Generic attack on Duplex-Based AEAD Modes using Random Function Statistics

Henri Gilbert, Rachelle Heim Boissier, Louiza Khati, Yann Rotella

ANSSI, UVSQ

Eurocrypt 2023, Lyon, France

Introduction

A generic attack against duplex-based AEAD modes

- A **forgery** attack
  in most cases, the key is recovered as well
- Based on **random function statistics**
  Previous works: average behaviour (see for example [BGW18])
  Our work: average and **exceptional** behaviour

Our contribution

- Improving knowledge of the **security of duplex-based modes**
- Breaking a security claim of XOODYAK [DHPVAVK20]
  (XOODYAK still meets the security requirement of NIST's LWC competition)

Duplex-based AEAD modes

Authenticated Encryption with Associated Data

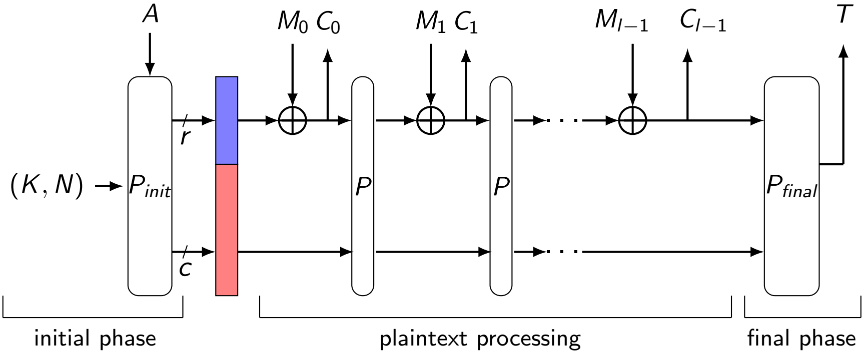
- Either **block-cipher based**: (tweakable) block cipher + mode
- Or **permutation-based**: public permutation + keyed mode
  Ex: Xoodyak = Xoodoo[12] + Cyclist [DHPVAVK20]

Duplex-based modes of operation

- Permutation-based modes introduced by Bertoni, Daemen, Peeters, Van Assche [BDPVA11]
- An adaptation to the AEAD context of the **Sponge construction** [BDPVA07]

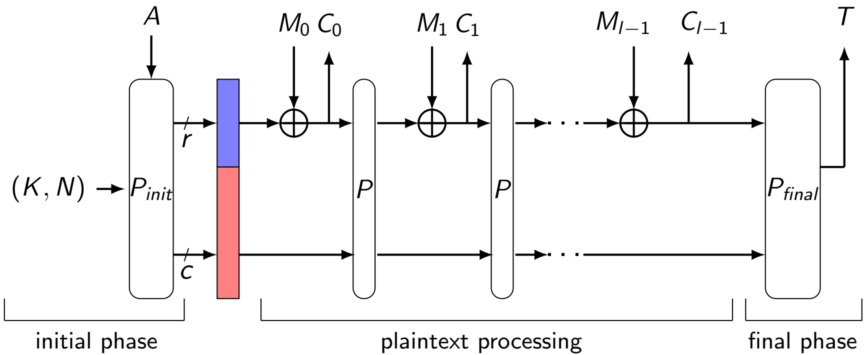
  Ex: SpongeWrap [BDPVA11], MonkeyWrap (Ketje) [BDPVAVK14], *etc.*

Duplex-based AEAD modes [BDPVA11]



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

Duplex-based AEAD modes [BDPVA11]



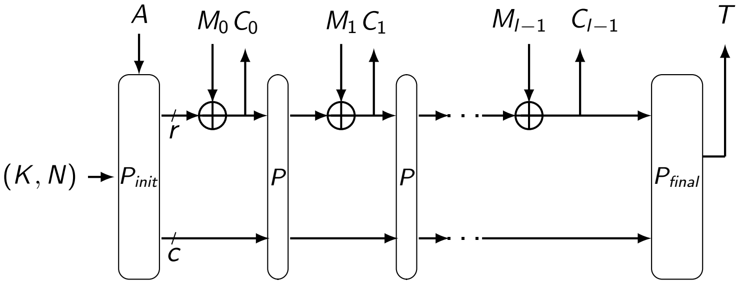
- Permutation P operates on a state of length $b = r + c$ bits, where r is the **rate** and c the **capacity**
- First r bits : the **outer state**
- Next c bits : the **inner state**

Ex: XOODYAK
r = 192
c = 192

Forgery attack on duplex-based modes

- **Privacy** and **integrity** are required in AEAD.
- It is assumed that: - the adversary is **nonce-respecting**
  - there is **no release of unverified plaintext**
- **Forgery attack:** find a decryption query (N, A, C, T) s.t. the tag verification succeeds (the decryption oracles returns the plaintext)

