^{-1}$ triplets with a determined value on 2 key bits.

Goal: Reduce the number of triplets as early as possible whilst maximizing the number of determined key bits.

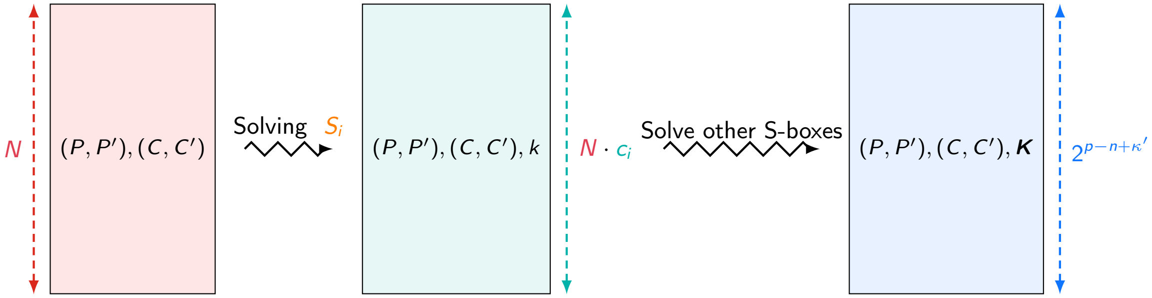
'Solving' S-boxes



'Solving' S-boxes

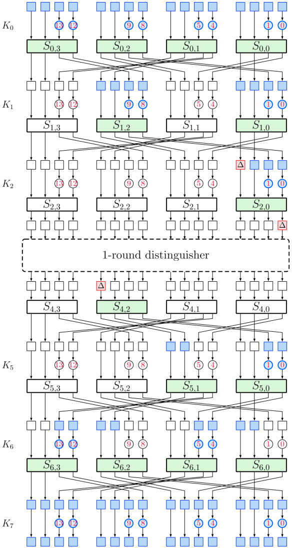


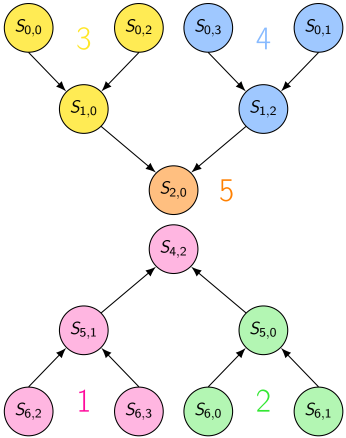
‘Solving’ S-boxes



This can be generalised to any subset of active S-boxes!

The key recovery problem as a graph





Key recovery = partition of the nodes + associated order

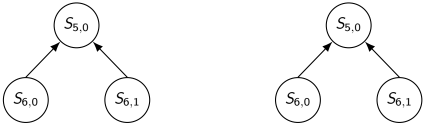
Considering strategies

Strategy $\mathscr{S}_X$ for a subgraph X

Procedure that defines a partition of X and an order in which each subgraph in the partition is solved.

Parameters of a strategy $\mathscr{S}_X$:

- number of solutions s_X
- online time complexity $T(\mathscr{S}_X)$



A strategy can be further refined with extra information: e.g. memory, offline time.

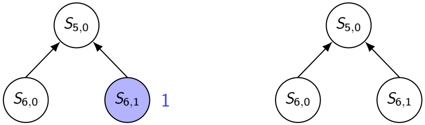
Considering strategies

Strategy $\mathscr{S}_X$ for a subgraph X

Procedure that defines a partition of X and an order in which each subgraph in the partition is solved.

Parameters of a strategy $\mathscr{S}_X$:

- number of solutions s_X
- online time complexity $T(\mathscr{S}_X)$



A strategy can be further refined with extra information: e.g. memory, offline time.

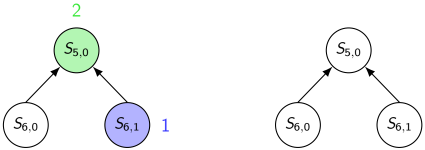
Considering strategies

Strategy $\mathscr{S}_X$ for a subgraph X

Procedure that defines a partition of X and an order in which each subgraph in the partition is solved.

Parameters of a strategy $\mathscr{S}_X$:

- number of solutions s_X
- online time complexity $T(\mathscr{S}_X)$



A strategy can be further refined with extra information: e.g. memory, offline time.

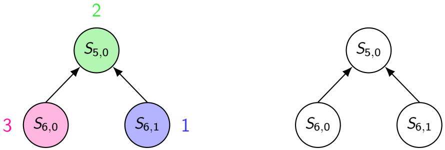
Considering strategies

Strategy $\mathscr{S}_X$ for a subgraph X

Procedure that defines a partition of X and an order in which each subgraph in the partition is solved.

Parameters of a strategy $\mathscr{S}_X$:

- number of solutions s_X
- online time complexity $T(\mathscr{S}_X)$



A strategy can be further refined with extra information: e.g. memory, offline time.

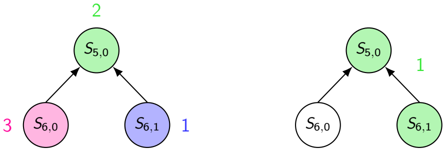
Considering strategies

Strategy $\mathscr{S}_X$ for a subgraph X

Procedure that defines a partition of X and an order in which each subgraph in the partition is solved.

