# Symmetric Cryptanalysis Beyond Primitives

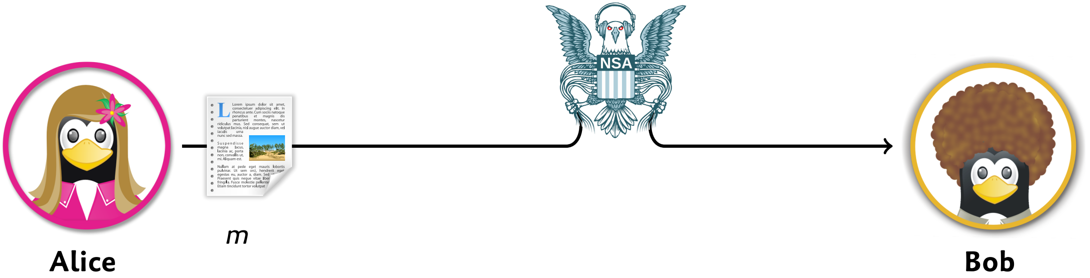
Gaëtan Leurent

Inria, France

HDR defense

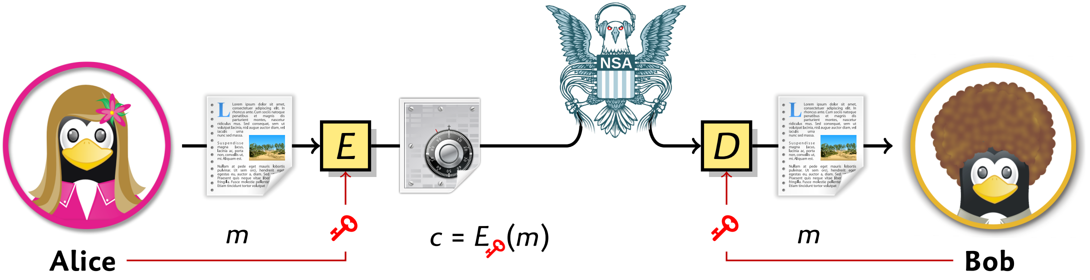
# Cryptography

- Cryptography aims to secure communication against an adversary
  - Confidentiality: keep the message secret
  - Authenticity: prove who sends the message



# Cryptography

- Cryptography aims to secure communication against an adversary
  - Confidentiality: keep the message secret
  - Authenticity: prove who sends the message



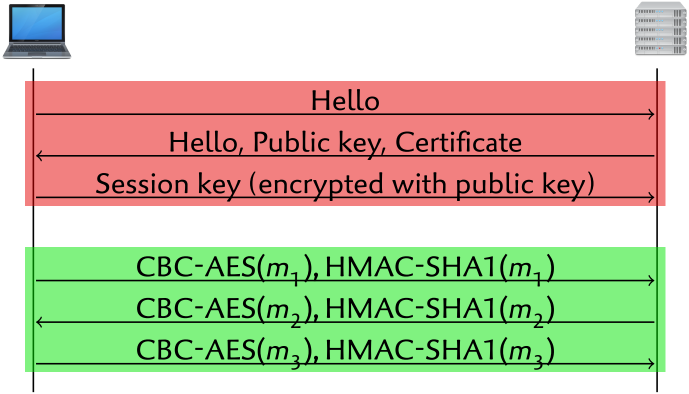
# Cryptography

- Cryptography aims to secure communication against an adversary
  - Confidentiality: keep the message secret
  - Authenticity: prove who sends the message
- More generally: mathematical tools to secure data in the presence of an adversary
  - Access control
  - Electronic voting
  - Digital certificates (eg COVID)
  - Lottery
- Used everyday



# Practical example: TLS (Secure channel)

- Widespread deployment of cryptography
  - 80% of webpages encrypted with TLS (HTTP → HTTPS)



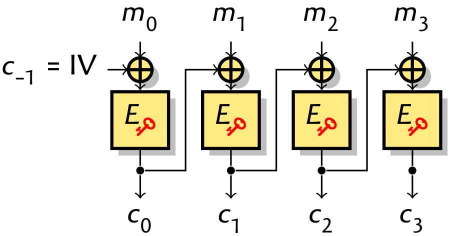

- Handshake protocol
  - Establish session key using public key crypto
- Record protocol
  - Exchange application data using secret key crypto

- I study symmetric cryptography
  - Alice and Bob share secret key used to encrypt and decrypt