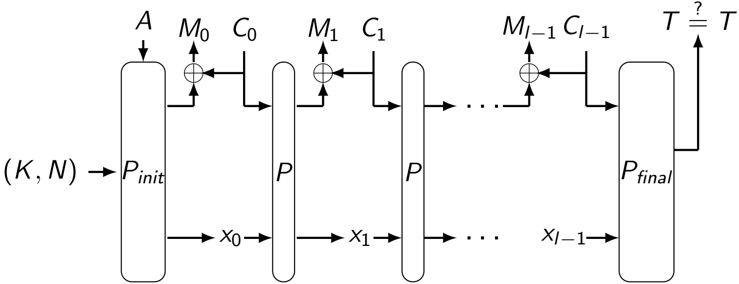
Encryption



Forgery attack on duplex-based modes

- **Privacy** and **integrity** are required in AEAD.
- It is assumed that:
  - the adversary is **nonce-respecting**
  - there is **no release of unverified plaintext**
- **Forgery attack:** find a decryption query (N, A, C, T) s.t. the tag verification succeeds (the decryption oracles returns the plaintext)

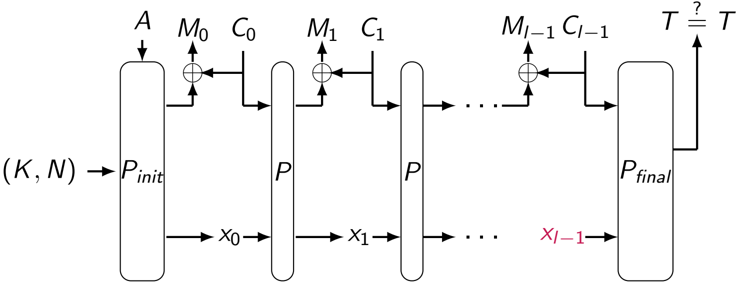
Decryption/verification



Forgery attack on duplex-based modes

- **Privacy** and **integrity** are required in AEAD.
- It is assumed that: - the adversary is **nonce-respecting**
  - there is **no release of unverified plaintext**
- **Forgery attack:** find a decryption query (N, A, C, T) s.t. the tag verification succeeds (the decryption oracles returns the plaintext)

Decryption/verification



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

Forgery attack on duplex-based modes

- **Privacy** and **integrity** are required in AEAD.
- It is assumed that: - the adversary is **nonce-respecting**
  - there is **no release of unverified plaintext**
- **Forgery attack:** find a decryption query (N, A, C, T) s.t. the tag verification succeeds (the decryption oracles returns the plaintext)

Total time complexity of an attack

$$\mathscr{T} = \sigma_e + \sigma_d + q_P + t_{extra-op}$$

where

σ_e is the number of online calls to P caused by encryption queries

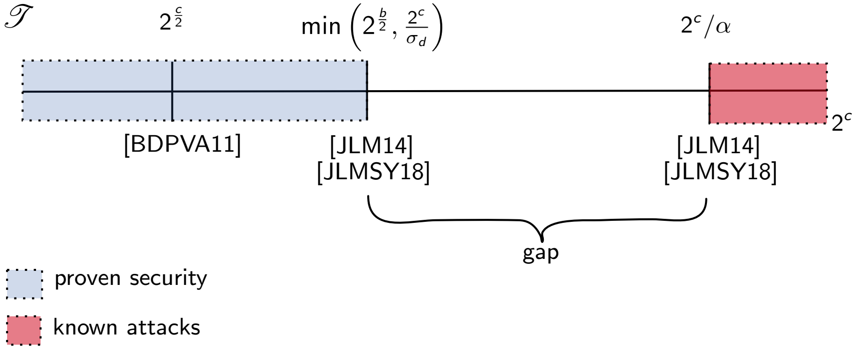
σ_d is the number of online calls to P caused by forgery attempts

q_P is the number of offline queries to P or P^{-1}

Our motivation

Disclaimer
this is (extremely) simplified

Assuming a sufficiently large key/tag length:

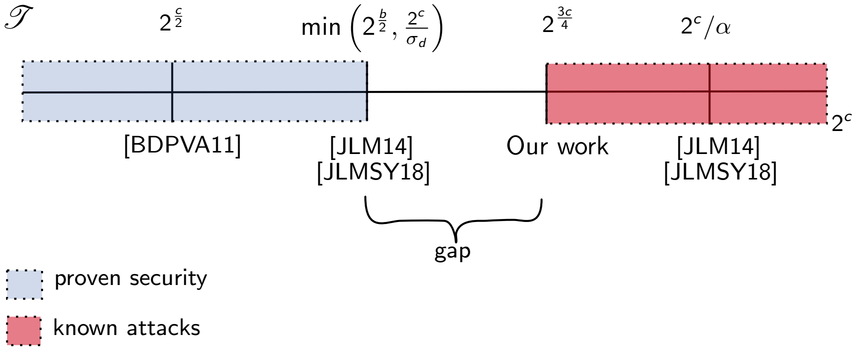


σ_d is the number of online calls to P caused by forgery attempts
α is a small constant