Parameters of a strategy $\mathscr{S}_X$:

- number of solutions s_X
- online time complexity $T(\mathscr{S}_X)$



A strategy can be further refined with extra information: e.g. memory, offline time.

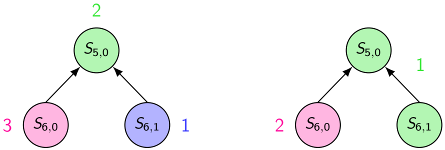
Considering strategies

Strategy $\mathscr{S}_X$ for a subgraph X

Procedure that defines a partition of X and an order in which each subgraph in the partition is solved.

Parameters of a strategy $\mathscr{S}_X$:

- number of solutions s_X
- online time complexity $T(\mathscr{S}_X)$



A strategy can be further refined with extra information: e.g. memory, offline time.

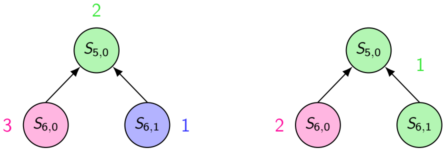
Considering strategies

Strategy $\mathscr{S}_X$ for a subgraph X

Procedure that defines a partition of X and an order in which each subgraph in the partition is solved.

Parameters of a strategy $\mathscr{S}_X$:

- number of solutions s_X
- online time complexity $T(\mathscr{S}_X)$



A strategy can be further refined with extra information: e.g. memory, offline time.

Goal: Build an efficient strategy for the whole graph.

- Based on basic strategies: strategies for a single S-box and an 'initial N pairs' strategy $\mathscr{O}$.

Merging two strategies

Assuming that $s_X < s_Y$, the merge $\mathscr{S}'$ of $\mathscr{S}_X$ and $\mathscr{S}_Y$ is the strategy which consists in

1. running $\mathscr{S}_X$, store the solutions in a hash table;
2. running $\mathscr{S}_Y$, and for each solution, look for matches.

Merging two strategies

Assuming that $s_X < s_Y$, the merge $\mathscr{S}'$ of $\mathscr{S}_X$ and $\mathscr{S}_Y$ is the strategy which consists in

1. running $\mathscr{S}_X$, store the solutions in a hash table;
2. running $\mathscr{S}_Y$, and for each solution, look for matches.

Parameters of $\mathscr{S}'$

- $s_{X \cup Y} = s_X + s_Y - \#$ bit-relations between the nodes of X and Y ⚠ log scale
- $T(\mathscr{S}') \approx \max(T(\mathscr{S}_X), T(\mathscr{S}_Y), s_{X \cup Y})$

Merging two strategies

Assuming that $s_X < s_Y$, the merge $\mathscr{S}'$ of $\mathscr{S}_X$ and $\mathscr{S}_Y$ is the strategy which consists in

1. running $\mathscr{S}_X$, store the solutions in a hash table;
2. running $\mathscr{S}_Y$, and for each solution, look for matches.

Parameters of $\mathscr{S}'$

- $s_{X \cup Y} = s_X + s_Y - \#$ bit-relations between the nodes of X and Y ⚠ log scale
- $T(\mathscr{S}') \approx \max(T(\mathscr{S}_X), T(\mathscr{S}_Y), s_{X \cup Y})$

An optimal strategy for a graph is obtained by merging two optimal strategies for two of its subgraphs.

A dynamic programming approach

'An optimal strategy for a graph is obtained by merging two optimal strategies for two of its subgraphs'

Dynamic programming approach:

- 'Clever' exhaustive search.
- Bottom-up approach: merge strategies with a small time complexity first.
- Keep only the optimal strategy found for each subgraph X.
- Restricting merges thanks to heuristics.

Applications

Start from an existing distinguisher that led to the best key recovery attack against the target cipher.

- `RECTANGLE-128`: Extended by one round the previous best attack.
  - From 18 to 19 rounds out of 25.
- `PRESENT-80`: Extended by two rounds the previous best differential attack.
  - From 16 to 18 rounds out of 31.
- `GIFT-64`: Best key recovery strategy without additional techniques.
  - 26 rounds out of 28.
- `SPEEDY-7-192`: New results available on eprint.

Future improvements, open questions

- Taking into account key-schedule relations more accurately (including non-linear ones?).
- Incorporate tree-based key recovery techniques [Bro+21].
- Handle ciphers with more complex linear layers.
- Prove optimality.
- Generalise to other attacks.

The best distinguisher does not always lead to the best key recovery!

Ultimate goal

Combine the tool with a distinguisher-search algorithm to find the best possible attacks.

