Generic Attacks against MAC algorithms

Gaëtan Leurent

Inria Rocquencourt, France

ASK 2015

Confidentiality and authenticity

- Cryptography has two main objectives:

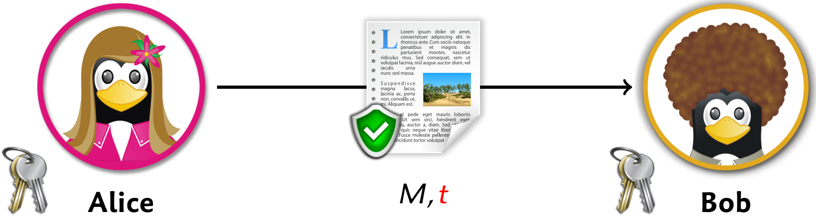
  Confidentiality keeping the message secret

  Authenticity making sure the message is authentic

- Authenticity is often more important than confidentiality
  - Email signature
  - Software update
  - Credit cards
  - Sensor networks
  - Remote control (*e.g.* garage door, car)
  - Remote access (*e.g.* password authentication)

- Authenticity achieved with signatures (asymmetric), or MACs (symmetric)

Message Authentication Codes

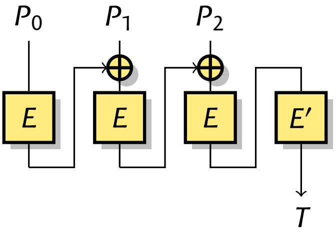


- Alice sends a message to Bob
- Bob wants to authenticate the message.
- Alice uses a key k to compute a tag: $t = \mathrm{MAC}_k(M)$
- Bob verifies the tag with the same key k: $t \stackrel{?}{=} \mathrm{MAC}_k(M)$

Security notions

- Key-recovery: given access to a MAC oracle, extract the key
- Forgery: given access to a MAC oracle, forge a valid pair
  - For a message chosen by the adversary: existential forgery
  - For a challenge given to the adversary: universal forgery
- Distinguishing games:
  - Distinguish $\mathrm{MAC}_k^{\mathcal{H}}$ from a PRF: distinguishing-R
    e.g. distinguish HMAC from a PRF
  - Distinguish $\mathrm{MAC}_k^{\mathcal{H}}$ from $\mathrm{MAC}_k^{\mathrm{PRF}}$: distinguishing-H
    e.g. distinguish HMAC-SHA1 from HMAC-PRF

CBC-MAC



- One of the first MAC [NIST, ANSI, ISO, '85?]
- Designed by practitioners, to be used with DES
- Based on CBC encryption mode
- Security proof [Bellare, Kilian & Rogaway '94]

Security of modes of operations

- Initially, security of CBC-MAC-DES was an assumption
- To reduce the number of assumptions,
  study the block cipher and the mode independently

1. Security proof for the mode
   - Assume that the block cipher is good, prove that the MAC is good
   - Lower bound on the security of the mode
2. Cryptanalysis of the block cipher
   - Try to show non-random behavior
3. Generic attacks for the mode
   - Attack that work for any choice of the block cipher
   - Upper bound on the security of the mode

Security of modes of operations

- Initially, security of CBC-MAC-DES was an assumption
- To reduce the number of assumptions, study the block cipher and the mode independently

1. Security proof for the mode
   - Assume that the block cipher is good, prove that the MAC is good
   - Lower bound on the security of the mode
2. Cryptanalysis of the block cipher
   - Try to show non-random behavior
3. Generic attacks for the mode
   - Attack that work for any choice of the block cipher
   - Upper bound on the security of the mode

Security of modes of operations

- Initially, security of CBC-MAC-DES was an assumption
- To reduce the number of assumptions,
  study the block cipher and the mode independently

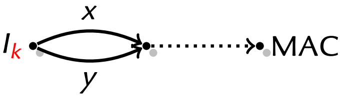
1. Security proof for the mode
   - Assume that the block cipher is good, prove that the MAC is good
   - Lower bound on the security of the mode
2. Cryptanalysis of the block cipher
   - Try to show non-random behavior
3. Generic attacks for the mode
   - Attack that work for any choice of the block cipher
   - Upper bound on the security of the mode

Security of modes of operations

- Initially, security of CBC-MAC-DES was an assumption
- To reduce the number of assumptions,
  study the block cipher and the mode independently

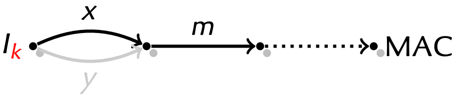
1. Security proof for the mode
   - Assume that the block cipher is good, prove that the MAC is good
   - Lower bound on the security of the mode
2. Cryptanalysis of the block cipher
   - Try to show non-random behavior
3. Generic attacks for the mode
   - Attack that work for any choice of the block cipher
   - Upper bound on the security of the mode

Generic Attack against Iterated Deterministic MACs



1. Find internal collisions [Preneel & van Oorschot '95]
   - Query $2^{n/2}$ random short messages
   - 1 internal collision expected, detected in the output
2. Query $t = \mathrm{MAC}(x \parallel m)$
3. $\left(y \parallel m, t\right)$ is a forgery

Generic Attack against Iterated Deterministic MACs

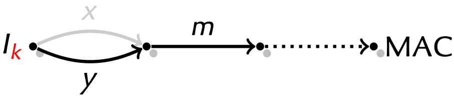


1. Find internal collisions **[Preneel & van Oorschot '95]**
   - Query $2^{n/2}$ random short messages
   - 1 internal collision expected, detected in the output

2. Query $t = \mathrm{MAC}(x \,\|\, m)$

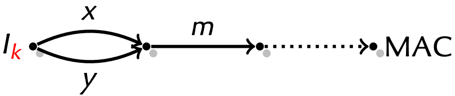
3. $\left(y \,\|\, m, t\right)$ is a forgery

Generic Attack against Iterated Deterministic MACs



1. Find internal collisions [Preneel & van Oorschot '95]
   - Query $2^{n/2}$ random short messages
   - 1 internal collision expected, detected in the output
2. Query $t = \mathrm{MAC}(x \parallel m)$
3. $\left(y \parallel m, t\right)$ is a forgery

Generic Attack against Iterated Deterministic MACs



1. Find internal collisions [Preneel & van Oorschot '95]
   - Query $2^{n/2}$ random short messages
   - 1 internal collision expected, detected in the output
2. Query $t = \mathrm{MAC}(x \,\|\, m)$
3. $(y \,\|\, m, t)$ is a forgery

Problem

- CBC-MAC with DES is unsafe after 2^{32} queries

Security Proofs

What's a security proof?

- $\mathrm{Adv}^{\mathrm{prf}}_{\text{CBC-}F}(q,t) \leq \mathrm{Adv}^{\mathrm{prp}}_{F}(mq, t + O(mqn)) + \frac{q^2 m^2}{2^{n-1}}$
- Bound on the success probability of an adversary against the MAC
  - q number of queries
  - t time
  - m max query length
- "If DES is a secure PRF, then CBC-MAC-DES is a secure PRF"

Limitations

- Birthday-bound security
  - Bound meaningless when $mq \approx 2^{n/2}$
- No information on security degradation after the birthday bound
  - Usually assumed that key-recovery attacks require more...

Remaining of this talk

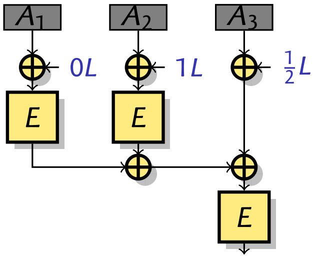
MAC security is well understood

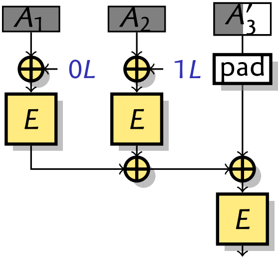
- Good MAC constructions have birthday bound security proof
- We have a generic existential forgery attack with birthday complexity

Or is it?

- Different MACs have different security loss after the birthday bound!
- We need to study generic attack to understand the security of modes

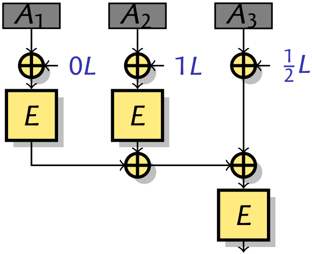
PMAC

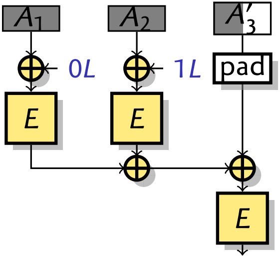




- PMAC: parallelisable block-cipher based MAC [Black & Rogaway '02]
  - Uses secret offsets to the block cipher input: $L = E_k(0)$

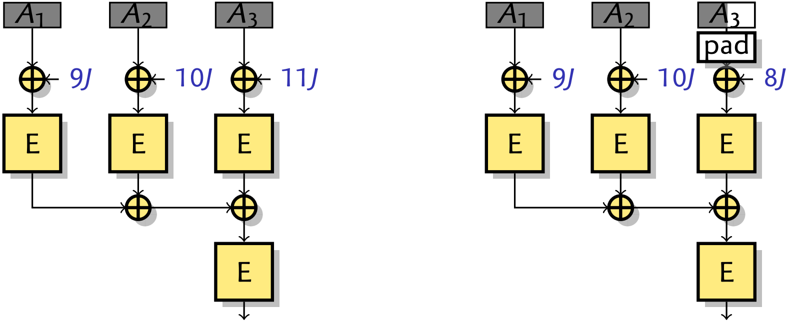
PMAC





- Collision attack: two sets of messages [Lee & al '06]
- $A_x = [x]$, $|x| = 128$
  - Full block
  - $\mathrm{MAC}(A_x) = E([x] \oplus \frac{1}{2}L)$
- $B_y = [y]$, $|y| < 128$
  - Partial block
  - $\mathrm{MAC}(B_y) = E(\mathrm{pad}([y]))$
- Collision (A_x, B_y)?
  - The MAC collide iff $[x] \oplus \frac{1}{2}L = \mathrm{pad}([y])$
  - Deduce L
  - Universal forgery attack