Our motivation

Disclaimer
this is (extremely) simplified

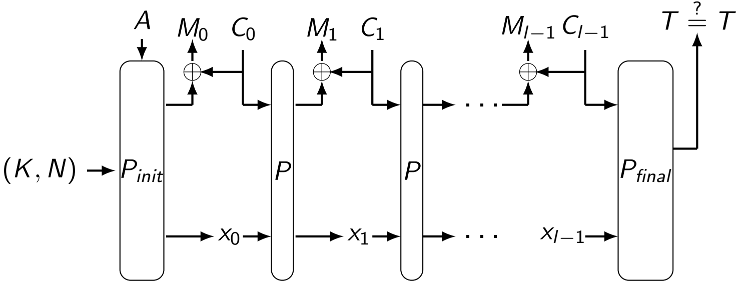
Assuming a sufficiently large key/tag length:



σ_d is the number of online calls to P caused by forgery attempts
α is a small constant

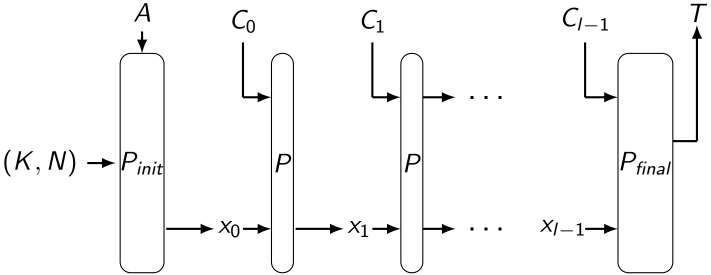
Main observation

Decrypting the ciphertext/tag pair $(C = C_0 \mid\mid \cdots \mid\mid C_{l-1}; T)$



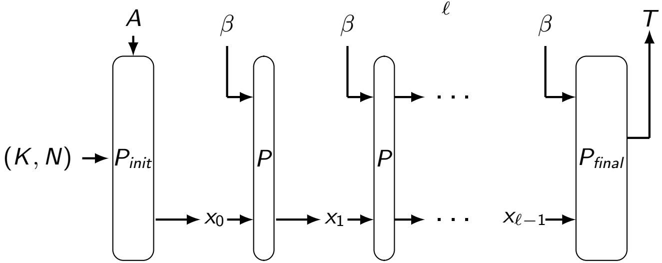
Main observation

Decrypting the ciphertext/tag pair $(C = C_0 \mid\mid \cdots \mid\mid C_{l-1}; T)$



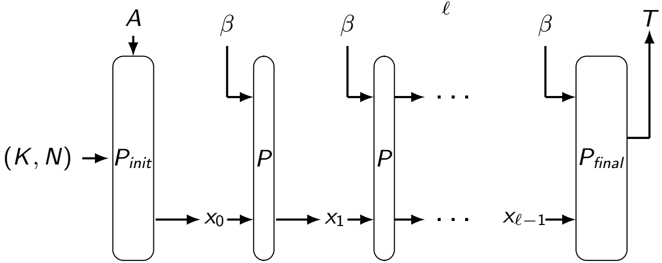
Main observation

Decrypting the long ciphertext/tag pair $(\beta_\ell = \underbrace{\beta || \cdots || \beta}_{\ell}; T)$

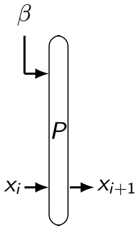


Main observation

Decrypting the long ciphertext/tag pair $(\beta_\ell = \underbrace{\beta || \cdots || \beta}_{\ell}; T)$

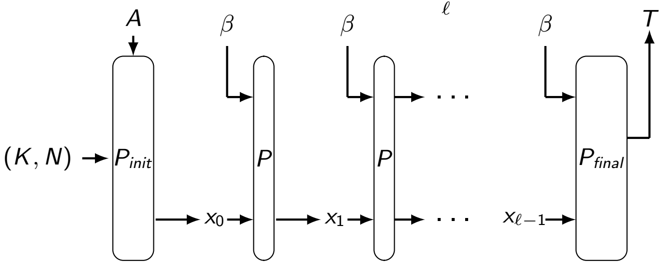


The tag verification iterates the function $F_\beta : \mathbb{F}_2^c \to \mathbb{F}_2^c$