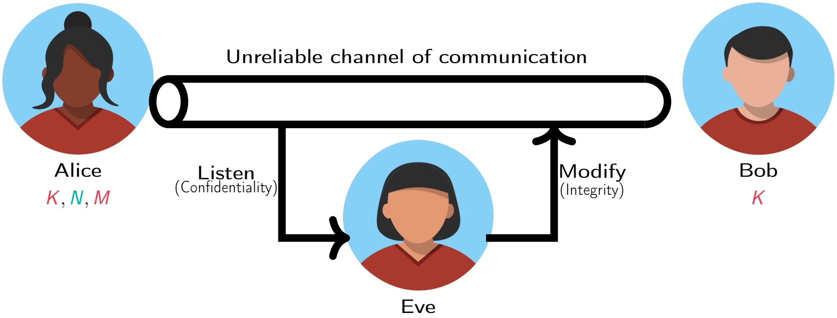
Outline

1 Symmetric cryptology

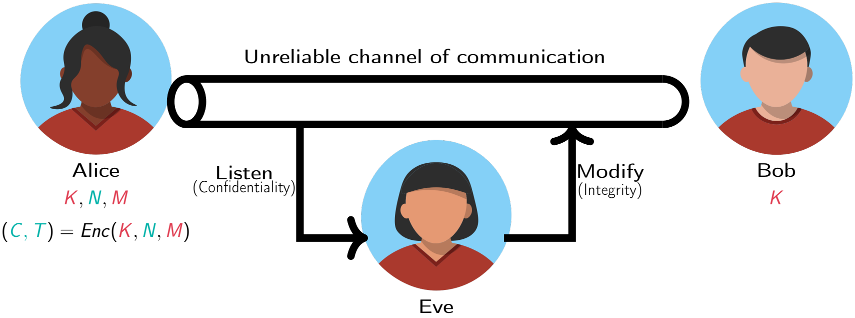
2 The key recovery step in differential attacks

3 Generic attacks on duplex-based AEAD modes

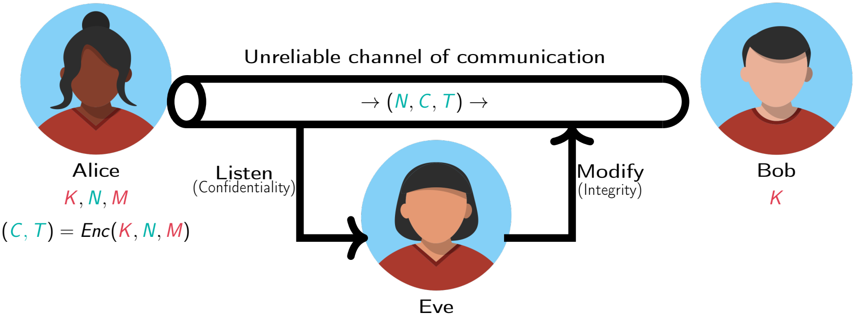
Authenticated Encryption



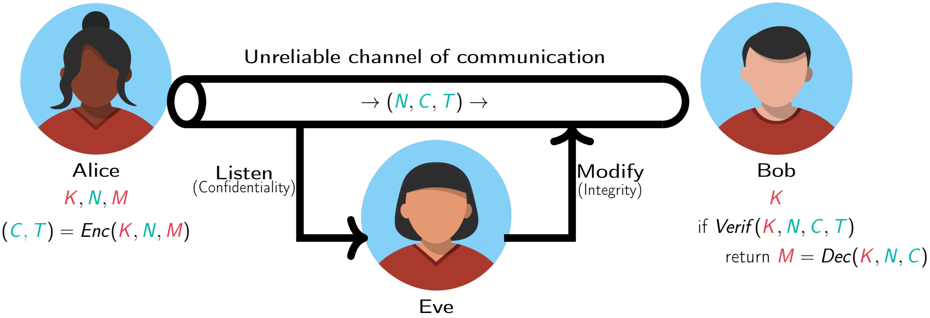
Authenticated Encryption



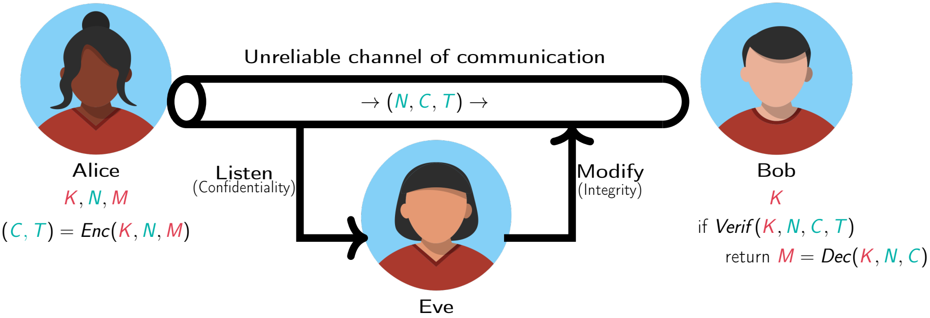
Authenticated Encryption



Authenticated Encryption

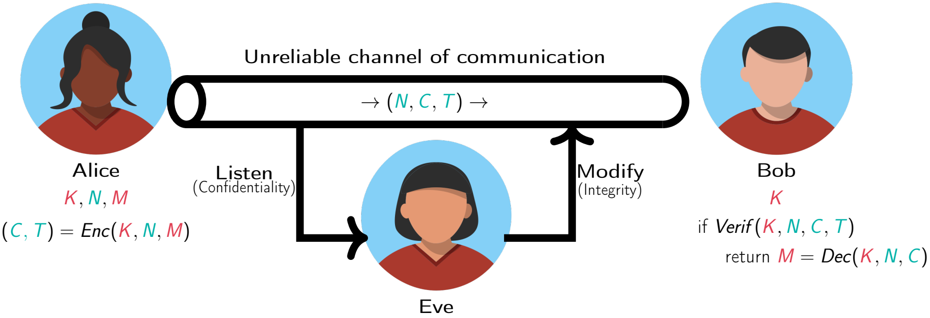


Authenticated Encryption



Forgery attack: find a decryption query (N, C, T) s.t. $Verif(K, N, C, T) = \texttt{True}$.

