Improving Generic Attacks Using Exceptional Functions

Xavier Bonnetain[1] Rachelle Heim Boissier[2]
Gaëtan Leurent[3] André Schrottenloher[4]

[1]Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

[2]Université Paris-Saclay, UVSQ, CNRS,
Laboratoire de mathématiques de Versailles, Versailles, France

[3]Inria, Paris, France

[4]Univ Rennes, Inria, CNRS, IRISA, Rennes, France

CRYPTO 2024

Generic attacks in symmetric cryptography

Security evaluation: classical approach

- Security proofs for modes of operation and constructions
  - Model primitives as ideal: PRF, Random Oracle
- Cryptanalysis of primitives
  - Evaluates whether concrete primitives behave like ideal model

Cryptanalysis of modes of operation and constructions

- Generic attacks target the mode without using properties of the primitives
  - Complementary to security proofs: gap between attacks and proofs
- Typical situation: birthday bound security
  - Security proof up to $2^{n/2}$ operations, with n the state size
  - Simple matching attack for simple security properties (*e.g.* collisions)
  - No matching attack for some more complex properties (*e.g.* preimage, state-recovery)

Generic attacks in symmetric cryptography

Security evaluation: classical approach

- Security proofs for modes of operation and constructions
  - Model primitives as ideal: PRF, Random Oracle
- Cryptanalysis of primitives
  - Evaluates whether concrete primitives behave like ideal model

Cryptanalysis of modes of operation and constructions

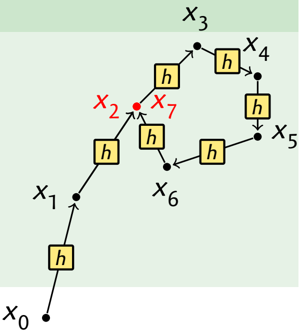
- Generic attacks target the mode without using properties of the primitives
  - Complementary to security proofs: gap between attacks and proofs
- Typical situation: birthday bound security
  - Security proof up to $2^{n/2}$ operations, with n the state size
  - Simple matching attack for simple security properties (*e.g.* collisions)
  - No matching attack for some more complex properties (*e.g.* preimage, state-recovery)

Simple example: Pollard rho

Pollard's rho

- Given a public n-bit function $h : \{0,1\}^n \to \{0,1\}^n$
- Find x, y with $h(x) = h(y)$

1. Iterate h: $x_i = h(x_{i-1})$
2. Eventually, sequence cycles
3. Detect cycle, locate collision (Floyd, Brent)

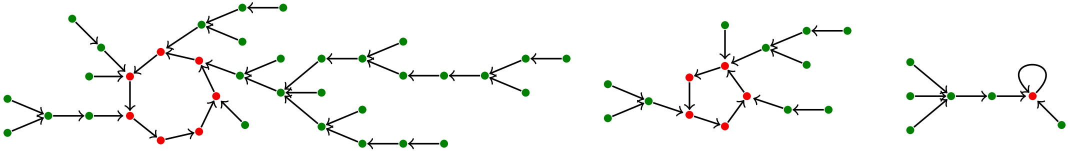


Complexity evaluation

- Assume average properties of random functions
  - Time to reach cycle (tail length) $\mathcal{O}(2^{n/2})$
  - Cycle length $\mathcal{O}(2^{n/2})$

Average properties of random functions

- Graph of a random function: trees connected to cycles

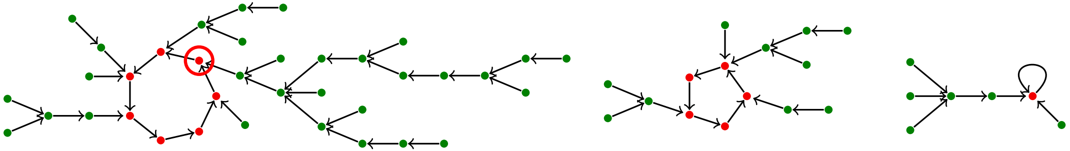


Expected properties of a random mapping over 2^n points [Flajolet & Odlyzko, EC'89]

- # Components: $n \log(2)/2$
- # Cyclic nodes: $\sqrt{\pi/2} \cdot 2^{n/2}$
- Tail length: $\sqrt{\pi/8} \cdot 2^{n/2}$
- Cycle length: $\sqrt{\pi/8} \cdot 2^{n/2}$
- Largest tree: $0.48 \cdot 2^n$
- Largest component: $0.76 \cdot 2^n$

Attacks using the giant tree

- Random functions have a giant component and a giant tree



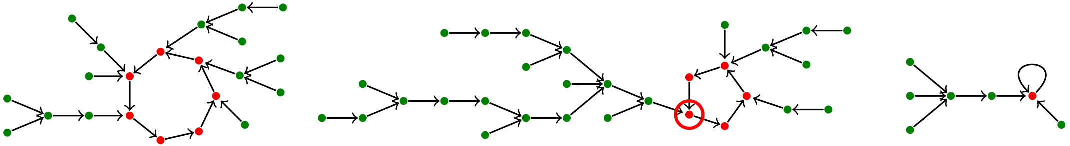
Expected properties of a random mapping over 2^n points [Flajolet & Odlyzko, EC'89]