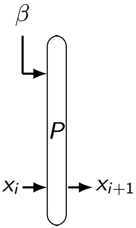


Main observation

Decrypting the long ciphertext/tag pair $(\beta_\ell = \underbrace{\beta || \cdots || \beta}_{\ell}; T)$



The tag verification iterates the function $F_\beta : \mathbb{F}_2^c \to \mathbb{F}_2^c$



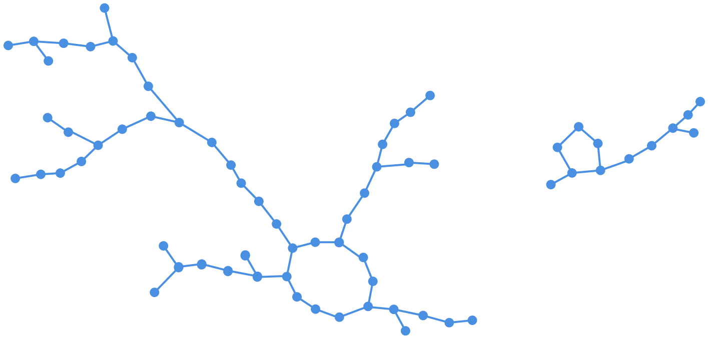
- For a random β, we expect F_β to behave as a **random function** drawn in $\mathfrak{F}_{2^c}$.
- For each nonce, we expect x_0 to behave as a **random point** drawn in the graph of F_β.

Graph of a random function F in $\mathfrak{F}_{2^c}$

Def: node i goes to node j iff $F(i) = j$.

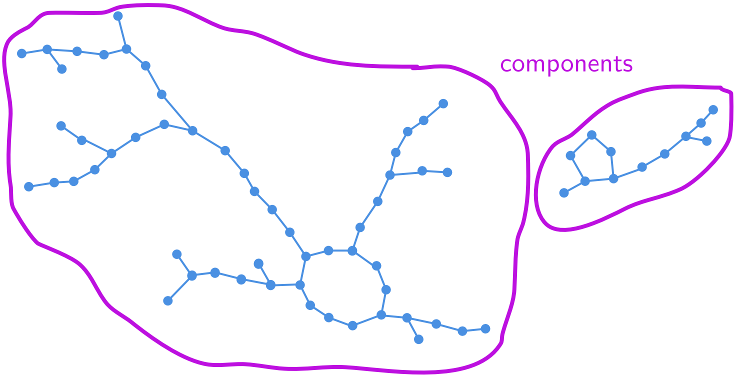
Graph of a random function F in $\mathfrak{F}_{2^c}$

Def: node i goes to node j iff $F(i) = j$.



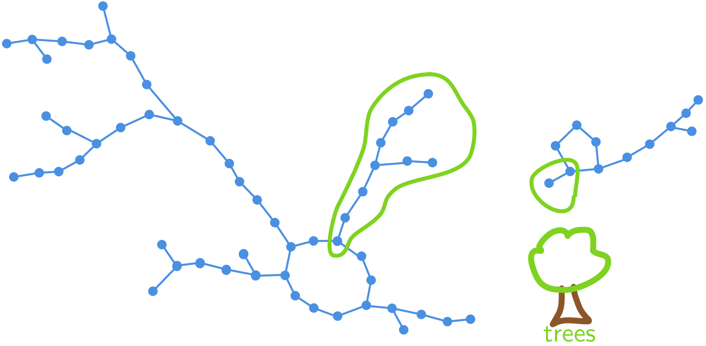
Graph of a random function F in $\mathfrak{F}_{2^c}$

Def: node i goes to node j iff $F(i) = j$.



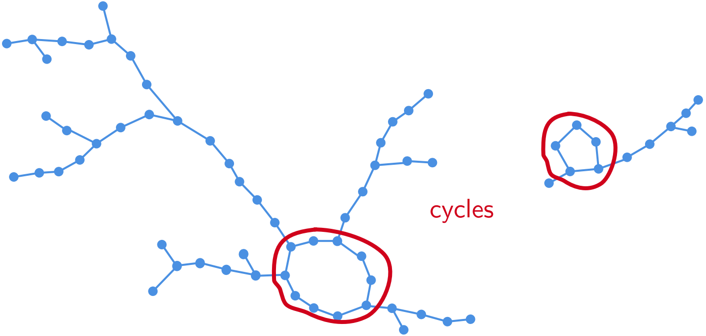
Graph of a random function F in $\mathfrak{F}_{2^c}$

Def: node i goes to node j iff $F(i) = j$.