Authenticated Encryption



Forgery attack: find a decryption query (N, C, T) s.t. $Verif(K, N, C, T) = \texttt{True}$.

- 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.
- Permutation-based modes are proven secure when instantiated with a random permutation.
- It is difficult to assess this ‘assumption’ in practice → cryptanalysis.

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]

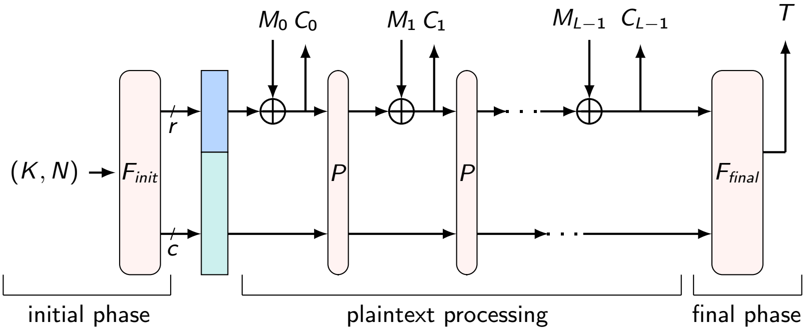
- Many candidates at the NIST lightweight competition (2018-2023), including the winner ASCON.
- Permutation-based modes are proven secure when instantiated with a random permutation.
- It is difficult to assess this 'assumption' in practice → cryptanalysis.

Our contribution [GHKR23,BHLS24]

- Generic forgery attack against duplex-based modes: we primarily break integrity.
- Based on statistics of random functions.

Duplex-based AE modes [BDPVA11,DMV17]

Encryption

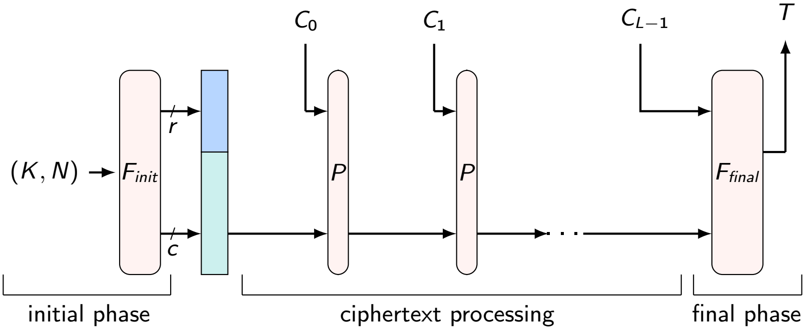


- Permutation P operates on a state of length $b = r + c$ bits.
- First r bits: the outer state
- Next c bits: the inner state

Ex: Xoodyak
$r = 192$, $c = 192$

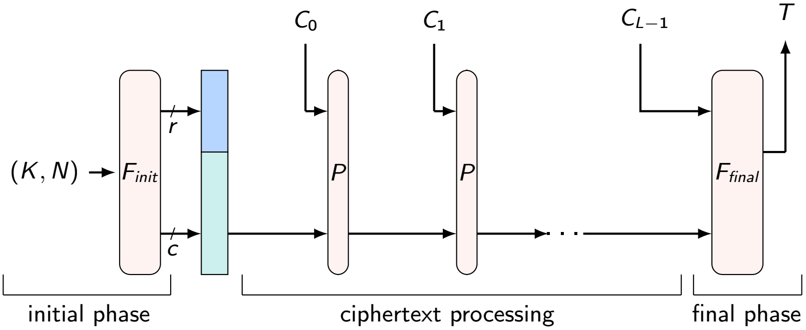
Duplex-based AE modes [BDPVA11,DMV17]

Verification



Duplex-based AE modes [BDPVA11,DMV17]

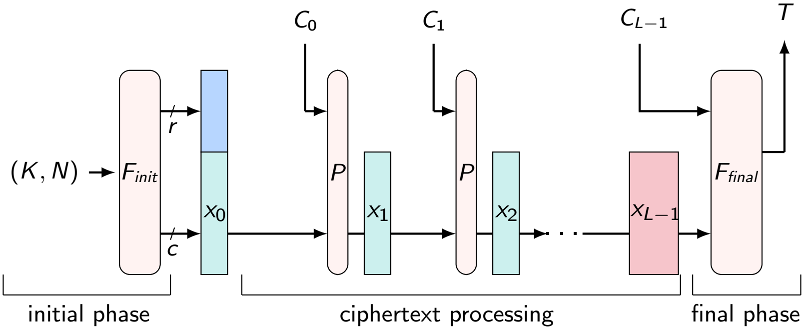
Verification



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

Duplex-based AE modes [BDPVA11,DMV17]

Verification



Recovering x_{L-1} for a known (N, C) allows to build a valid query (N, C, T).