# Modes and primitives

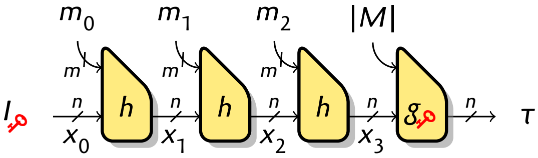
- Primitive with fixed-size inputs, and mode of operation

- Encryption example: CBC-AES




- Mode: CBC
  - Encryption mode
- Primitive: AES block cipher
  - $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$
  - $E$🔑 $: x \mapsto E($🔑$, x)$ permutation

- Authentication example: HMAC-SHA1




- Mode: HMAC
  - Message Authentication Code (MAC)
- Primitive: SHA-1 compression function
  - $h : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$
  - Public function without structural property

# Security of cryptographic protocols

**Kerckhoffs principles** [Kerckhoffs, 1883]

1. Le système doit être matériellement, sinon mathématiquement, indéchiffrable;
2. Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi;
3. La clef doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants;
4. ...

- Two ways to approach security
  - Prove security based on mathematical assumption
  - Cryptanalysis: a system is secure if people tried to attack it and failed

*Anybody can design a system that he himself cannot break* [Schneier]

# Cryptanalysis

## *Classical approach*

- Security of the protocol (TLS, SSH, ...)
  - Security proofs, assuming security of cryptographic operations
- Security of the modes (HMAC, CBC, ...)
  - Security proofs, assuming security of the primitive
- Security of the primitives (AES, SHA-1, RSA, ...)
  - Studied with cryptanalysis

- We need public cryptanalysis research
  - Evaluation by the community
  - Only way to evaluate primitives
- Goal: replace weak algorithms before attacks are practical
  - We know that some government agencies attack weak cryptography

# Cryptanalysis beyond primitives

- Cryptanalysis usually applied inside primitives
- If this talk: cryptanalysis techniques outside the primitive

1. Generic attacks
   - Target the mode itself without using properties of the primitive
   - Nice algorithmic problems & mathematical properties
2. Real-world impact of cryptanalysis
   - Extend cryptanalysis of primitives to break high-level construction
   - Demonstrate attacks in practice to convince users they are real

## High-level goal

- Better understanding of the security by considering both proofs and cryptanalysis
  - Security proofs give lower bound on the security
  - Attacks give upper bound on the security

# Overview of my results (I)

## Generic attacks

- Hash-based MACs
- Combiner preimage $H_1(M) \oplus H_2(M)$
- Beyond-birthday-bound MACs
- CTR plaintext recovery
- Quantum forgery against MACs

## Real-world impact of crytanalysis

- Transcript-collision attacks [Sloth]
- Practical CBC collisions (64-bit BC) [Sweet32]
- Practical SHA-1 chosen-prefix collision [Shambles]

# Overview of my results (II)

## Design of primitives

- SPRING (lattice-based PRF)
- LS-Designs (block ciphers)
- SCREAM (CAESAR candidate)
- Spook (NIST LW candidate)
- Saturnin (NIST LW candidate)
- Lightweight MDS

## Cryptanalysis of primitives

- ARXtools
- Chaskey (differential-linear)
- Quantum differential/linear cryptanalysis
- SHA-1 chosen-prefix collisions
- AES key-schedule new representation
- Gimli distinguishers (full permutation)
- Simon/Simeck (differential and linear)
- GEA (GEA-1 intentional weakness)
- Algebraic attacks against AOC

# Outline

Generic Attacks Against Encryption Modes

Generic Attacks Against MACs in the Quantum Setting

Generic Attacks Against Hash Combiners

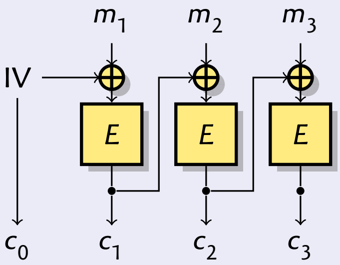
Generic Attacks Against Hash-based MACs

Chosen-prefix Collision Attacks

# Outline

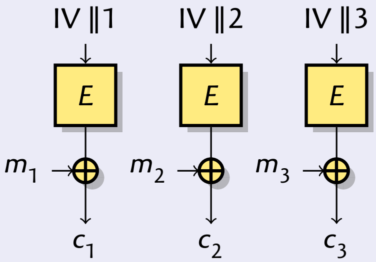
# CBC and CTR

## CBC mode




- Security proof up to $2^{n/2}$ queries
- $m_i \oplus m_j = c_{i-1} \oplus c_{j-1}$ if $c_i = c_j$
- Collisions reveals xor of two plaintext blocks