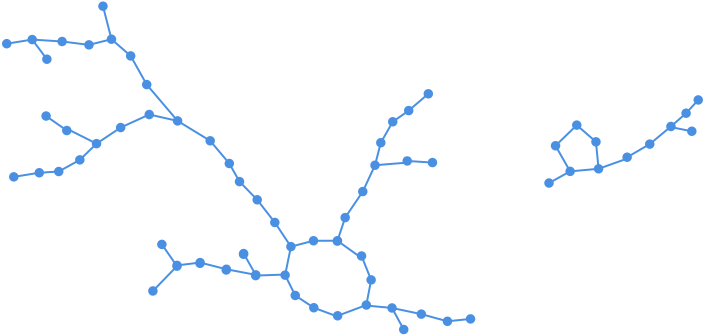


Graph of a random function F in $\mathfrak{F}_{2^c}$

Def: node i goes to node j iff $F(i) = j$.



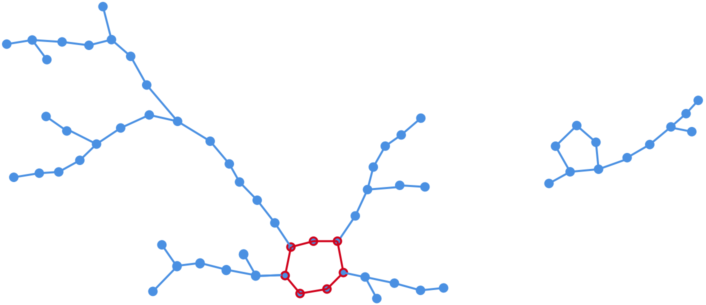
Graph of a random function F in $\mathfrak{F}_{2^c}$



Average...

- Size of the largest component: $2^c \times 0.76$. [FO89]
- Cycle/tail length of a random point: $2^{\frac{c}{2}}\sqrt{\pi/8}$

Graph of a random function F in $\mathfrak{F}_{2^c}$



The probability that a random function has a component

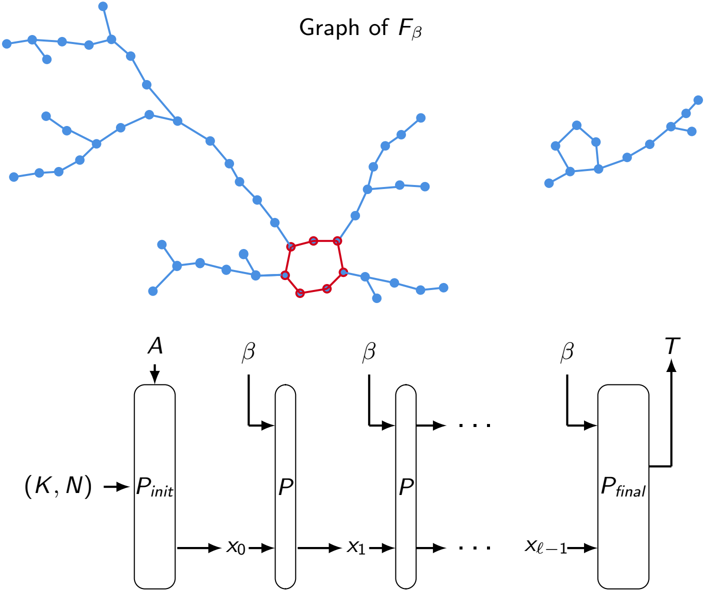
- of cycle length at most $\leq 2^{\frac{c}{2}-\nu} \rightarrow$ its cycle is **exceptionally small**:
- of size at least $\geq 2^c \times s \rightarrow$ this component is **reasonably large**;

$$p_{s,\nu} \approx \sqrt{\frac{2(1-s)}{\pi s}} 2^{c-\nu}$$

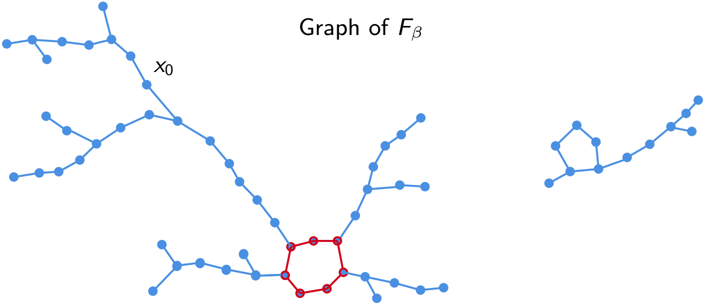
[DeLaurentis87]