AEZ



- AEZ uses a variant of PMAC [Hoang, Krovetz & Rogaway '15]

- A collision gives J: $[x] \oplus 9J = \text{pad}([y]) \oplus 8J$
- Key derivation (AEZ v2) $J = E_0(k)$
- Collisions reveal the master key! [FLS, AC'15]

Security of block cipher based MACs

Proofs

CBC-MAC, PMAC, and AEZ have security proofs
up to the birthday bound

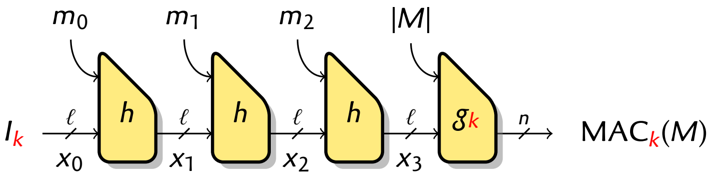
Attacks

Effect of collision attacks with $2^{n/2}$ queries

- CBC-MAC: almost universal forgeries [Jia & al '09]
- PMAC: universal forgeries
- AEZ: key recovery

Outline

Hash-based MACs



- ℓ-bit chaining value
- n-bit output
- k-bit key

we focus on $\ell = n = k$

- Key-dependant initial value I_k
- Unkeyed compression function h
- Key-dependant finalization, with message length g_k

HMAC

- HMAC has been designed by Bellare, Canetti, and Krawczyk in 1996
- Standardized by ANSI, IETF, ISO, NIST.
- Used in many applications:
  - To provide authentication:
    - SSL, IPSEC, ...
  - To provide identification:
    - Challenge-response protocols
    - CRAM-MD5 authentication in SASL, POP3, IMAP, SMTP, ...
  - For key-derivation:
    - HMAC as a PRF in IPsec
    - HMAC-based PRF in TLS

Security of hash-based MACS

- Security proofs up to the birthday bound
- Generic attacks based on collisions
  - Proof is tight for some security notions
    - Existential forgery
    - Distinguishing-R
- What is the remaining security above the birthday bound?
  - Generic distinguishing-H attack?
  - Generic state-recovery attack?
  - Generic universal forgery attack?
  - Generic key-recovery attack?

Outline

Bibliography

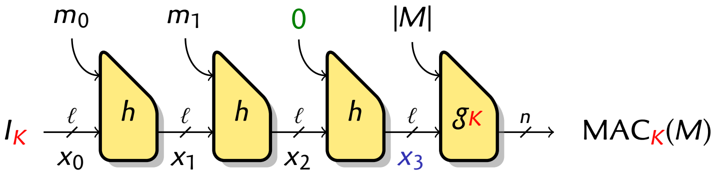
Y. Naito, Y. Sasaki, L. Wang, K. Yasuda
Generic State-Recovery and Forgery Attacks on ChopMD-MAC and on NMAC/HMAC
IWSEC 2013

G. Leurent, T. Peyrin, L. Wang
New Generic Attacks against Hash-Based MACs
ASIACRYPT 2013

I. Dinur, G. Leurent
Improved Generic Attacks against Hash-Based MACs and HAIFA
CRYPTO 2014

Multi-collision based attack

[Naito, Sasaki, Wang & Yasuda '13]



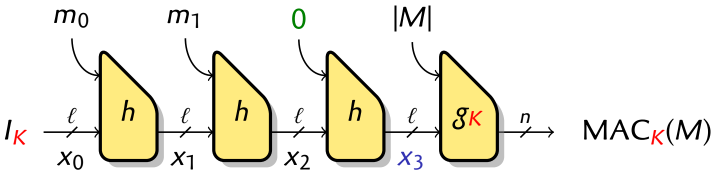
- Using a fixed message block, we apply a fixed function
- Starting point and ending point unknown because of the key

Can we detect properties of the function $h_0 : x \mapsto h(x, 0)$?

- Use bias in the output of the compression function
  - Some outputs are more likely than others
  - With $2^{\ell-\varepsilon}$ work, find a value x^* with ℓ preimages (offline)
- How to detect when this state is reached?

Multi-collision based attack

[Naito, Sasaki, Wang & Yasuda '13]



- Using a fixed message block, we apply a fixed function
- Starting point and ending point unknown because of the key

Can we detect properties of the function $h_0 : x \mapsto h(x, 0)$?

- Use bias in the output of the compression function
  - Some outputs are more likely than others
  - With $2^{\ell-\varepsilon}$ work, find a value x^* with ℓ preimages (offline)
- How to detect when this state is reached?

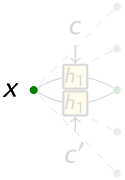
Building filters

Filters to compare online and online states

Test whether the state reached after processing M is equal to x

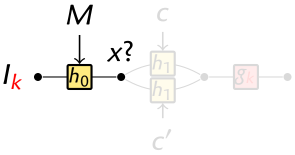
- Collisions are preserved by the finalization (for same-length messages)

1 Find a collision: $h(x, c) = h(x, c')$



Offline Structure

2 $\mathrm{MAC}(M \| c) \stackrel{?}{=} \mathrm{MAC}(M \| c')$



Online Structure

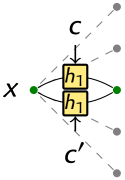
Building filters

Filters to compare online and online states

Test whether the state reached after processing M is equal to x

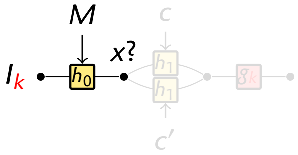
- Collisions are preserved by the finalization (for same-length messages)

1 Find a collision: $h(x,c) = h(x,c')$



Offline Structure

2 $\mathrm{MAC}(M \| c) \stackrel{?}{=} \mathrm{MAC}(M \| c')$



Online Structure

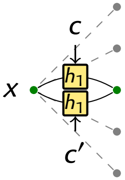
Building filters

Filters to compare online and online states

Test whether the state reached after processing M is equal to x

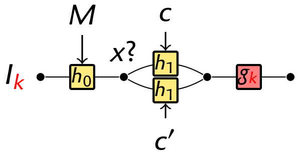
- Collisions are preserved by the finalization (for same-length messages)

1 Find a collision: $h(x,c) = h(x,c')$



Offline Structure

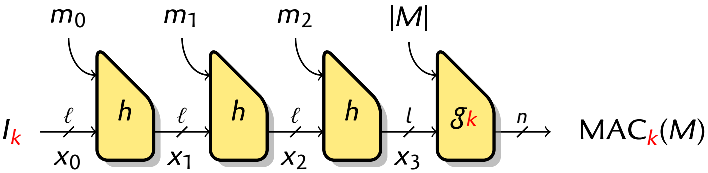
2 $\mathrm{MAC}(M \| c) \stackrel{?}{=} \mathrm{MAC}(M \| c')$



Online Structure

First state-recovery attack

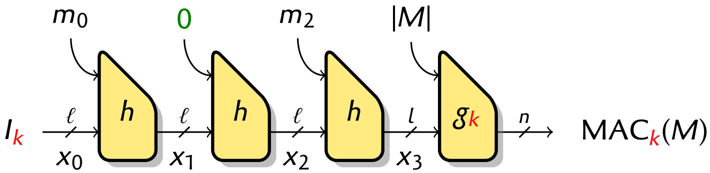
[Naito, Sasaki, Wang & Yasuda '13]



1. Fix a message block $m_1 = [0]$.
   With $2^{\ell-\varepsilon}$ work, find a value x^* with ℓ preimages
2. Find a collision $h(x^*, c) = h(x^*, c')$
3. For random m_0, compare $\mathrm{MAC}(m_0 \| [0] \| c)$ and $\mathrm{MAC}(m_0 \| [0] \| c')$
   If they are equal, $x_2 = x^*$

First state-recovery attack

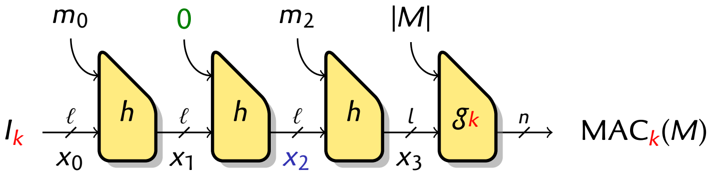
[Naito, Sasaki, Wang & Yasuda '13]



1. Fix a message block $m_1 = [0]$.
   With $2^{\ell-\varepsilon}$ work, find a value x^* with ℓ preimages
2. Find a collision $h(x^*, c) = h(x^*, c')$
3. For random m_0, compare $\mathrm{MAC}(m_0 \,\|\, [0] \,\|\, c)$ and $\mathrm{MAC}(m_0 \,\|\, [0] \,\|\, c')$
   If they are equal, $x_2 = x^*$