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:

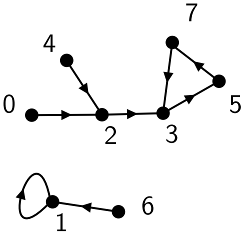
The graph of f, denoted by $G(f)$, is a directed graph such that:

- nodes are elements in the domain of f
- an edge goes from node i to node j if and only if $f(i) = j$.

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 $f(i) = j$.

$$f : [\![0;7]\!] \longrightarrow [\![0;7]\!]$$
$$\begin{cases} 0 \longmapsto 2 \\ 1 \longmapsto 1 \\ 2 \longmapsto 3 \\ 3 \longmapsto 5 \\ 4 \longmapsto 2 \\ 5 \longmapsto 7 \\ 6 \longmapsto 1 \\ 7 \longmapsto 3 \end{cases}$$



In our attacks, n is typically big, e.g. $n = 2^{128}$.

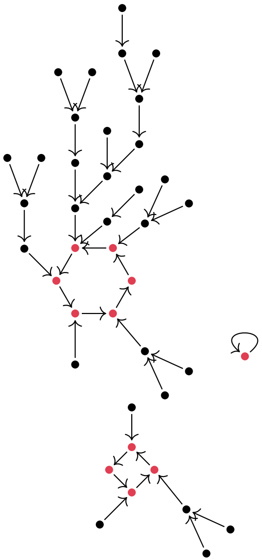
Functional graphs (1)

Definitions.

- The graph of f is 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]).

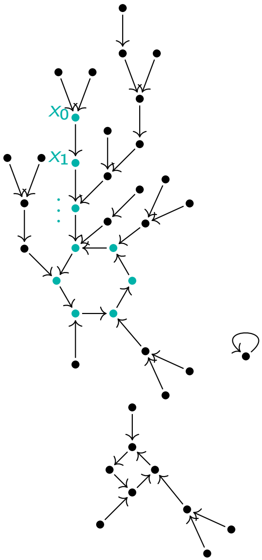
- Expected size of f's largest component: $0.76n$
- Expected size of f's largest tree: $0.48n$



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 $x_0 \in G(f)$

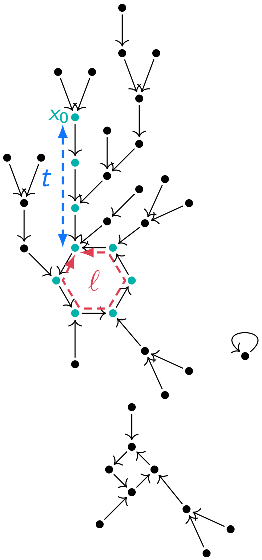
- $(x_i := f^i(x_0))_{i\in\mathbb{N}}$ is eventually periodic.
- $(x_i)_{i\in\mathbb{N}}$ graphically corresponds to a path linked to a cycle.

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 $\ell(x)$: $\sqrt{\pi n/8}$.



Cycle finding algorithms

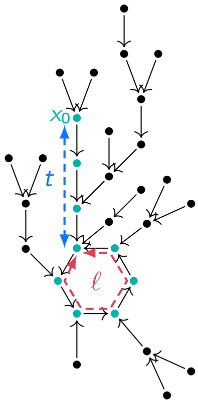
Allow to recover a cycle element using any starting point x_0 in the graph.

Ex: Floyd's algorithm, Brent's algorithm.

Use cases

- Finding a collision on a function $f \in \mathscr{F}_n$.
- Finding the cycle length.

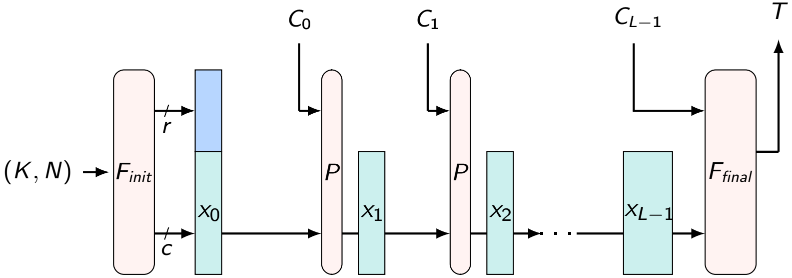
High-level idea: use iterates $x_i := f^i(x_0)$



- Time $\approx t + \ell \approx O(\sqrt{n})$.
- Memory: negligible

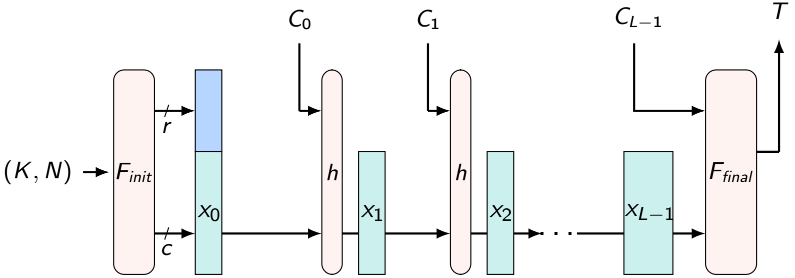
Main observation (1/2)

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



Main observation (1/2)