- Largest tree: $0.48 \cdot 2^n$
- Largest component: $0.76 \cdot 2^n$

- Assume iteration of fixed public function, with secret state
- With constant probability, a random point is in the giant tree
- In particular, the first cyclic point is the root of the giant tree
  - Used in attacks against HMAC [L, Peyrin & Wang, Asiacrypt'13]

Exceptional properties of random functions

- With some probability, giant tree is connected to small cycle

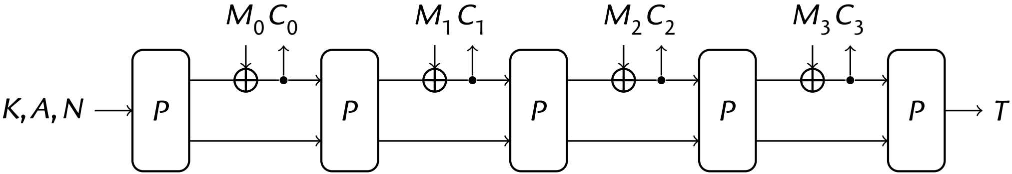


Exceptional properties of a random mapping over 2^n points [DeLaurentis, Crypto'87]

- Giant component has a cycle of length $\leq 2^{\mu}$ with probability $\Theta(2^{\mu-n/2})$

- Assume iteration of public function, with chosen parameter $h_u : \{0,1\}^n \to \{0,1\}^n$
- Find parameter β such that h_β has giant component with cycle length $\leq 2^{\mu}$
  - Complexity $2^{n-\mu}$ [Gilbert, Heim Boissier, Khati & Rotella, EC'23]
- With constant probability, a random point reaches the small cycle of h_β

Duplex Sponge AEAD

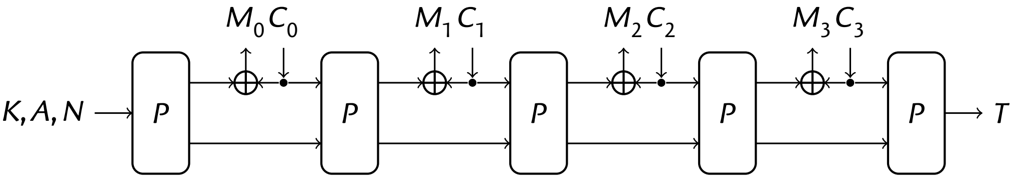


- Encryption XORs message inside state, extracts ciphertext
- Decryption replaces state with ciphertext
- Tag verification iterates public function with parameter
  - With a fixed ciphertext β, iteration of a fixed function

$$h_\beta : \{0,1\}^n \to \{0,1\}^n$$
$$x_i \mapsto x_{i+1} = P(\beta \parallel x_i)$$

  - With long ciphertext β^L, $L \geq 2^{n/2}$ final state in main cycle of h_β with high probability

Duplex Sponge AEAD

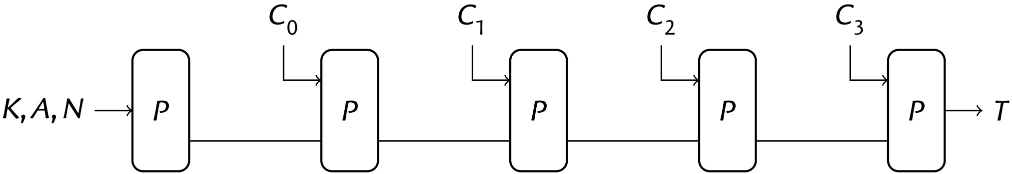


- Encryption XORs message inside state, extracts ciphertext
- Decryption replaces state with ciphertext
- Tag verification iterates public function with parameter
  - With a fixed ciphertext β, iteration of a fixed function

$$h_\beta : \{0,1\}^n \to \{0,1\}^n$$
$$x_i \mapsto x_{i+1} = P(\beta \parallel x_i)$$

  - With long ciphertext β^L, $L \geq 2^{n/2}$ final state in main cycle of h_β with high probability