First state-recovery attack

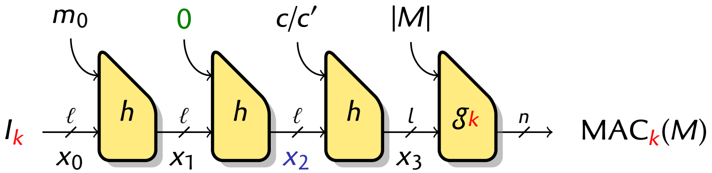
[Naito, Sasaki, Wang & Yasuda '13]



1. Fix a message block $m_1 = [0]$.
   With $2^{\ell-\varepsilon}$ work, find a value x^* with ℓ preimages
2. Find a collision $h(x^*, c) = h(x^*, c')$
3. For random m_0, compare $\mathrm{MAC}(m_0 \,\|\, [0] \,\|\, c)$ and $\mathrm{MAC}(m_0 \,\|\, [0] \,\|\, c')$
   If they are equal, $x_2 = x^*$

First state-recovery attack

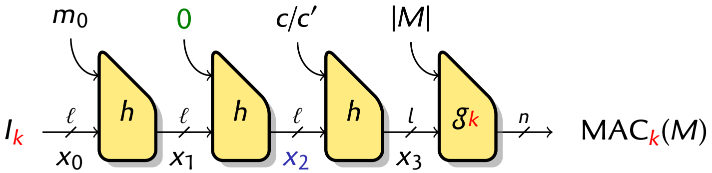
[Naito, Sasaki, Wang & Yasuda '13]



1. Fix a message block $m_1 = [0]$.
   With $2^{\ell-\varepsilon}$ work, find a value x^* with ℓ preimages
2. Find a collision $h(x^*, c) = h(x^*, c')$
3. For random m_0, compare $\mathrm{MAC}(m_0 \| [0] \| c)$ and $\mathrm{MAC}(m_0 \| [0] \| c')$
   If they are equal, $x_2 = x^*$

First state-recovery attack

[Naito, Sasaki, Wang & Yasuda '13]



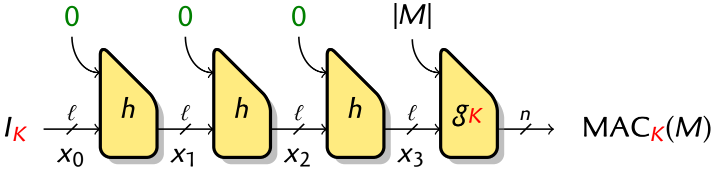
1. Fix a message block $m_1 = [0]$.
   With $2^{\ell-\varepsilon}$ work, find a value x^* with ℓ preimages
2. Find a collision $h(x^*, c) = h(x^*, c')$
3. For random m_0, compare $\mathrm{MAC}(m_0 \| [0] \| c)$ and $\mathrm{MAC}(m_0 \| [0] \| c')$
   If they are equal, $x_2 = x^*$

Structure of state-recovery attacks

1. Identify special states easier to reach
2. Build filter for special states
3. Build messages to reach special states
   Test if special state reached using filters

- In this attack, steps 1 & 2 offline, step 3 online.

Cycle based attack



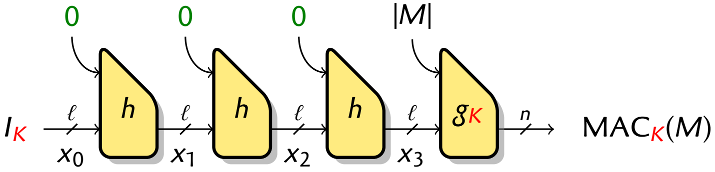
- Using a fixed message block, we iterate a fixed function
- Starting point and ending point unknown because of the key

Can we detect properties of the function $h_0 : x \mapsto h(x, 0)$?

- Study the cycle structure of random mappings
- Used to attack HMAC in related-key setting

[Peyrin, Sasaki & Wang, Asiacrypt 12]

Cycle based attack



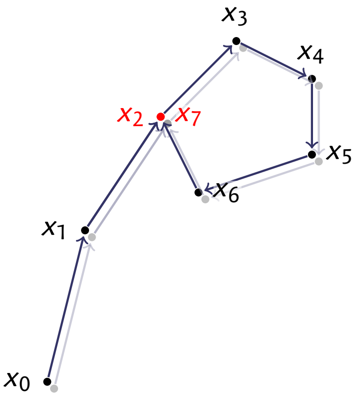
- Using a fixed message block, we iterate a fixed function
- Starting point and ending point unknown because of the key

Can we detect properties of the function $h_0 : x \mapsto h(x, 0)$?

- Study the cycle structure of random mappings
- Used to attack HMAC in related-key setting

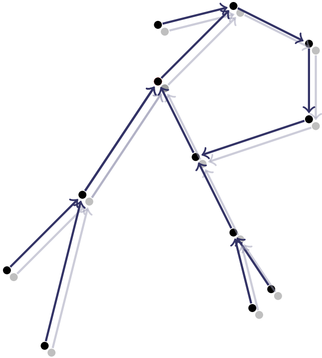
[Peyrin, Sasaki & Wang, Asiacrypt 12]

Random Mappings



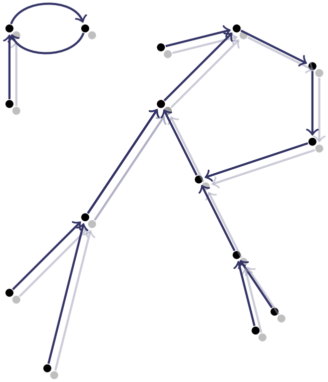
- Functional graph of a random mapping $x \rightarrow f(x)$
- Iterate f: $x_i = f(x_{i-1})$
- Collision after $\approx 2^{\ell/2}$ iterations
  - Cycles
- Trees rooted in the cycle
- Several components

Random Mappings



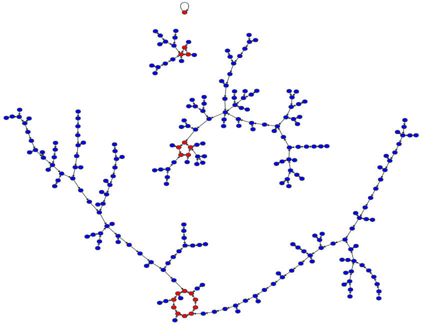
- Functional graph of a random mapping $x \rightarrow f(x)$
- Iterate f: $x_i = f(x_{i-1})$
- Collision after $\approx 2^{\ell/2}$ iterations
  - Cycles
- Trees rooted in the cycle
- Several components

Random Mappings



- Functional graph of a random mapping $x \to f(x)$
- Iterate f: $x_i = f(x_{i-1})$
- Collision after $\approx 2^{\ell/2}$ iterations
  - Cycles
- Trees rooted in the cycle
- Several components

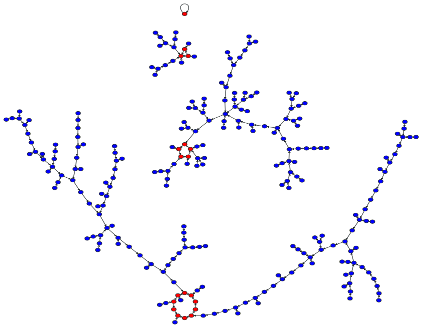
Cycle structure



Expected properties of a random mapping over N points:

- # Components: $\frac{1}{2}\log N$
- # Cyclic nodes: $\sqrt{\pi N/2}$
- Tail length: $\sqrt{\pi N/8}$
- Rho length: $\sqrt{\pi N/2}$
- Largest tree: $0.48N$
- Largest component: $0.76N$

Cycle structure

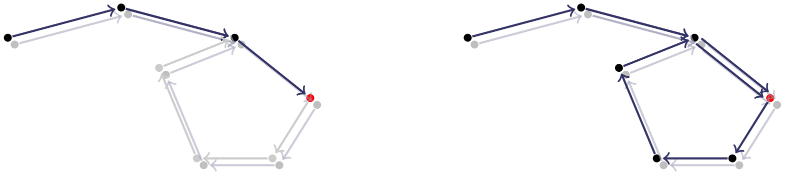


Expected properties of a random mapping over N points:

- # Components: $\frac{1}{2}\log N$
- # Cyclic nodes: $\sqrt{\pi N/2}$
- Tail length: $\sqrt{\pi N/8}$
- Rho length: $\sqrt{\pi N/2}$
- Largest tree: $0.48N$
- Largest component: $0.76N$

Using the cycle length

1. Offline: find the cycle length L of the main component of h_0
2. Online: query $t = \mathrm{MAC}(r \parallel [0]^{2^{\ell/2}})$ and $t' = \mathrm{MAC}(r \parallel [0]^{2^{\ell/2}+L})$

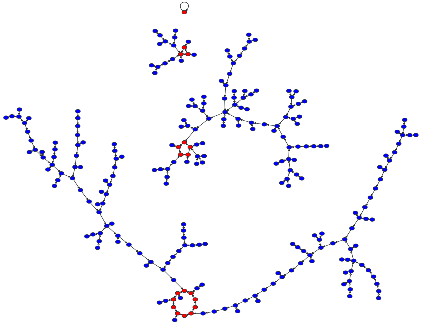


Success if

- The starting point is in the main component $p = 0.76$
- The cycle is reached with less than $2^{\ell/2}$ iterations $p \geq 0.5$

Randomize starting point

Cycle structure

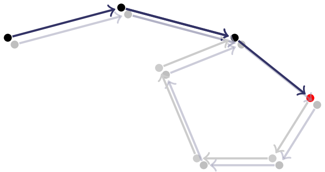


Expected properties of a random mapping over N points:

- # Components: $\frac{1}{2}\log N$
- # Cyclic nodes: $\sqrt{\pi N/2}$
- Tail length: $\sqrt{\pi N/8}$
- Rho length: $\sqrt{\pi N/2}$
- Largest tree: $0.48N$
- Largest component: $0.76N$

Using the cycle length

1. Offline: find the cycle length L of the main component of h_0
2. Online: query $t = \mathrm{MAC}(r \,\|\, [0]^{2^{\ell/2}})$ and $t' = \mathrm{MAC}(r \,\|\, [0]^{2^{\ell/2}+L})$



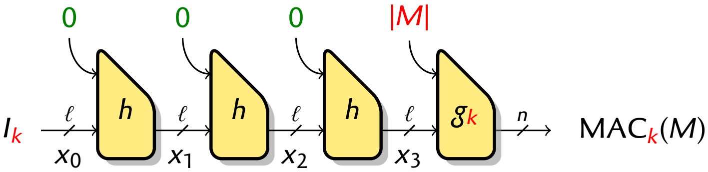
Success if

- The starting point is in the main component $p = 0.76$
- The cycle is reached with less than $2^{\ell/2}$ iterations $p \geq 0.5$

Randomize starting point

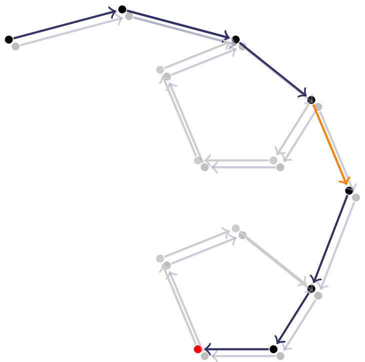
Dealing with the message length

Problem: most MACs use the message length.



Dealing with the message length

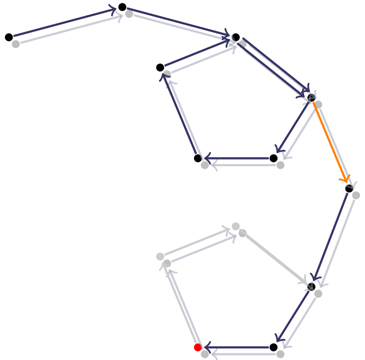
Solution: reach the cycle twice



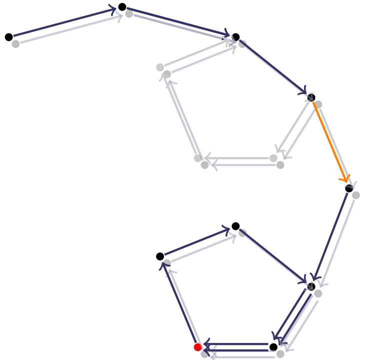
$$M = r \,\|\, [0]^{2^{\ell/2}} \,\|\, [1] \,\|\, [0]^{2^{\ell/2}}$$

Dealing with the message length

Solution: reach the cycle twice



$M_1 = r \,\|\, [0]^{2^{\ell/2}+L} \,\|\, [1] \,\|\, [0]^{2^{\ell/2}}$



$M_2 = r \,\|\, [0]^{2^{\ell/2}} \,\|\, [1] \,\|\, [0]^{2^{\ell/2}+L}$

Distinguishing-H attack

1. Offline: find the cycle length L of the main component of h_0
2. Online: query

$$t = \mathrm{MAC}(r \parallel [0]^{2^{\ell/2}} \parallel [1] \parallel [0]^{2^{\ell/2}+L})$$
$$t' = \mathrm{MAC}(r \parallel [0]^{2^{\ell/2}+L} \parallel [1] \parallel [0]^{2^{\ell/2}})$$

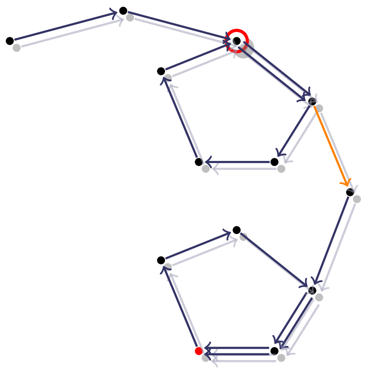
3. If $t = t'$, then h is the compression function in the oracle

Analysis

- Complexity: $2^{\ell/2}$ compression function calls
- Success probability: $p \simeq 0.14$
  - Both starting point are in the main component $p = 0.76^2$
  - Both cycles are reached with less than $2^{\ell/2}$ iterations $p \geq 0.5^2$

State recovery attack

- Consider the first cyclic point
- With high pr., root of the giant tree

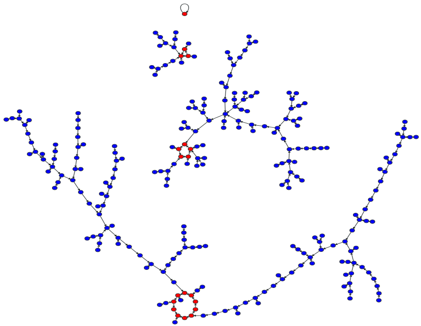


1. Offline: find cycle length L, and root of giant tree α
2. Online: Binary search for smallest z with collisions: $\mathrm{MAC}(r \,\|\, [0]^{z} \,\|\, [x] \,\|\, [0]^{2^{\ell/2}+L})$, $\mathrm{MAC}(r \,\|\, [0]^{z+L} \,\|\, [x] \,\|\, [0]^{2^{\ell/2}})$
3. State after $r \,\|\, [0]^z$ is α (with high pr.)

Analysis

- Complexity $2^{\ell/2} \times \ell \times \log(\ell)$

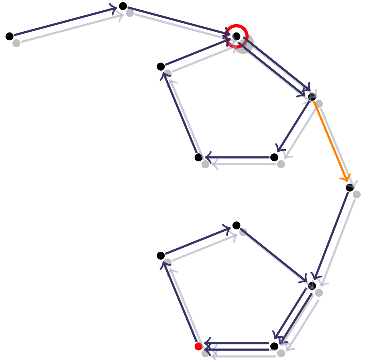
Cycle structure



Expected properties of a random mapping over N points:

- # Components: $\frac{1}{2}\log N$
- # Cyclic nodes: $\sqrt{\pi N/2}$
- Tail length: $\sqrt{\pi N/8}$
- Rho length: $\sqrt{\pi N/2}$
- Largest tree: $0.48N$
- Largest component: $0.76N$

State recovery attack



- Consider the first cyclic point
- With high pr., root of the giant tree

1. Offline: find cycle length L, and root of giant tree α
2. Online: Binary search for smallest z with collisions:
   $\mathrm{MAC}(r \parallel [0]^{z} \parallel [x] \parallel [0]^{2^{\ell/2}+L})$,
   $\mathrm{MAC}(r \parallel [0]^{z+L} \parallel [x] \parallel [0]^{2^{\ell/2}})$
3. State after $r \parallel [0]^z$ is α (with high pr.)

Analysis

- Complexity $2^{\ell/2} \times \ell \times \log(\ell)$

Short message attacks

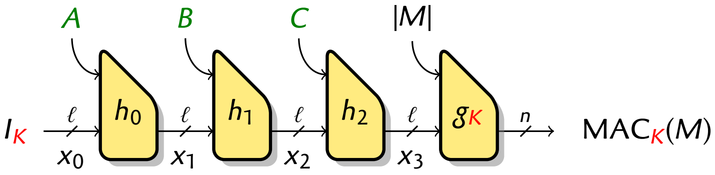
Limitations of cycle-based attacks

- Messages of length $2^{\ell/2}$ are not very practical...
  - SHA-1 and HAVAL limit the message length to 2^{64} bits
- Cycle detection impossible with messages shorter than $L \approx 2^{\ell/2}$
  - Shorter cycles have a small component
- Not applicable to HAIFA hash functions

Compare with collision finding algorithms

- Pollard's rho algorithm use cycle detection
- Parallel collision search for van Oorschot and Wiener uses shorter chains

Chain-based attack



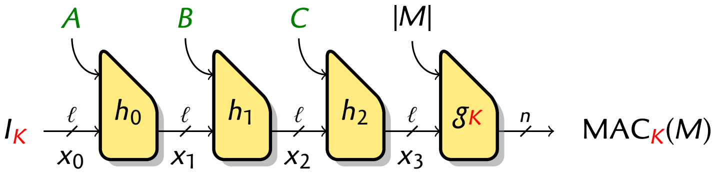
- Using a fixed message, we iterate a fixed sequence of function
- Starting point and ending point unknown because of the key

Can we detect properties of the iteration of fixed functions?

- Study the entropy loss

skip details

Chain-based attack



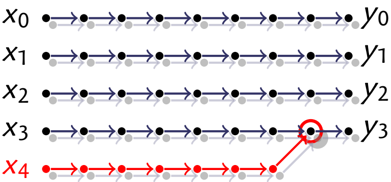
- Using a fixed message, we iterate a fixed sequence of function
- Starting point and ending point unknown because of the key

Can we detect properties of the iteration of fixed functions?