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

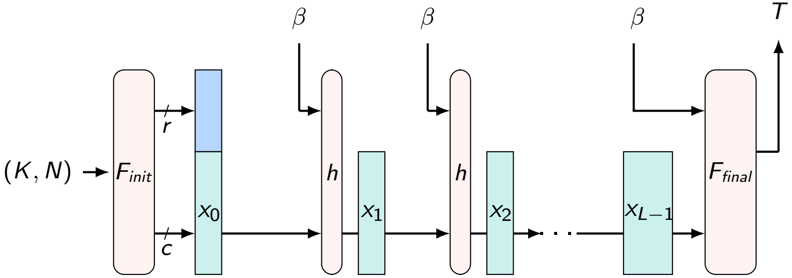


We define a compression function h induced by P:

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

Main observation (1/2)

Verification (β^L, T)

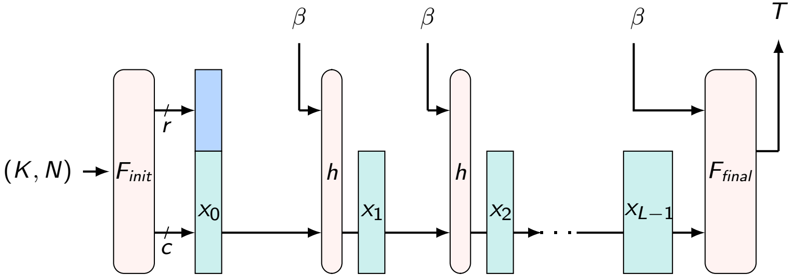


We define a compression function h induced by P:

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

Main observation (1/2)

Verification (β^L, T)

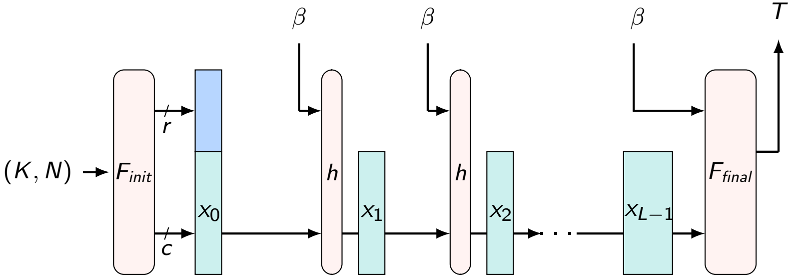


The tag verification iterates the function

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

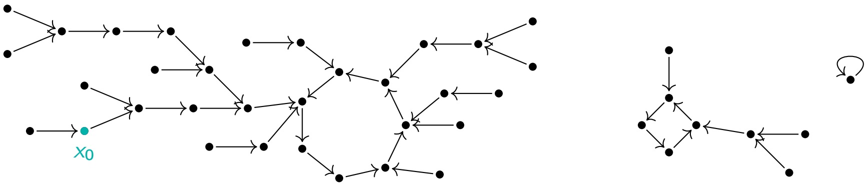
Main observation (1/2)

Verification (β^L, T)

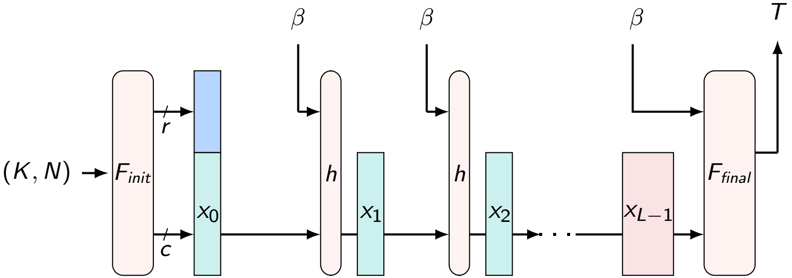


- For a random β, we expect h_β to behave as a random function drawn in $\mathfrak{F}_{2^c}$.
- For each nonce, we expect x_0 to behave as a random point drawn in the graph of h_β.

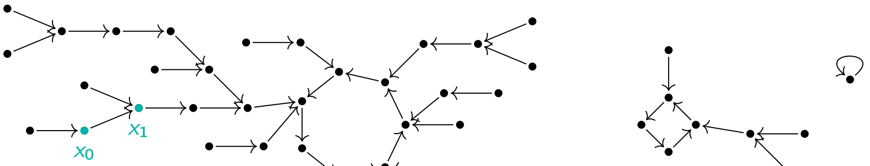
Main observation (2/2)



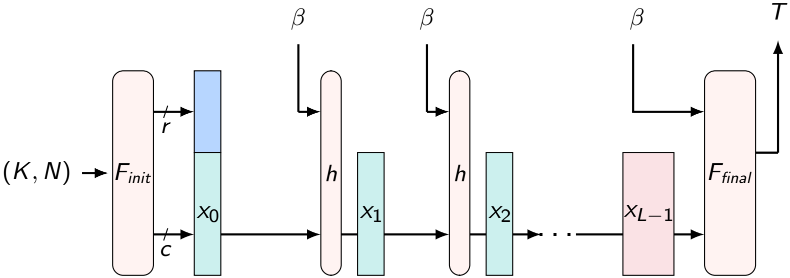
Graph of h_β



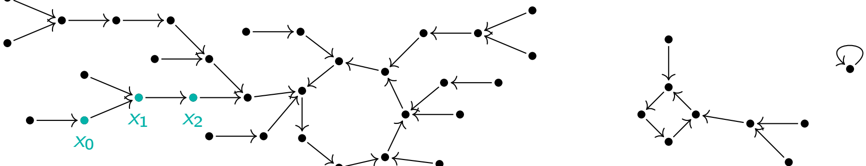
Main observation (2/2)