Duplex Sponge AEAD

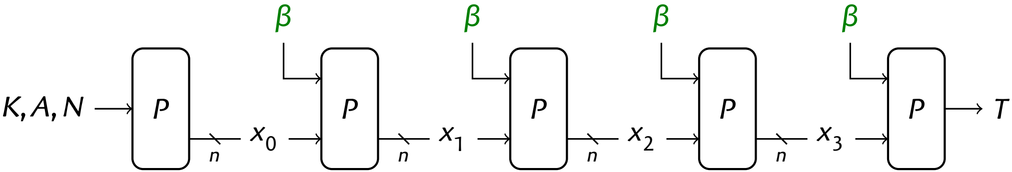


- Encryption XORs message inside state, extracts ciphertext
- Decryption replaces state with ciphertext
- Tag verification iterates public function with parameter
  - With a fixed ciphertext β, iteration of a fixed function

$$h_\beta : \{0,1\}^n \to \{0,1\}^n$$
$$x_i \mapsto x_{i+1} = P(\beta \parallel x_i)$$

  - With long ciphertext β^L, $L \geq 2^{n/2}$ final state in main cycle of h_β with high probability

Duplex Sponge AEAD

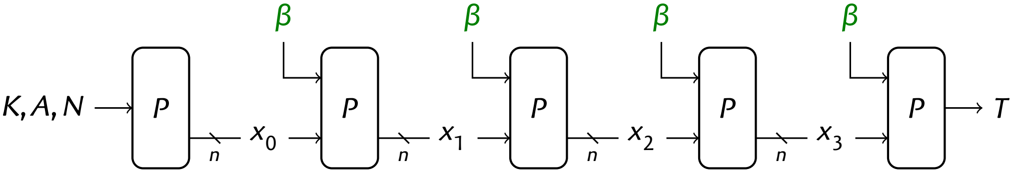


- Encryption XORs message inside state, extracts ciphertext
- Decryption replaces state with ciphertext
- Tag verification iterates public function with parameter
  - With a fixed ciphertext β, iteration of a fixed function

$$h_\beta : \{0,1\}^n \to \{0,1\}^n$$
$$x_i \mapsto x_{i+1} = P(\beta \parallel x_i)$$

  - With long ciphertext β^L, $L \geq 2^{n/2}$ final state in main cycle of h_β with high probability

Forgery attack [Gilbert, Heim Boissier, Khati & Rotella, EC'23]



0. Find cycle $\mathcal{C}$ of h_β, cycle length 2^μ *Offline*
   - Compute $T = P(\beta \parallel x^*)$ with arbitrary $x^* \in \mathcal{C}$
1. Make forgery attempt (β^L, T), with $L \geq 2^{n/2}$ *Online*
   - With high probability, final state in cycle $\mathcal{C}$
   - With probability $\approx 2^{-\mu}$, final state matches x^* and tag is valid

Using arbitrary β

- Precomputation cost $2^{n/2}$
- Cycle length $2^\mu \approx 2^{n/2}$
- Complexity $2^{n/2+\mu} = 2^n$

Using small cycle ($\mu \ll n/2$)

- Precomputation cost $2^{n-\mu}$
- Balance $2^{n-\mu}$ and $2^{n/2+\mu}$
- Complexity $2^{3n/4}$ ($\mu = n/4$)

Our results

- We extend the use of exceptional functions for cryptanalysis

1. New technique nesting exceptional functions
   - Improved attack on duplex AEAD
   - Alternative attacks on hash combiners
2. Revisit attack based on average properties of random functions, improve them using exceptional properties of random functions
   - Improved attack on hash combiners (XOR, zipper, hash-twice)

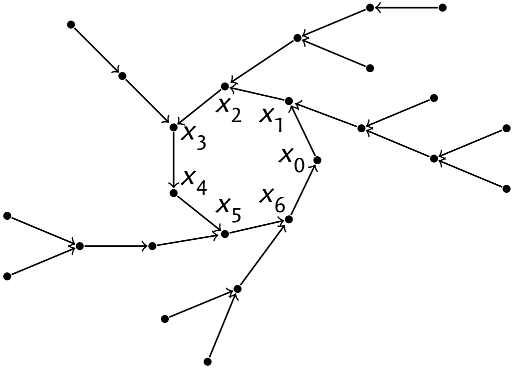
Outline

- We extend the use of exceptional functions for cryptanalysis

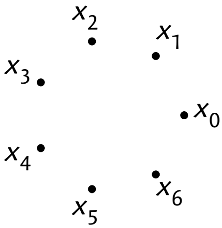
1. New technique nesting exceptional functions
   - Improved attack on duplex AEAD
   - Alternative attacks on hash combiners
2. Revisit attack based on average properties of random functions, improve them using exceptional properties of random functions
   - Improved attack on hash combiners (XOR, zipper, hash-twice)

Nesting exceptional functions

- Find β such that h_β has small main cycle
- Build function from the cycle to the cycle: $g_{\beta,\gamma} : x \mapsto h_\beta^L(h_\gamma(x))$, with $L \geq 2^{n/2}$
  - h_γ randomizes state
  - Iteration of h_β reaches main cycle with high probability