- Study the entropy loss

skip details

Collision finding with short chains



1. Compute chains $x \rightsquigarrow y$
   Stop when y distinguished
2. If $y \in \{y_i\}$, collision found

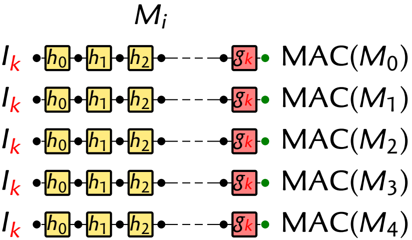
Theorem (Entropy loss)

Let $f_1, f_2, \dots, f_{2^s}$ be a fixed sequence of random functions; the image of $g_{2^s} \triangleq f_{2^s} \circ \dots \circ f_2 \circ f_1$ contains about $2^{\ell-s}$ points.

- Use these state as special states (instead of cycle entry point)

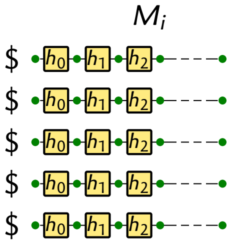
State-recovery attacks

- Send messages to the oracle



Online Structure

- Do some computations offline with the compression function

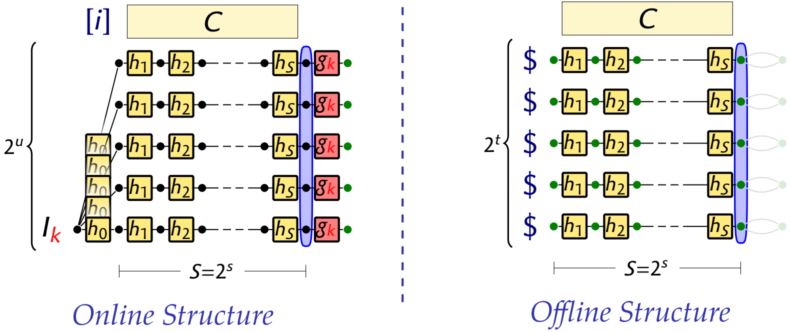


Offline Structure

- Match the sets of points?
  - How to test equality? Online chaining values unknown
  - How many equality test do we need?

First attempt

- Chains of length 2^s, with a fixed message C



1. Evaluate 2^t chains offline
   Build filters for endpoints
2. Query 2^u message $M_i = [i] \| C$
   Test endpoints with filters

$s + t + u = \ell$

Cplx: 2^{s+t+u}

Building filters

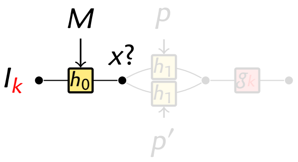
Filters to compare online and online states

Test whether the state reached after processing M is equal to x

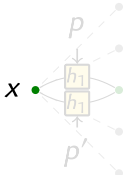
- Collisions are preserved by the finalization (for same-length messages)

2 $\mathrm{MAC}(M\|p) \stackrel{?}{=} \mathrm{MAC}(M\|p')$

1 Find a collision: $h(x,p) = h(x,p')$



Online Structure



Offline Structure

Building filters

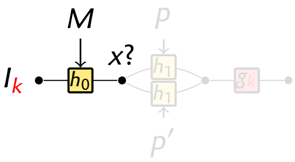
Filters to compare online and online states

Test whether the state reached after processing M is equal to x

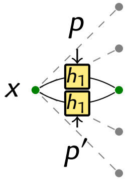
- Collisions are preserved by the finalization (for same-length messages)

2 $\mathrm{MAC}(M\|p) \stackrel{?}{=} \mathrm{MAC}(M\|p')$

1 Find a collision: $h(x,p) = h(x,p')$



Online Structure



Offline Structure

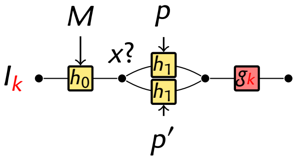
Building filters

Filters to compare online and online states

Test whether the state reached after processing M is equal to x

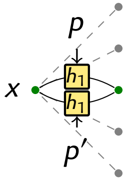
- Collisions are preserved by the finalization (for same-length messages)

2 $\mathrm{MAC}(M\|p) \stackrel{?}{=} \mathrm{MAC}(M\|p')$



Online Structure

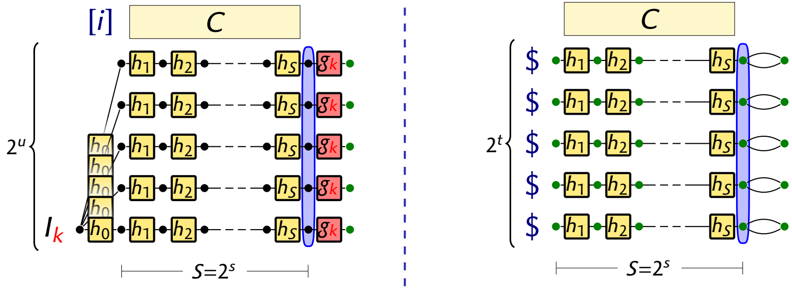
1 Find a collision: $h(x,p) = h(x,p')$



Offline Structure

First attempt

- Chains of length 2^s, with a fixed message C



Online Structure *Offline Structure*

1. Evaluate 2^t chains offline
   Build filters for endpoints
2. Query 2^u message $M_i = [i] \parallel C$
   Test endpoints with filters

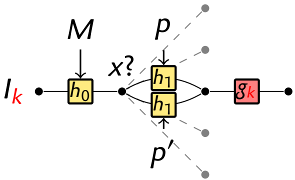
$s + t + u = \ell$

Cplx: 2^{s+t+u}

Online filters

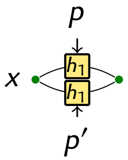
- Using the filters is too expensive.
- If we build filters online, using them is cheap.

1 Find p, p' s.t.
$\mathrm{MAC}(M \| p) = \mathrm{MAC}(M \| p')$



Online Structure

2 $h(x, m) \stackrel{?}{=} h(x, m')$

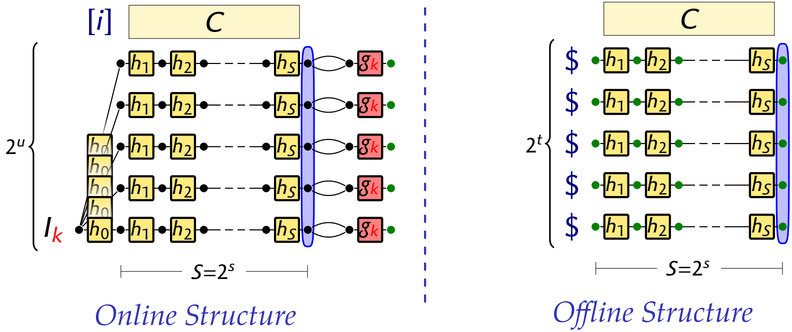


Offline Structure

Cost	Build	Test
Offline filter	$2^{\ell/2}$	2^s
Online filter	$2^{\ell/2+s}$	1

First attack on HMAC-HAIFA

- Chains of length 2^s, with a fixed message C

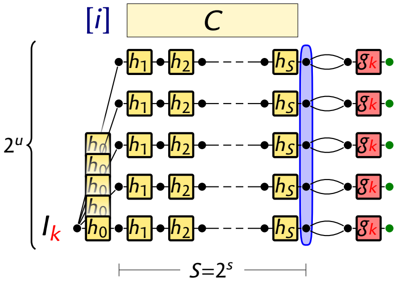


$s + t + u = \ell$

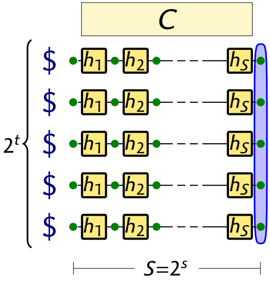
1. Query 2^u message $M_i = [i] \parallel C$
   Build filters for M_i — Cplx: $2^{s+u+\ell/2}$
2. Evaluate 2^t chains offline — Cplx: 2^{t+s}
   Test endpoints with filters — Cplx: 2^{t+u}

First attack on HMAC-HAIFA

- Chains of length 2^s, with a fixed message C



Online Structure



Offline Structure

1. Query 2^u message $M_i = [i] \parallel C$
   Build filters for M_i
2. Evaluate 2^t chains offline
   Test endpoints with filters

Optimal complexity

$2^{\ell - s}$, for $s \leq \ell/6$
(using $u = s$)
Minimum: $2^{5\ell/6}$

Diamond filters

- Building filers is a bottleneck.
- Can we amortize the cost of building many filters?

Diamond structure [Kelsey & Kohno, EC'06]

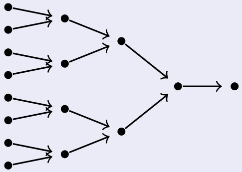
Herd N initial states to a common state

- Try $\approx 2^{\ell/2}/\sqrt{N}$ msg from each state.
- Whp, the initial states can be paired
- Repeat... Total $\approx \sqrt{N} \cdot 2^{\ell/2}$

Diamond filters

- Building filers is a bottleneck.
- Can we amortize the cost of building many filters?

Diamond structure *[Kelsey & Kohno, EC'06]*



Herd N initial states to a common state

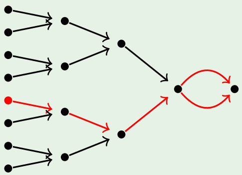
- Try $\approx 2^{\ell/2}/\sqrt{N}$ msg from each state.
- Whp, the initial states can be paired
- Repeat...

Total $\approx \sqrt{N} \cdot 2^{\ell/2}$

Diamond filters

- Building filers is a bottleneck.
- Can we amortize the cost of building many filters?