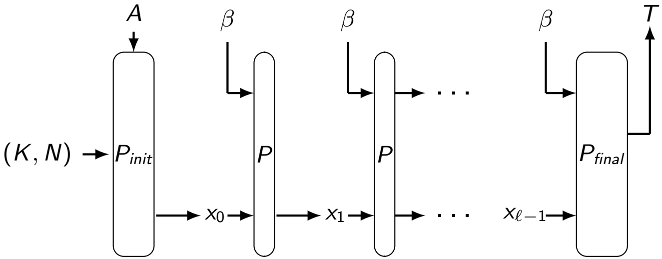
Ex: proba for $s = 65\%$ and $\nu = \frac{c}{4}$ (cycle of length $\leq 2^{\frac{c}{4}}$): $0.6 \times 2^{-\frac{c}{4}}$

Core idea of our forgery attack

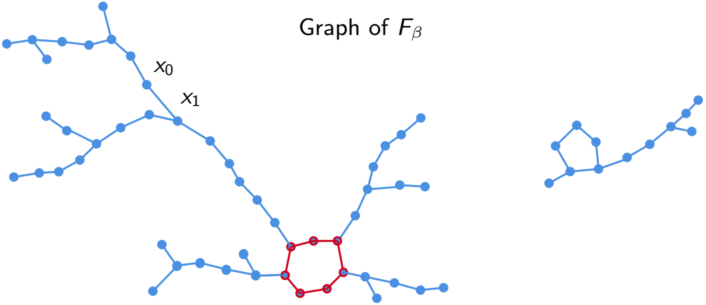


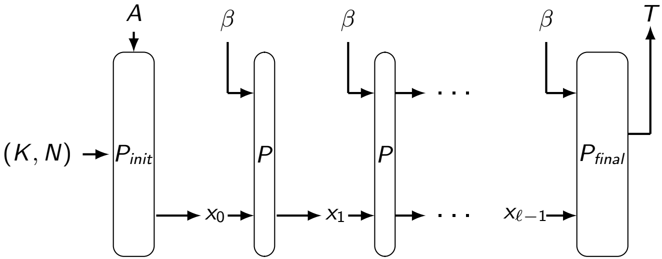
Core idea of our forgery attack



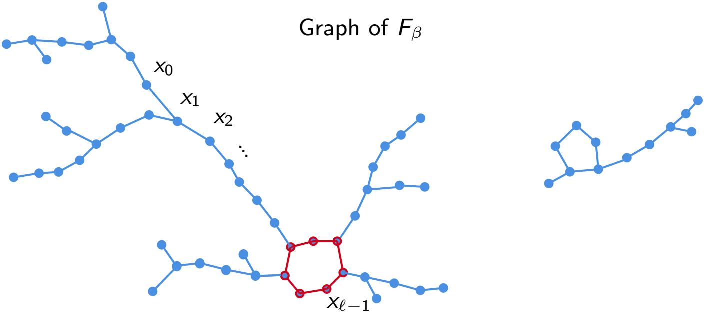


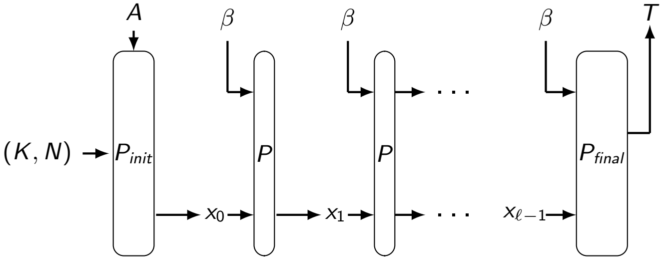
Core idea of our forgery attack



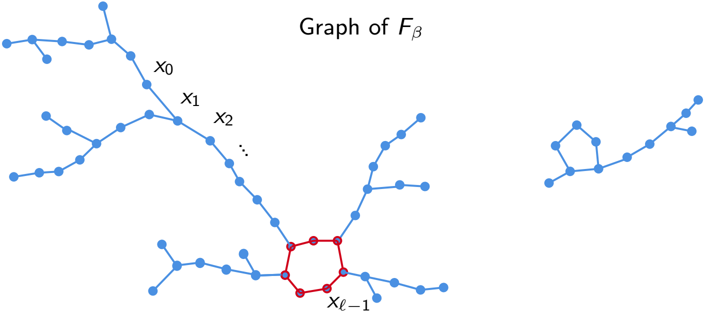


Core idea of our forgery attack



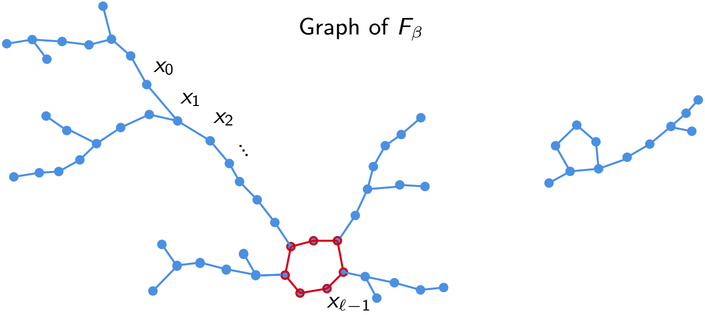


Core idea of our forgery attack



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

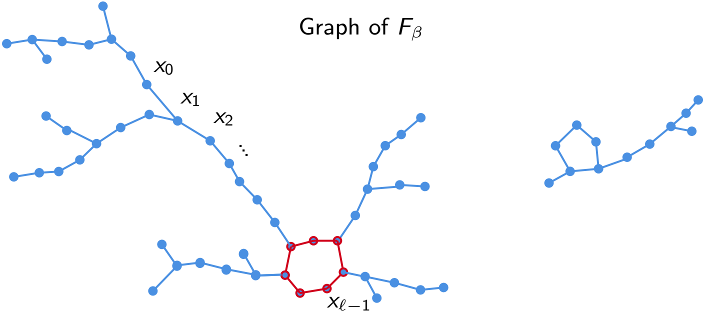
Core idea of our forgery attack



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

$\rightarrow$ Since the component is large, x_0 belongs to it with good probability (≈ 0.65)

Core idea of our forgery attack

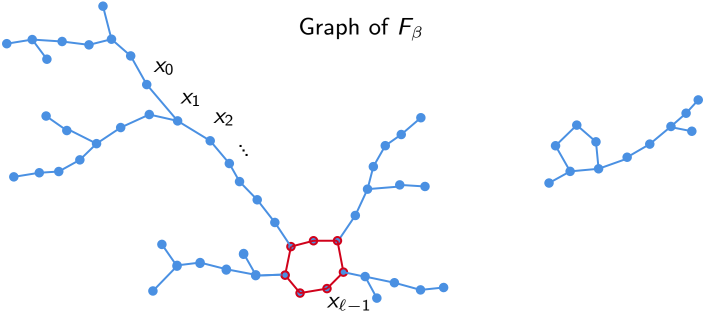


If one finds β s.t. F_β has a reasonably **large component** (say $\geq 0.65 \times 2^c$) with an exceptionnally **small cycle** (say $\leq 2^{\frac{c}{4}}$)...

$\rightarrow$ Since the component is large, x_0 belongs to it with good probability (≈ 0.65)