Graph of h_β

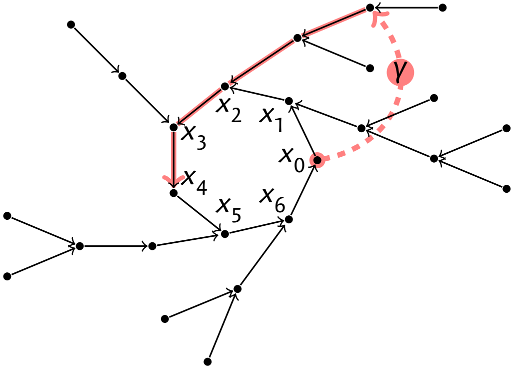


Graph of $g_{\beta,\gamma}$

- Find γ such that $g_{\beta,\gamma'}$ has small main cycle

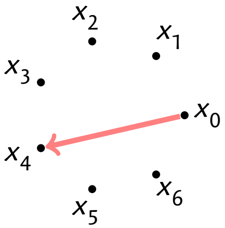
Nesting exceptional functions

- Find β such that h_β has small main cycle
- Build function from the cycle to the cycle: $g_{\beta,\gamma} : x \mapsto h^L_\beta(h_\gamma(x))$, with $L \geq 2^{n/2}$
  - h_γ randomizes state
  - Iteration of h_β reaches main cycle with high probability



Graph of h_β

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

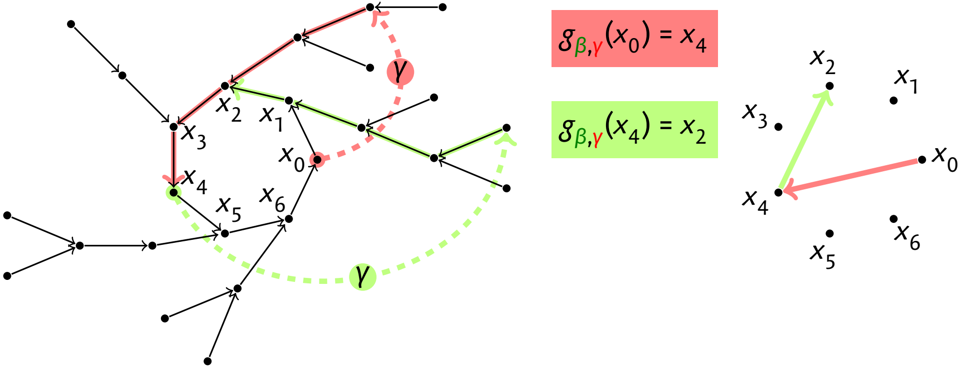


Graph of $g_{\beta,\gamma}$

- Find γ such that $g_{\beta,\gamma}$, has small main cycle

Nesting exceptional functions

- Find β such that h_β has small main cycle
- Build function from the cycle to the cycle: $g_{\beta,\gamma} : x \mapsto h_\beta^L(h_\gamma(x))$, with $L \geq 2^{n/2}$
  - h_γ randomizes state
  - Iteration of h_β reaches main cycle with high probability



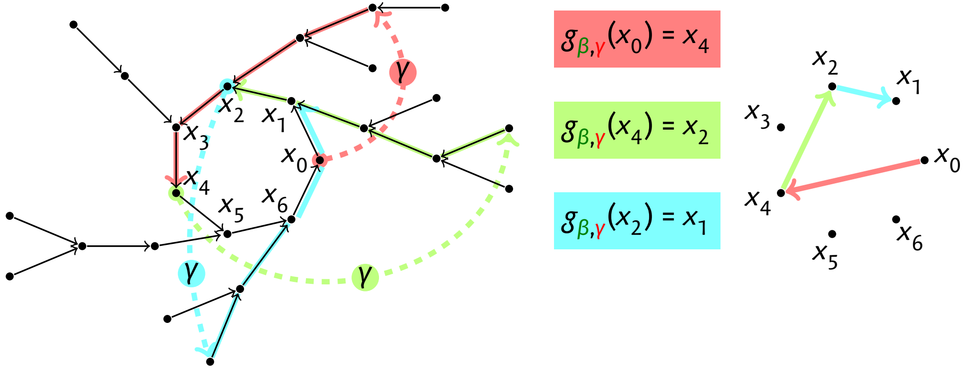
Graph of h_β

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

- Find γ such that $g_{\beta,\gamma}$, has small main cycle

Nesting exceptional functions

- Find β such that h_β has small main cycle
- Build function from the cycle to the cycle: $g_{\beta,\gamma} : x \mapsto h_\beta^L(h_\gamma(x))$, with $L \geq 2^{n/2}$
  - h_γ randomizes state
  - Iteration of h_β reaches main cycle with high probability