Diamond filter



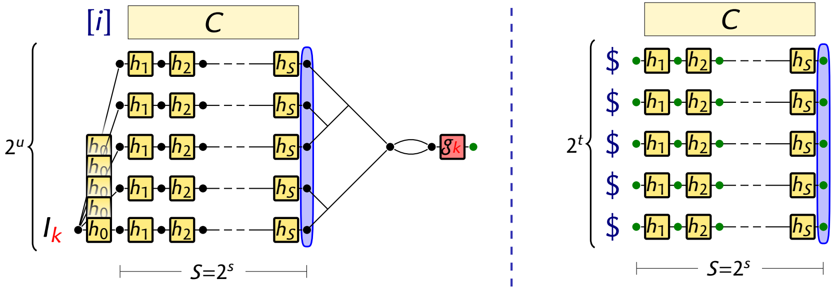
1. Build a diamond structure
2. Build a collision filter for the final state

- Can also be built online

- Building N offline filters: $\sqrt{N} \cdot 2^{\ell/2}$ rather than $N \cdot 2^{\ell/2}$
- Building N online filters: $\sqrt{N} \cdot 2^{\ell/2+s}$ rather than $N \cdot 2^{\ell/2+s}$

Improved attack on HMAC-HAIFA

- Chains of length 2^s, with a fixed message C



Online Structure

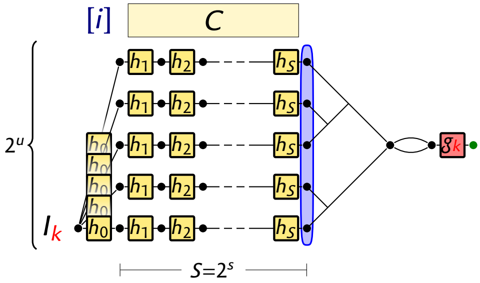
Offline Structure

$s + t + u = \ell$

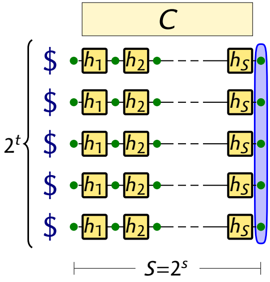
1. Query 2^u message $M_i = [i] \,\|\, C$
   Build diamond filter for M_i — Cplx: $2^{s+u/2+\ell/2}$
2. Evaluate 2^t chains offline — Cplx: 2^{t+s}
   Test endpoints with filters — Cplx: 2^{t+u}

Improved attack on HMAC-HAIFA

- Chains of length 2^s, with a fixed message C



Online Structure



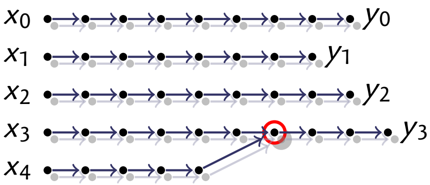
Offline Structure

1. Query 2^u message $M_i = [i] \parallel C$
   Build diamond filter for M_i
2. Evaluate 2^t chains offline
   Test endpoints with filters

Optimal complexity

$2^{\ell - s}$, for $s \leq \ell/5$
(using $u = s$)
Minimum: $2^{4\ell/5}$

Improvement using collisions (fixed function)



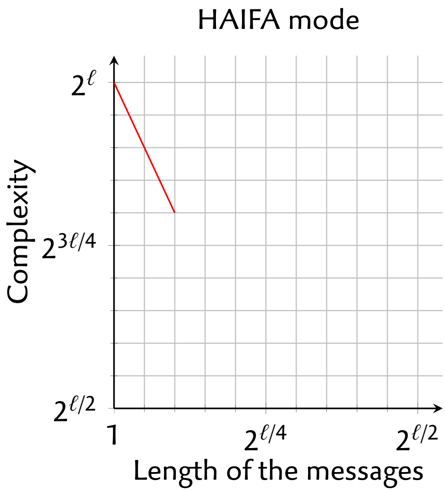
1. Compute chains $x \rightsquigarrow y$
   Stop when y distinguished
2. If $y \in \{y_i\}$, collision found

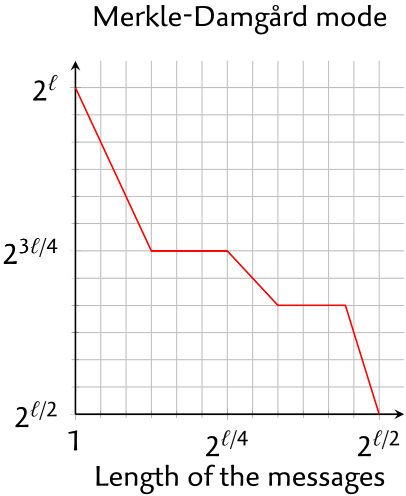
Theorem (Entropy loss for collisions)

Let $\hat{x}$ and $\hat{y}$ be two collisions found using chains of length 2^s, with a fixed ℓ-bit random function f. Then $\Pr\left[\hat{x} = \hat{y}\right] = \Theta(2^{2s-\ell})$.

- Use the collisions as special states (instead of cycle entry point)

Trade-offs for state-recovery attacks





Outline

Bibliography

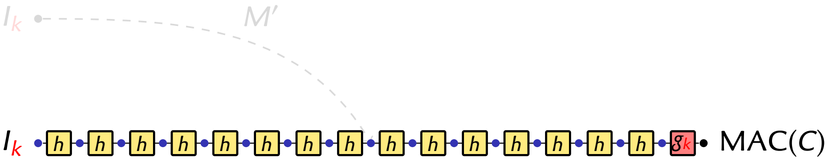
- T. Peyrin, L. Wang
  Generic Universal Forgery Attack on Iterative Hash-Based MACs
  EUROCRYPT 2014
- J. Guo, T. Peyrin, Y. Sasaki, L. Wang
  Updates on Generic Attacks against HMAC and NMAC
  CRYPTO 2014
- I. Dinur, G. Leurent
  Improved Generic Attacks against Hash-Based MACs and HAIFA
  CRYPTO 2014

Universal forgery attack

- Given a challenge message C, compute MAC(C)
  - $\text{len}(C) = 2^s$
  - Oracle access to the MAC, can't ask MAC(C)

- Study internal states for the computation of MAC(C)
  - Unknown because of initial key and final key

1. Build a different message reaching same states
2. Query MAC(M'), use as forgery

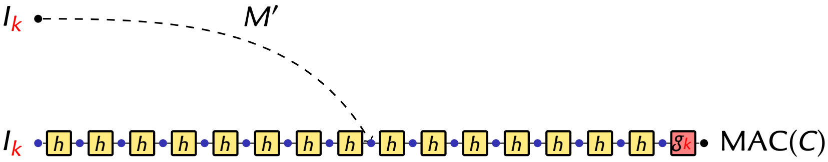


Universal forgery attack

- Given a challenge message C, compute MAC(C)
  - $\text{len}(C) = 2^s$
  - Oracle access to the MAC, can't ask MAC(C)

- Study internal states for the computation of MAC(C)
  - Unknown because of initial key and final key

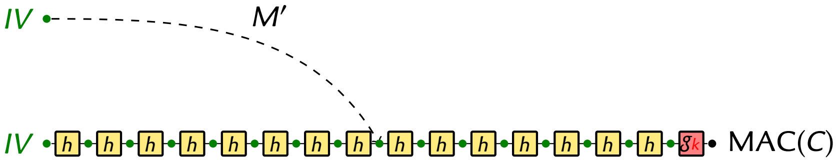
1. Build a different message reaching same states
2. Query MAC(M'), use as forgery



UF against secret-suffix MAC

- Secret-suffix has no key at the beginning
  - All internal states for challenge message are known!
- Long-message second-preimage attack [Kelsey & Schneier '05]
  - $H(M) = H(C) \Longrightarrow MAC(M) = H(M \,\|\, k) = H(C \,\|\, k) = MAC(C)$

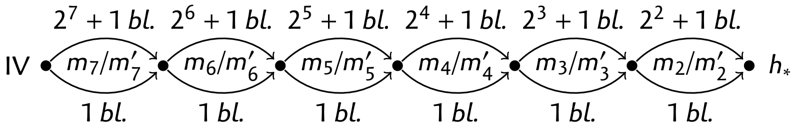
1. Build a expandable message — Cplx: $2^{\ell/2}$
2. Find a connexion from the IV to the target states — Cplx: $2^{\ell-s}$
3. Select expandable message

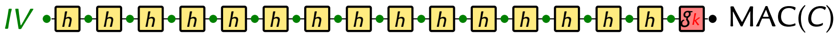


UF against secret-suffix MAC

- Secret-suffix has no key at the beginning
  - All internal states for challenge message are known!
- Long-message second-preimage attack [Kelsey & Schneier '05]
  - $H(M) = H(C) \implies MAC(M) = H(M \,\|\, k) = H(C \,\|\, k) = MAC(C)$

1 Build a expandable message Cplx: $2^{\ell/2}$

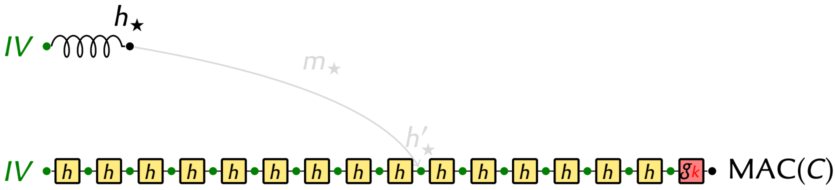




UF against secret-suffix MAC

- Secret-suffix has no key at the beginning
  - All internal states for challenge message are known!
- Long-message second-preimage attack [Kelsey & Schneier '05]
  - $H(M) = H(C) \implies MAC(M) = H(M \,\|\, k) = H(C \,\|\, k) = MAC(C)$