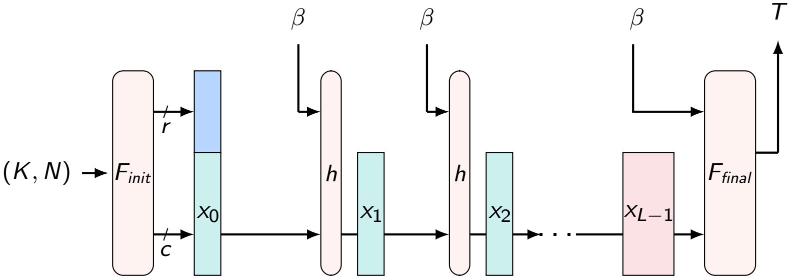
Graph of h_β



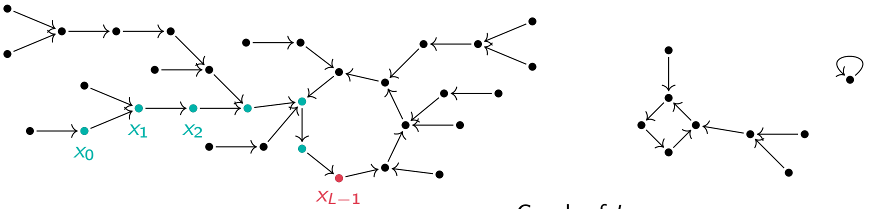
Main observation (2/2)



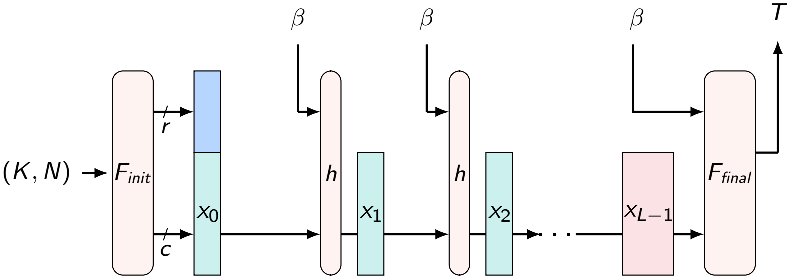
Graph of h_β



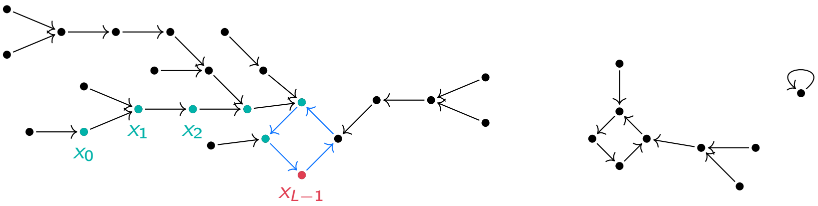
Main observation (2/2)



Graph of h_β



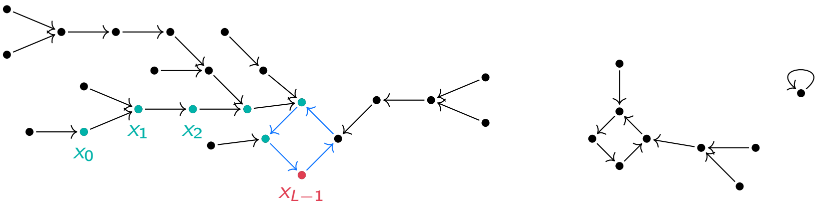
High-level idea: Exceptional functions



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 $\leq 2^{\frac{c}{4}}$)...

High-level idea: Exceptional functions

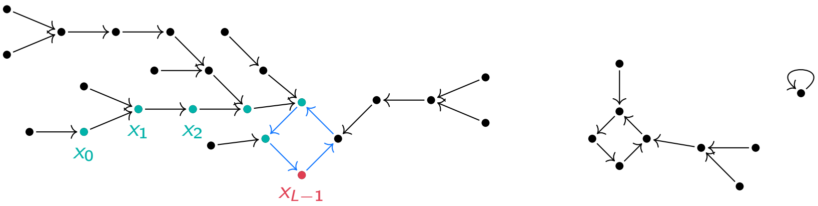


Graph of an exceptional h_β

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).

High-level idea: Exceptional functions

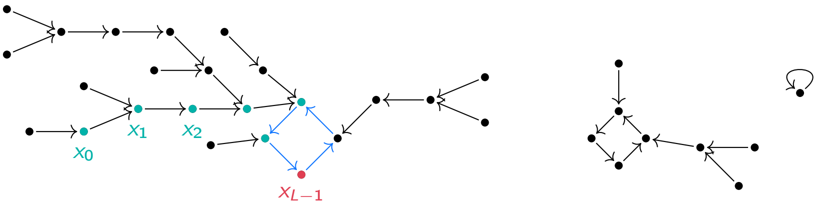


Graph of an exceptional h_β

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.