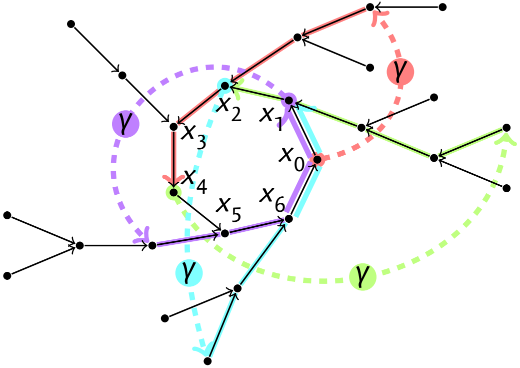


Graph of h_β — Graph of $g_{\beta,\gamma}$

- Find γ such that $g_{\beta,\gamma}$, has small main cycle

Nesting exceptional functions

- Find β such that h_β has small main cycle
- Build function from the cycle to the cycle: $g_{\beta,\gamma} : x \mapsto h_\beta^L(h_\gamma(x))$, with $L \geq 2^{n/2}$
  - h_γ randomizes state
  - Iteration of h_β reaches main cycle with high probability



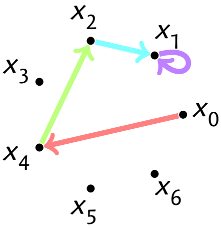
Graph of h_β

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

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

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

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

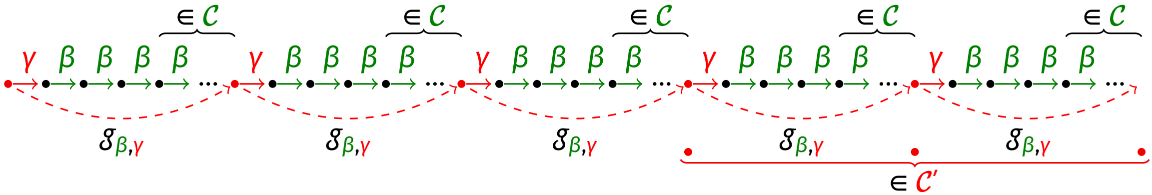


Graph of $g_{\beta,\gamma}$

- Find γ such that $g_{\beta,\gamma'}$ has small main cycle

Improved forgery attack

- Build ciphertext $(\gamma \parallel \beta^L)^\Lambda$, with $L \geq 2^{n/2}$, $\Lambda > 2^{\mu/2}$



0 Find β such that h_β has cycle $\mathcal{C}$ of length 2^μ — $2^{n-\mu}$

Find γ such that $g_{\beta,\gamma}$ has cycle $\mathcal{C}'$ of length 2^ν — $2^{n/2} \times 2^{\mu-\nu}$

- Compute $T = P(\beta \parallel x^*)$ with arbitrary $x^* \in \mathcal{C}'$

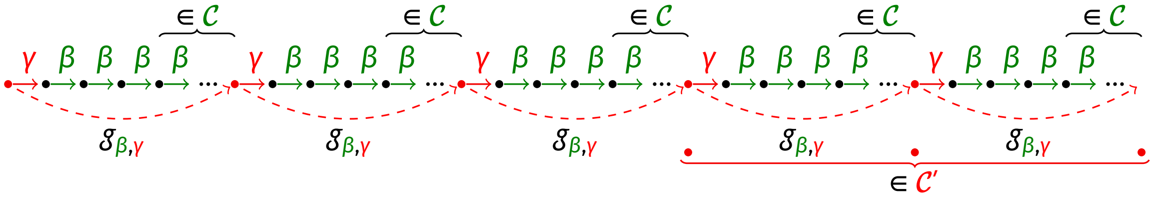
1 Make forgery attempt $(\gamma \parallel \beta^L)^\Lambda$, with $L \geq 2^{n/2}$, $\Lambda > 2^{\mu/2}$ — $2^{n/2+\mu/2}$

- With high probability, final state in cycle $\mathcal{C}'$
- With probability $\approx 2^{-\mu}$, final state matches x^* and tag is valid — $\times 2^\nu$

- Balance $2^{n-\mu}$, $2^{n/2} \times 2^{\mu-\nu}$, $2^{n/2+\mu/2} \times 2^\nu$
- Optimal complexity: $2^{5n/7} \approx 2^{0.71n}$ — $\mu = 2n/7, \nu = n/14$

Improved forgery attack

- Build ciphertext $(\gamma \parallel \beta^L)^\Lambda$, with $L \geq 2^{n/2}$, $\Lambda > 2^{\mu/2}$



0 Find β such that h_β has cycle $\mathcal{C}$ of length 2^μ — $2^{n-\mu}$

Find γ such that $g_{\beta,\gamma}$ has cycle $\mathcal{C}'$ of length 2^ν — $2^{n/2} \times 2^{\mu-\nu}$