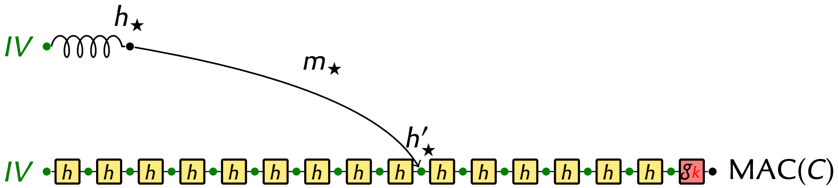
1. Build a expandable message Cplx: $2^{\ell/2}$
2. Find a connexion from $x_\star$ to the target states Cplx: $2^{\ell-s}$
3. Select expandable message



UF against secret-suffix MAC

- Secret-suffix has no key at the beginning
  - All internal states for challenge message are known!
- Long-message second-preimage attack [Kelsey & Schneier '05]
  - $H(M) = H(C) \implies MAC(M) = H(M \| k) = H(C \| k) = MAC(C)$

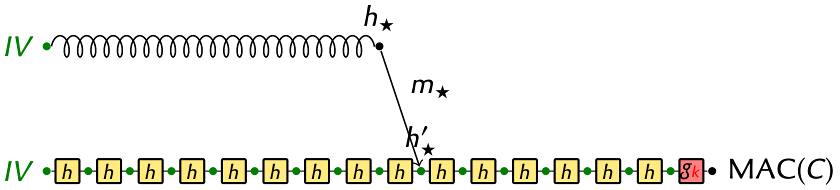
1. Build a expandable message Cplx: $2^{\ell/2}$
2. Find a connexion from $x_\star$ to the target states Cplx: $2^{\ell-s}$
3. Select expandable message



UF against secret-suffix MAC

- Secret-suffix has no key at the beginning
  - All internal states for challenge message are known!
- Long-message second-preimage attack [Kelsey & Schneier '05]
  - $H(M) = H(C) \implies MAC(M) = H(M \,\|\, k) = H(C \,\|\, k) = MAC(C)$

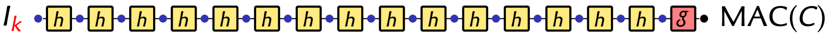
1. Build a expandable message — Cplx: $2^{\ell/2}$
2. Find a connexion from $x_\star$ to the target states — Cplx: $2^{\ell-s}$
3. Select expandable message



UF against secret-prefix MAC

- Secret-suffix has no key at the end
  - Finalization function is known!

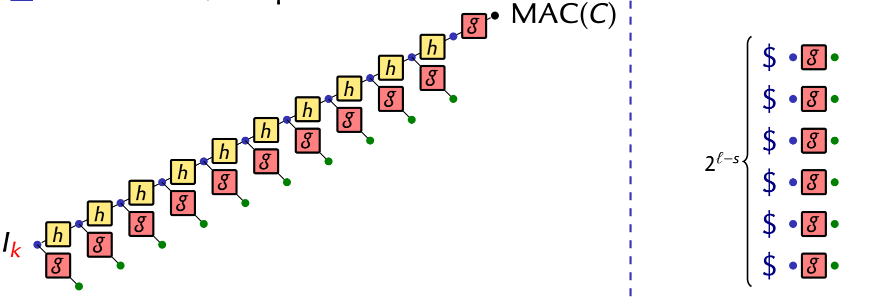
1. Query the MAC of $C_{|i}$ (truncated to i blocks) Cplx: $2^{2\cdot s}$
2. Evaluate the finalization function on $2^{\ell-s}$ states Cplx: $2^{\ell-s}$
3. Find a match, compute MAC



UF against secret-prefix MAC

- Secret-suffix has no key at the end
  - Finalization function is known!

1. Query the MAC of $C_{|i}$ (truncated to i blocks) — Cplx: $2^{2 \cdot s}$
2. Evaluate the finalization function on $2^{\ell - s}$ states — Cplx: $2^{\ell - s}$
3. Find a match, compute MAC



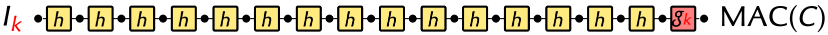
Online Structure *Offline Structure*

UF attack against hash-based MAC

- Combine both techniques
  1. Recover an internal state of the challenge
  2. Use second-preimage attack with known state

- Hard part is to recover an internal state

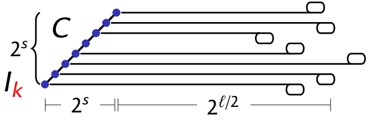
- Extract information about the challenge state through g_k
  - Compute distance to cycle
  - Use entropy loss of iterations



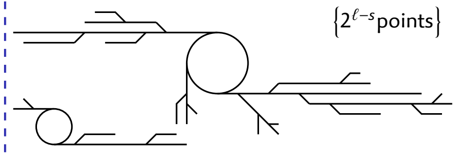
Using cycles

Main idea

- Measure the distance from challenge point to cycle in $h_{[0]}$
  - Add zero blocks after the challenge
- Match with offline points with known distance



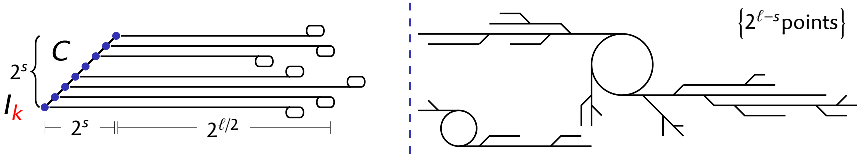
Online Structure



Offline Structure

Using cycles

1. (online) For each challenge state, use binary search to find distance
   $\mathrm{MAC}(C_{|i} \parallel 0^{d+L} \parallel 1 \parallel 0^{2^{\ell/2}}) \stackrel{?}{=} \mathrm{MAC}(C_{|i} \parallel 0^{d} \parallel 1 \parallel 0^{2^{\ell/2+L}})$
2. (offline) Build a structure with $2^{\ell-s}$ points with known distance.
3. (offline) Match the challenge states and the offline structure
4. (online) Test candidates at the right distance.

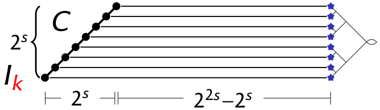


Online Structure *Offline Structure*

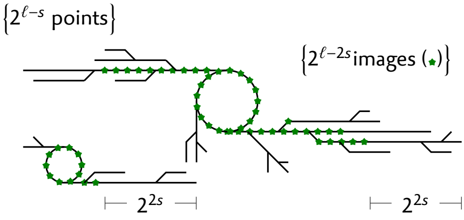
Using chains

Main idea

- Add a sequence of fixed message blocks to reduce image space
- Match in the reduced space



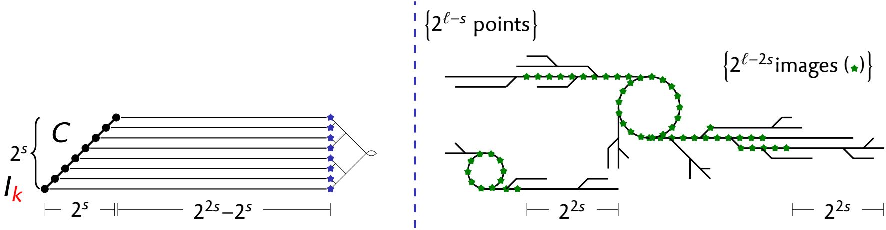
Online Structure



Offline Structure

Using chains

1. (online) Query messages $M_i = C_{|i} \parallel [0]^{2^{2s}-i}$.
   Build diamond filter for endpoints Y
2. (offline) Build a structure with $2^{\ell-s}$ points.
   Consider 2^{2s}-images X. $|X| \leq 2^{\ell-2s}$
3. (offline) Match X and Y.
4. (offline) For each match, find preimages as candidates.



Online Structure

Offline Structure

Universal forgery attacks: summary

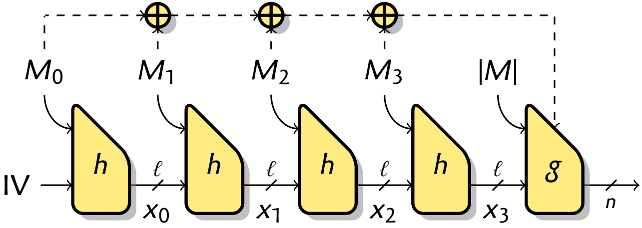
Universal forgery attacks

- It is possible to perform a generic universal forgery attack
- Best attack so far: $2^{\ell-s}$, with $s \leq \ell/4$ ($2^{3\ell/4}$ with $s = \ell/4$)

- Using distance to the cycle: query length $2^{\ell/2}$
  - Complexity $2^{\ell-s}, s \leq \ell/6$ [Peyrin & Wang, EC '14]
    Optimal: $2^{5\ell/6}$, with $s = 2^{\ell/6}$
  - Complexity $2^{\ell-s}, s \leq \ell/4$ [Guo, Peyrin, Sasaki & Wang, CR '14]
    Optimal: $2^{3\ell/4}$, with $s = 2^{\ell/4}$
- Later attack using chains: shorter query length 2^t
  - Complexity $2^{\ell-s}$ $, s \leq$ $\ell/7, t = 2s$ [Dinur & L, CR '14]
    Optimal: $2^{6\ell/7}$, with $s = 2^{\ell/7}, t = 2\ell/7$
  - Complexity $2^{\ell-s/2}, s \leq 2\ell/5, t =$ s [Dinur & L, CR '14]
    Optimal: $2^{4\ell/5}$, with $s = 2^{2\ell/5}, t = 2\ell/5$