$\rightarrow$ If so, if ℓ is 'large enough' (say $\ell \approx 2^{\frac{c}{2}}$), $x_{\ell-1}$ is in the cycle with good probability

Core idea of our forgery attack



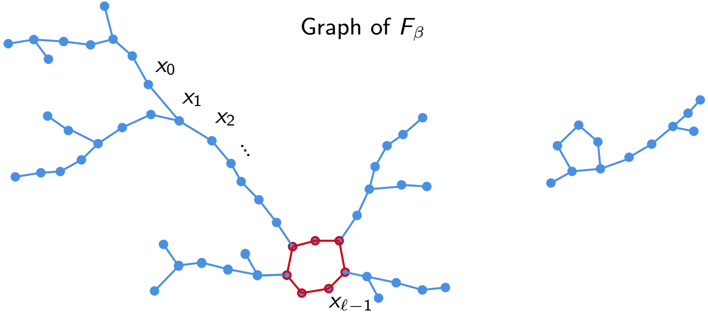
If one finds β s.t. F_β has a reasonably **large component** (say $\geq 0.65 \times 2^c$) with an exceptionnally **small cycle** (say $\leq 2^{\frac{c}{4}}$)...

$\rightarrow$ Since the component is large, x_0 belongs to it with good probability (≈ 0.65)

$\rightarrow$ If so, if ℓ is 'large enough' (say $\ell \approx 2^{\frac{c}{2}}$), $x_{\ell-1}$ is in the cycle with good probability

$\rightarrow$ If so, there are at most $2^{\frac{c}{4}}$ possible values for $x_{\ell-1}$ *i.e.* **at most $2^{\frac{c}{4}}$ possible tags**

Core idea of our forgery attack



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

$\rightarrow$ Since the component is large, x_0 belongs to it with good probability (≈ 0.65)

$\rightarrow$ If so, if ℓ is 'large enough' (say $\ell \approx 2^{\frac{c}{2}}$), $x_{\ell-1}$ is in the cycle with good probability

$\rightarrow$ If so, there are at most $2^{\frac{c}{4}}$ possible values for $x_{\ell-1}$ *i.e.* **at most $2^{\frac{c}{4}}$ possible tags**

Resulting forgery attack: try the $\leq 2^{\frac{c}{4}}$ possible values for T.