- Compute $T = P(\beta \parallel x^*)$ with arbitrary $x^* \in \mathcal{C}'$

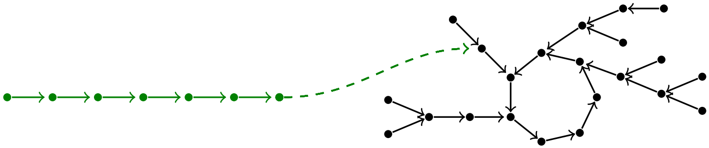
1 Make forgery attempt $(\gamma \parallel \beta^L)^\Lambda$, with $L \geq 2^{n/2}$, $\Lambda > 2^{\mu/2}$ — $2^{n/2+\mu/2}$

- With high probability, final state in cycle $\mathcal{C}'$
- With probability $\approx 2^{-\mu}$, final state matches x^* and tag is valid — $\times 2^\nu$

- Balance $2^{n-\mu}$, $2^{n/2} \times 2^{\mu-\nu}$, $2^{n/2+\mu/2} \times 2^\nu$
- Optimal complexity: $2^{5n/7} \approx 2^{0.71n}$ — $\mu = 2n/7, \nu = n/14$

More precomputation

[Peyrin&Wang, EC'14]



0 Find β such that h_β has cycle $\mathcal{C}$ of length 2^μ — $2^{n-\mu}$

- Precompute and store 2^t points in the graph of h_β — 2^t
- A chain β^L can be evaluated with only 2^{n-t} operations

Find γ such that $g_{\beta,\gamma}$ has cycle $\mathcal{C}'$ of length 2^ν — $2^{n-t} \times 2^{\mu-\nu}$

- Compute $T = P(\beta \parallel x^*)$ with arbitrary $x^* \in \mathcal{C}'$

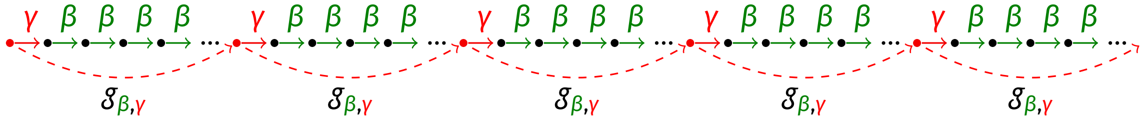
1 Make forgery attempt $(\gamma \parallel \beta^L)^\Lambda$, with $L \geq 2^{n/2}$, $\Lambda > 2^{\mu/2}$ — $2^{n/2+\mu/2}$

- With high probability, final state in cycle $\mathcal{C}'$
- With probability $\approx 2^{-\mu}$, final state matches x^* and tag is valid — $\times 2^\nu$

- Balance 2^t, $2^{n-\mu}$, $2^{n-t} \times 2^{\mu-\nu}$, $2^{n/2+\mu/2} \times 2^\nu$
- Optimal complexity: $2^{2n/3} \approx 2^{0.67n}$ — $t = 2n/3$, $\mu = n/3$, $\nu = 0$

Nesting exceptional functions: summary

- Assume iteration of public function, with chosen parameter $h_u : \{0,1\}^n \to \{0,1\}^n$
- With $2^{2n/3}$ operations, construct sequence of $2^{2n/3}$ parameters such that final state is a known fixed value with high probability ($v = 0$)



Applications

- Forgery attack against duplex AEAD with complexity $2^{2n/3}$ (previously $2^{3n/4}$)
  - Does not violate security proof, but some proposals had wrong parameters
- Provides alternative attacks on HMAC, zipper hash, hash twice, …
  - Less efficient than best known attacks *(improved attacks in next section)*

Outline

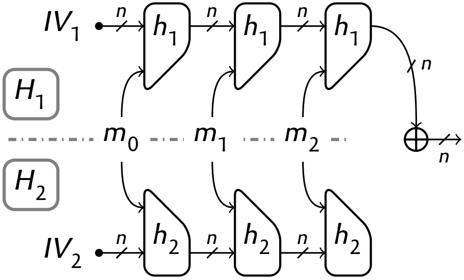
- We extend the use of exceptional functions for cryptanalysis

1. New technique nesting exceptional functions
   - Improved attack on duplex AEAD
   - Alternative attacks on hash combiners
2. Revisit attack based on average properties of random functions, improve them using exceptional properties of random functions
   - Improved attack on hash combiners (XOR, zipper, hash-twice)

Preimage attack against Xor combiner

[L & Wang, EC'15]

$H(M) = H_1(M) \oplus H_2(M)$



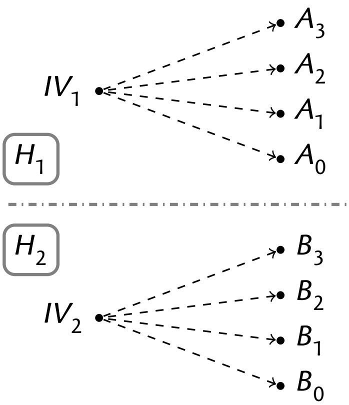
Strategy:

1. Structure to control H_1 and H_2 independently:
   - Sets of states $\mathcal{A} = \{A_j\}$, $\mathcal{B} = \{B_k\}$
   - Set of messages $\{\mathbf{M}_{jk}\}$ with
     $$h_1^*(\mathbf{M}_{jk}) = A_j$$
     $$h_2^*(\mathbf{M}_{jk}) = B_k$$
2. Preimage search for $\overline{H}$:
   - For random blocks w, match $\{h_1(A_j, w)\}$ and $\{h_2(B_k, w) \oplus \overline{H}\}$
   - If there is a match (j, k): Get $\mathbf{M}_{jk}$, preimage is $M = \mathbf{M}_{jk} \parallel w$
   - Complexity $\mathcal{O}(2^n / \min\{|\mathcal{A}|, |\mathcal{B}|\})$

Preimage attack against Xor combiner

[L & Wang, EC'15]

$$H(M) = H_1(M) \oplus H_2(M)$$



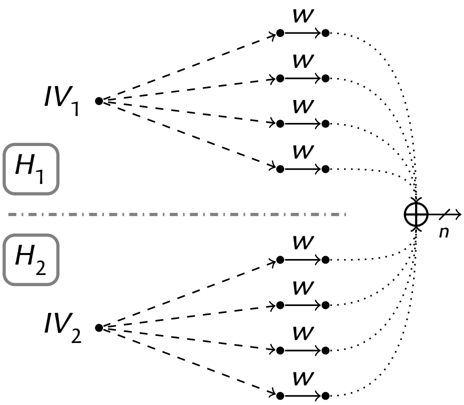
Strategy:

1. Structure to control H_1 and H_2 independently:
   - Sets of states $\mathcal{A} = \{A_j\}$, $\mathcal{B} = \{B_k\}$
   - Set of messages $\{\mathbf{M}_{jk}\}$ with
     $$h_1^*(\mathbf{M}_{jk}) = A_j$$
     $$h_2^*(\mathbf{M}_{jk}) = B_k$$
2. Preimage search for $\overline{H}$:
   - For random blocks w, match $\{h_1(A_j, w)\}$ and $\{h_2(B_k, w) \oplus \overline{H}\}$
   - If there is a match (j, k): Get $\mathbf{M}_{jk}$, preimage is $M = \mathbf{M}_{jk} \parallel w$
   - Complexity $\mathcal{O}(2^n / \min\{|\mathcal{A}|, |\mathcal{B}|\})$

Preimage attack against Xor combiner

[L & Wang, EC'15]

$H(M) = H_1(M) \oplus H_2(M)$



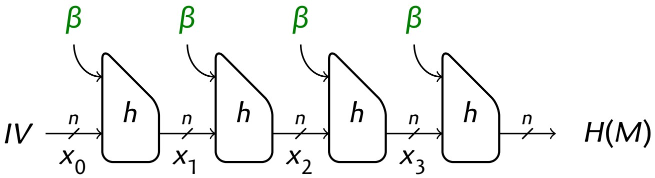
Strategy:

1. Structure to control H_1 and H_2 independently:
   - Sets of states $\mathcal{A} = \{A_j\}$, $\mathcal{B} = \{B_k\}$
   - Set of messages $\{\mathbf{M}_{jk}\}$ with
     $$h_1^*(\mathbf{M}_{jk}) = A_j$$
     $$h_2^*(\mathbf{M}_{jk}) = B_k$$
2. Preimage search for $\overline{H}$:
   - For random blocks w, match $\{h_1(A_j, w)\}$ and $\{h_2(B_k, w) \oplus \overline{H}\}$
   - If there is a match (j, k): Get $\mathbf{M}_{jk}$, preimage is $M = \mathbf{M}_{jk} \parallel w$
   - Complexity $\mathcal{O}(2^n / \min\{|\mathcal{A}|, |\mathcal{B}|\})$

Cycle-based attack

- Hard part: build structure to control H_1 and H_2 independently
- Several techniques have been proposed (interchange, deep iterates, multicycles, ...)

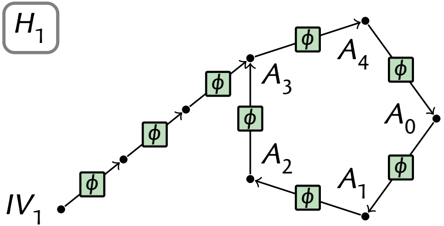
- In this talk: alternative presentation of "multicycles" [Bao, Wang, Guo, Gu, C'17]