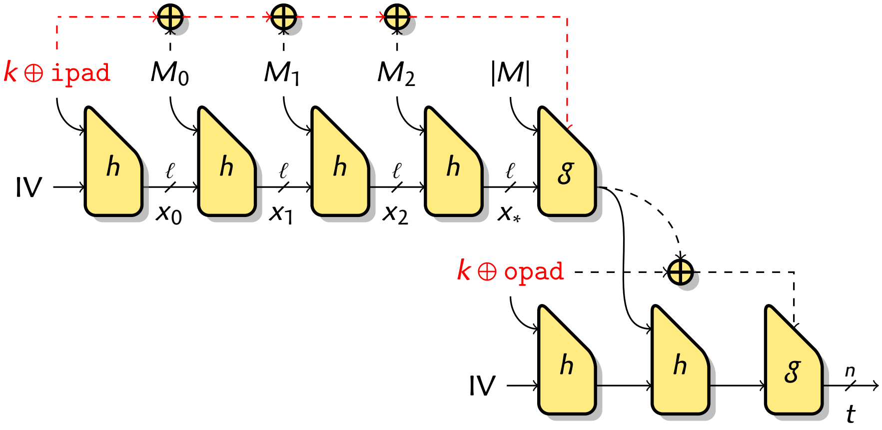
Outline

GOST hash functions



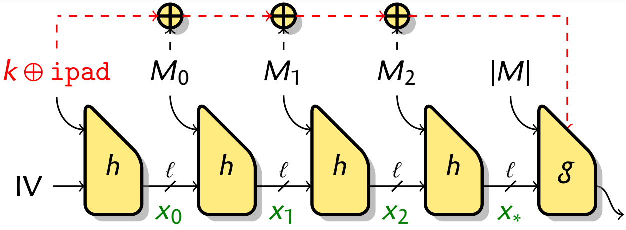
- Family of Russian standards
  - GOST-1994: $n = \ell = 256$
  - GOST-2012: $n \leq \ell = 512$, HAIFA mode (aka Streebog)
- GOST and HMAC-GOST standardized by IETF
- Checksum (dashed lines)
  - Larger state should increase the security

HMAC-GOST



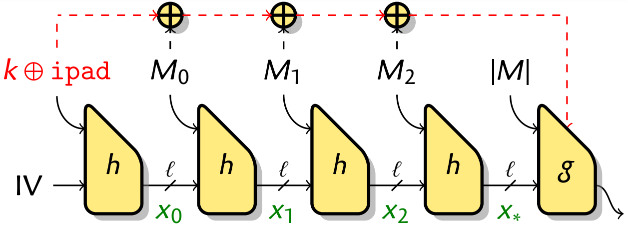
- In HMAC, key-dependant value used after the message
  - Related-key attacks on the last block

Key recovery attack on HMAC-GOST



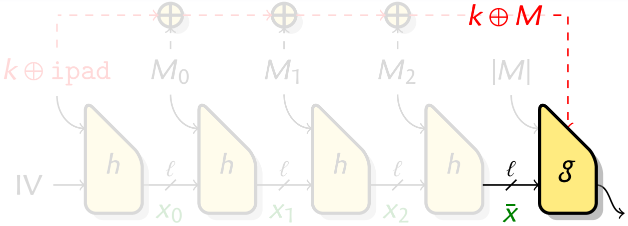
1. Recover the state of a short message
2. Build a multicollision: $2^{3l/4}$ messages with the same x_*
3. Query messages, detect collisions $g(\bar{x}, k \oplus M) = g(\bar{x}, k \oplus M')$
   Store $(M \oplus M', M)$ for $2^{\ell/2}$ collisions
4. Find collisions $g(\bar{x}, y) = g(\bar{x}, y')$ offline
   Store $(x \oplus y', y)$ for $2^{\ell/2}$ collisions
5. Detect match $M \oplus M' = y \oplus y'$. With high probability $M \oplus k = y$

Key recovery attack on HMAC-GOST



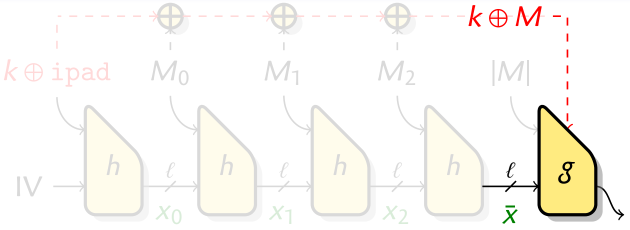
1. Recover the state of a short message
2. Build a multicollision: $2^{3l/4}$ messages with the same x_*
3. Query messages, detect collisions $g(\bar{x}, k \oplus M) = g(\bar{x}, k \oplus M')$
   Store $(M \oplus M', M)$ for $2^{\ell/2}$ collisions
4. Find collisions $g(\bar{x}, y) = g(\bar{x}, y')$ offline
   Store $(x \oplus y', y)$ for $2^{\ell/2}$ collisions
5. Detect match $M \oplus M' = y \oplus y'$. With high probability $M \oplus k = y$

Key recovery attack on HMAC-GOST



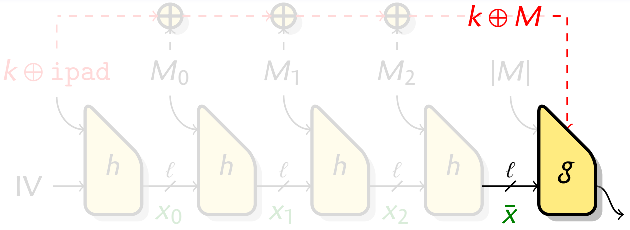
1. Recover the state of a short message
2. Build a multicollision: $2^{3l/4}$ messages with the same x_*
3. Query messages, detect collisions $g(\bar{x}, k \oplus M) = g(\bar{x}, k \oplus M')$
   Store $(M \oplus M', M)$ for $2^{\ell/2}$ collisions
4. Find collisions $g(\bar{x}, y) = g(\bar{x}, y')$ offline
   Store $(x \oplus y', y)$ for $2^{\ell/2}$ collisions
5. Detect match $M \oplus M' = y \oplus y'$. With high probability $M \oplus k = y$

Key recovery attack on HMAC-GOST



1. Recover the state of a short message
2. Build a multicollision: $2^{3l/4}$ messages with the same x_*
3. Query messages, detect collisions $g(\bar{x}, k \oplus M) = g(\bar{x}, k \oplus M')$
   Store $(M \oplus M', M)$ for $2^{\ell/2}$ collisions
4. Find collisions $g(\bar{x}, y) = g(\bar{x}, y')$ offline
   Store $(x \oplus y', y)$ for $2^{\ell/2}$ collisions
5. Detect match $M \oplus M' = y \oplus y'$. With high probability $M \oplus k = y$

Key recovery attack on HMAC-GOST



1. Recover the state of a short message
2. Build a multicollision: $2^{3l/4}$ messages with the same x_*
3. Query messages, detect collisions $g(\bar{x}, k \oplus M) = g(\bar{x}, k \oplus M')$
   Store $(M \oplus M', M)$ for $2^{\ell/2}$ collisions
4. Find collisions $g(\bar{x}, y) = g(\bar{x}, y')$ offline
   Store $(x \oplus y', y)$ for $2^{\ell/2}$ collisions
5. Detect match $M \oplus M' = y \oplus y'$. With high probability $M \oplus k = y$

Complexity

Surprising result

The checksum actually make the hash function weaker!

- HMAC-GOST-1994 is weaker than HMAC-SHA256
- HMAC-GOST-2012 is weaker than HMAC-SHA512

It is important to recover the state of a short message

- For GOST-1994, we can recover the state of a short message from a longer one using padding tricks Total complexity $2^{3\ell/4}$
- For GOST-2012, we use an advanced attack with message length $2^{\ell/10}$ Total complexity $2^{4\ell/5}$

Attack complexity

Function	Mode	ℓ	s	St. rec.	Univ. F	K. rec.
SHA-1	MD	160	2^{55}	2^{107}	2^{132}	
SHA-224	MD	256	2^{55}	2^{192}		
SHA-256	MD	256	2^{55}	2^{192}	2^{228}	
SHA-512	MD	512	2^{118}	2^{384}	2^{453}	
HAVAL	MD	256	2^{54}	2^{192}	2^{229}	
WHIRLPOOL	MD	512	2^{247}	2^{283}	2^{446}	
BLAKE-256	HAIFA	256	2^{55}	2^{213}		
BLAKE-512	HAIFA	512	2^{118}	2^{419}		
Skein-512	HAIFA	512	2^{90}	2^{419}		
GOST-94	MD+σ	256	∞	2^{128}	2^{192}	2^{192}
Streebog	HAIFA+σ	512	∞	2^{419}	2^{419}	2^{419}

Conclusion

Be carefull with security proof

- "CBC-MAC is proven secure" does not mean "CBC-MAC-AES is a secure as AES"
  - Most security proofs are up to the birthday bound
  - Is 64-bit security enough?
- Don't assume too much after the security bound of the proof
  - Generic key-recovery for envelope-MAC, AEZ, HMAC-GOST

Gaps between proofs and attacks!

- Better generic attacks?
- Better proofs?

Thanks

Questions?