High-level idea: Exceptional functions

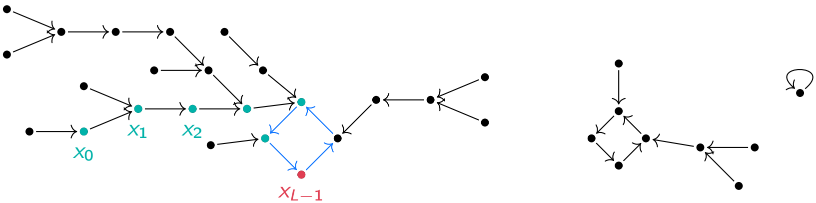


Graph of an exceptional h_β

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.

High-level idea: 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 $\leq 2^{\frac{c}{4}}$)...

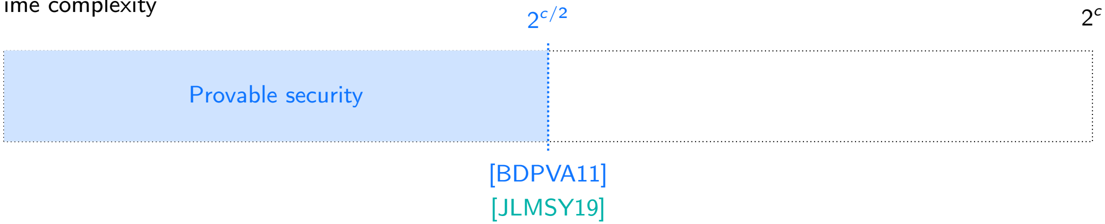
- x_0 belongs to the large component with good probability (≥ 0.65).
- If so, if L is 'large enough' ($L = cst \cdot 2^{\frac{c}{2}}$), x_{L-1} is in the small cycle with good probability.
- If so, there are at most $2^{\frac{c}{4}}$ possible values for x_{L-1}; i.e., at most $2^{\frac{c}{4}}$ possible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

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

Security of duplex-based modes

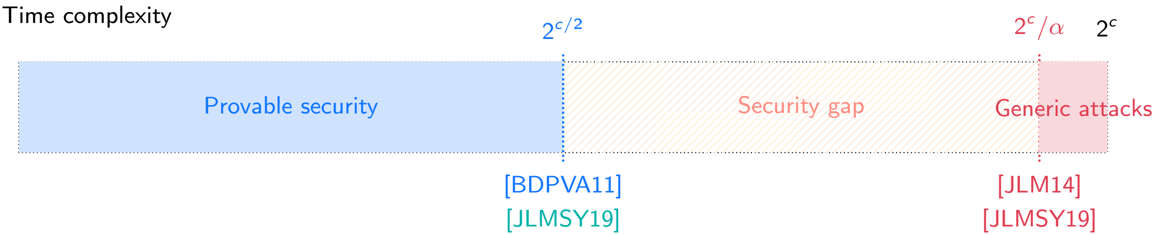
Assuming a sufficiently large key/tag/state length:

Time complexity



Security of duplex-based modes

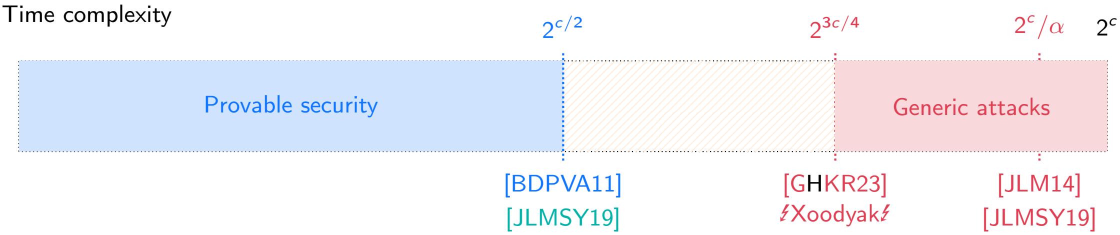
Assuming a sufficiently large key/tag/state length:



α: small constant

Security of duplex-based modes

Assuming a sufficiently large key/tag/state length:



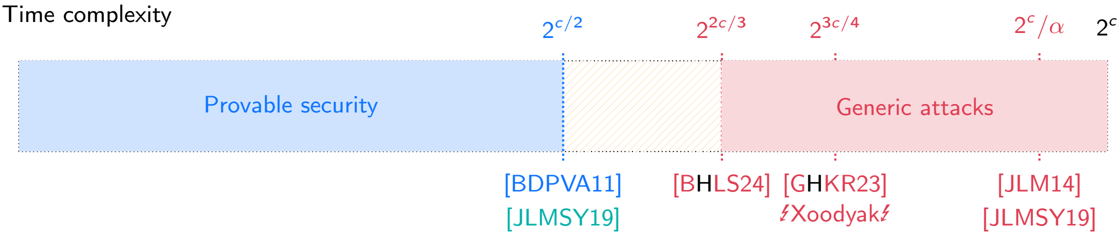
α: small constant

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

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

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**

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 (experimentally verified)

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

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 (experimentally verified)

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

Thank you for your attention!