Core idea of our forgery attack

Precomputation phase
Find β s.t. F_β has a **large component** ($\geq 0.65 \times 2^c$) with an exceptionnally **small cycle** ($\leq 2^{\frac{c}{4}}$), recover this cycle

key independant

Online phase
Submit $(N, A, C = \underbrace{\beta || \cdots || \beta}_{\ell}, T)$ queries to the decryption oracle where:

- N is randomly sampled
- A is set to the empty string
- ℓ is 'big enough' ($\approx 2^{\frac{c}{2}}$)
- $T = P_{final}(\beta || x)$, for x in the small cycle

Simplified complexity analysis (precomputation phase)

Precomputation phase: Find β s.t. F_β has a **large component** ($\geq 0.65 \times 2^c$) with an exceptionnally **small cycle** ($\leq 2^{\frac{c}{4}}$), recover this cycle

Complexity analysis:

- Drawing about $1/p_{s,\nu} \approx 2^{\frac{c}{4}}$ random β's
- For each β, investigating F_β costs $\approx 2^{\frac{c}{2}}$ per β thanks to Floyd's algorithm.

The total complexity is $\approx 2^{\frac{3c}{4}}$ **applications of** P.

Simplified complexity analysis (precomputation phase)

Precomputation phase: Find β s.t. F_β has a **large component** ($\geq 0.65 \times 2^c$) with an exceptionnally **small cycle** ($\leq 2^{\frac{c}{4}}$), recover this cycle

Complexity analysis:

- Drawing about $1/p_{s,\nu} \approx 2^{\frac{c}{4}}$ random β's
- For each β, investigating F_β costs $\approx 2^{\frac{c}{2}}$ per β thanks to Floyd's algorithm.

The total complexity is $\approx 2^{\frac{3c}{4}}$ **applications of** P.

Note: the algorithm includes a test that the component is likely to be large enough.

Simplified complexity analysis (online phase)

Online phase. Submit $(N, A, C = \underbrace{\beta || \cdots || \beta}_{\ell}, T)$ queries to the decryption oracle where $T = P_{final}(\beta || x)$, x in the cycle.

Simplified complexity analysis (online phase)

Online phase. Submit $(N, A, C = \underbrace{\beta || \cdots || \beta}_{\ell}, T)$ queries to the decryption oracle where $T = P_{final}(\beta || x)$, x in the cycle.

Complexity analysis:

- x_0 belongs to the desired component with probability $s = 65\%$
- For $x_{\ell-1}$ to belong to the cycle with good probability, we set $\ell = 3 \times 2^{\frac{c}{2}}$
- We try at most $2^{\frac{c}{4}}$ values for T (at most the length of the cycle).

The total complexity is $\approx 2^{\frac{3c}{4}}$ **applications of** P.

Simplified complexity analysis (online phase)

Online phase. Submit $(N, A, C = \underbrace{\beta || \cdots || \beta}_{\ell}, T)$ queries to the decryption oracle where $T = P_{final}(\beta || x)$, x in the cycle.

Complexity analysis:

- x_0 belongs to the desired component with probability $s = 65\%$
- For $x_{\ell-1}$ to belong to the cycle with good probability, we set $\ell = 3 \times 2^{\frac{c}{2}}$
- We try at most $2^{\frac{c}{4}}$ values for T (at most the length of the cycle).

The total complexity is $\approx 2^{\frac{3c}{4}}$ **applications of** P.

Note: At the cost of a more expensive prec. phase, the complexity of this step can be brought close(r) to $2^{\frac{c}{2}}$.

Small scale experiments

- Our attack is somewhat heuristic based.

→ Ex: corroborate that the F_β behave as **random functions** in practice.

- We implemented experiments with XOODOO[12] as P.

- All our practical results match our heuristic-based results.

→ Ex: the average tail length for a random F_β matches the average tail length for a random permutation.

- We also implemented the **precomputation algorithm**.

→ We found some **valid β values** for c up to 40.

Summary of our results

Our attack

- has **total time complexity** $\leq 21 \times 2^{\frac{3c}{4}}$;
- a **probability of success** $\geq 95\%$;
- can be transformed into a **key recovery** at a negligible extra cost if P_{init} is reversible (**how**: using the plaintext);
- is applicable to the modes of `Norx` v2, KETJE, KNOT and KEYAK
- breaks the 184-bit security claim made by the designers of XOODYAK with an attack of complexity 2^{148}.

Preventing the attack

Two main features frustrate our cryptanalysis:

- **Key-dependent final phase.** (ASCON, NORX v3)

$\rightarrow$ a correct guess on $x_{\ell-1}$ cannot be transformed into a forgery

- **No outer state overwriting.** (Beetle, SPARKLE, Subterranean)

$\rightarrow$ the decryption of $\underbrace{\beta||\cdots||\beta}_{\ell}$ does not correspond to the iteration of a function

Thank you for your attention :)

Any questions?