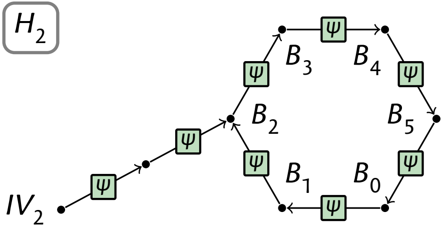


- Using a long message repeating a fixed block $M = \beta^\lambda$, we iterate fixed functions:

$$\phi : x \mapsto h_1(x, \beta)$$
$$\psi : x \mapsto h_2(x, \beta)$$

Cycle-based attack





- Use cyclic nodes as end-point:
  - $\mathcal{A} = H_1$ cycle, length 2^{μ_1}
  - $\mathcal{B} = H_2$ cycle, length 2^{μ_2}
- With suitable naming, for λ large enough:

$$h_1^*(\beta^\lambda) = A_{\lambda \bmod 2^{\mu_1}} \qquad h_2^*(\beta^\lambda) = B_{\lambda \bmod 2^{\mu_2}}$$

- To reach (A_j, B_k), use Chinese Remainder Theorem

$$\begin{cases} h_1^*(\beta^\lambda) = A_j \\ h_2^*(\beta^\lambda) = B_k \end{cases} \iff \begin{cases} \lambda \bmod 2^{\mu_1} = i \\ \lambda \bmod 2^{\mu_2} = j \end{cases}$$

  - Note: μ_1, μ_2 are not integers
  - λ uniformly distributed in range of size $2^{\mu_1+\mu_2}$

Complexity analysis

Preimage search, with maximal length 2^ℓ

- For random w, match $\{h_1(A_j, w)\}$ and $\{h_2(B_k, w)) \oplus \overline{H}\}$ — Complexity 2^μ
- If there is a match (j, k), Find λ such that $h_1^*(\beta^\lambda) = A_j$, $h_2^*(\beta^\lambda) = B_k$ using CRT — Proba $2^{\mu_1+\mu_2-n}$
- If $\lambda < 2^\ell$, return $\beta^\lambda \parallel w$ — Proba $2^{\ell-\mu_1-\mu_2}$

- $2^{n-\ell}$ iterations, total complexity $2^{n-\ell+\mu}$

Using arbitrary β

- Cycle length $\mu_1 \approx \mu_2 \approx n/2$
- Balance $2^{n-\ell+\mu}$ and 2^ℓ
- Optimal tradeoff $\ell = 3n/4$
- Complexity $2^{3n/4} = 2^{0.75n}$

Using small cycles $\mu \ll n/2$

- Precomputation cost $2^{3n/2-2\mu}$
- Balance $2^{3n/2-2\mu}$, $2^{n-\ell+\mu}$ and 2^ℓ
- Optimal tradeoff $\ell = 7n/10$, $\mu = 2n/5$
- Complexity $2^{7n/10} = 2^{0.7n}$

Complexity analysis

Preimage search, with maximal length 2^ℓ

- For random w, match $\{h_1(A_j, w)\}$ and $\{h_2(B_k, w)) \oplus \overline{H}\}$ — Complexity 2^μ
- If there is a match (j, k),
  Find λ such that $h_1^*(\beta^\lambda) = A_j$, $h_2^*(\beta^\lambda) = B_k$ using CRT — Proba $2^{\mu_1+\mu_2-n}$
- If $\lambda < 2^\ell$, return $\beta^\lambda \parallel w$ — Proba $2^{\ell-\mu_1-\mu_2}$

- $2^{n-\ell}$ iterations, total complexity $2^{n-\ell+\mu}$

Using arbitrary β

- Cycle length $\mu_1 \approx \mu_2 \approx n/2$
- Balance $2^{n-\ell+\mu}$ and 2^ℓ
- Optimal tradeoff $\ell = 3n/4$
- Complexity $2^{3n/4} = 2^{0.75n}$

Using small cycles $\mu \ll n/2$

- Precomputation cost $2^{3n/2-2\mu}$
- Balance $2^{3n/2-2\mu}$, $2^{n-\ell+\mu}$ and 2^ℓ
- Optimal tradeoff $\ell = 7n/10$, $\mu = 2n/5$
- Complexity $2^{7n/10} = 2^{0.7n}$

Hash combiners: summary

- Exceptional functions with small main cycle improve the "multicycles" technique

	Techniques	Complexity	Ref
Preimage on XOR combiner			
	Interchange + Multicycles	$2^{11n/18} \approx 2^{0.611n}$	[JC:BDGLW20]
	Interchange + Multicycles + Small cycles	$2^{3n/5} = 2^{0.6n}$	New
Second-preimage on zipper hash			
	Multicollisions + Multicycles	$2^{3n/5} = 2^{0.6n}$	[C:BWGG17]
	Multicollisions + Multicycles + Small cycles	$2^{7n/12} = 2^{0.583n}$	New
Second-preimage on hash-twice			
	Interchange + Multicycles	$2^{13n/22} = 2^{0.591n}$	[JC:BDGLW20]
	Interchange + Multicycles + Small cycles	$2^{15n/26} = 2^{0.577n}$	New
All	Lower bound (security proof)	$2^{n/2} = 2^{0.5n}$	

- Bonus result: quantum 2nd-preimage on hash-twice (not using exceptional functions)