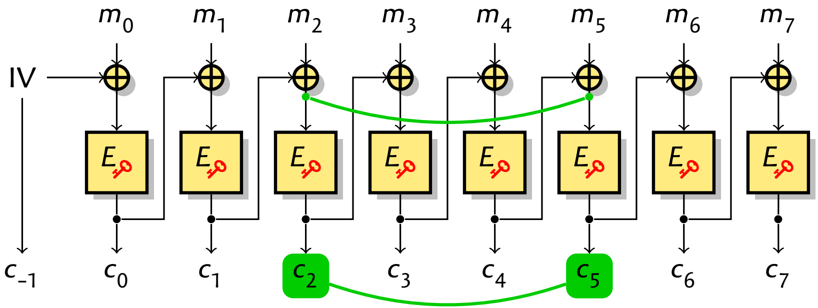
## CTR mode




- Security proof up to $2^{n/2}$ queries
- $m_i \oplus m_j \neq c_i \oplus c_j$ $\forall i,j$
- Distinguishing attack: Key stream doesn't collide

# CBC collisions

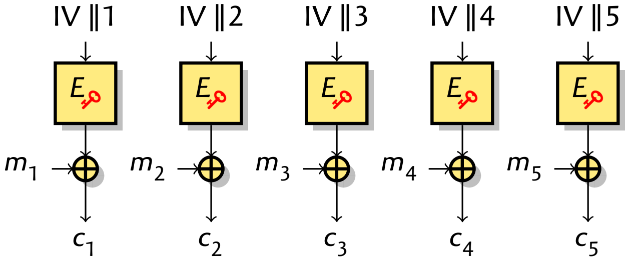
- Well known collision attack against CBC




- If $c_i = c_j$, then $c_{i-1} \oplus m_i = c_{j-1} \oplus m_j$
  - $m_i \oplus m_j = c_{i-1} \oplus c_{j-1}$
- Ciphertext collision reveals the xor of two plaintext blocks

# Birthday distinguishing on CTR

- Well known distinguisher against CTR




- All block-cipher inputs are distinct
- For all $i \neq j$, $m_i \oplus c_i \neq m_j \oplus c_j$
  - $m_i \oplus m_j \neq c_i \oplus c_j$
  - Hard to extract plaintext information from inequations
- Distinguisher: collision after $2^{n/2}$ blocks with random ciphertext

# The birthday bound

- Security of common modes of operations is limited by collisions
- With an $n$-bit state, collisions after $2^{n/2}$ blocks

## The birthday paradox

- Draw $r$ random values from $\{0,1\}^n$
  - Expected number of collisions is about $r^2/2^{n+1}$
  - Constant probability of having a collision with $r = \Theta(2^{n/2})$
- Variant: Let $\mathcal{A}, \mathcal{B}$ be random subsets of $\{0,1\}^n$
  - Expected number of matches $|\mathcal{A} \cap \mathcal{B}| \approx |\mathcal{A}| \times |\mathcal{B}|/2^n$
  - In particular, $\mathcal{A} \cap \mathcal{B} \neq \emptyset$ with high probability if $|\mathcal{A}| = |\mathcal{B}| = 2^{n/2}$

- Many generic attacks are based on finding special collisions

# Birthday security in practice

## Block size does matter


- State size is an important security parameter
  - Hash functions and stream ciphers use large state size $n \geq 160$
- Modern block ciphers have a 128-bit block size (*e.g.* AES)
  - $2^{64}$ blocks correspond to 256 EB
- Block ciphers from the 90's have a 64-bit block size (Blowfish, 3DES)
  - $2^{32}$ blocks correspond to 32 GB

- In 2016, 64-bit block ciphers were still used in practice
  - Around 1—2% of HTTPS connections used 3DES-CBC in 2015–2017
    - Mandatory support in TLS 1.0 and TLS 1.1
    - Supported for compatibility with old client/server
    - Many servers supported AES but preferred 3DES
  - OpenVPN used Blowfish-CBC by default

# Proof-of-concept Attack Demo: Sweet32 [Bhargavan & L, CCS'16]

- Target HTTPS with 3DES-CBC
  - BEAST man-in-the browser setting: chosen plaintext
  - Targeting authentication cookie: repeated secret
- Wait for collision between blocks from secret cookie and known plaintext
- Demo with Firefox (Linux), and IIS 6.0 (Windows Server 2003)
  - Default configuration of IIS 6.0 does not support AES

1. Generate traffic with malicious JavaScript
2. Capture on the network with `tcpdump`
3. Remove header, extract ciphertext at fixed position
4. Sort ciphertext (`stdxxl`), look for collisions

- Expected time: 38 hours for 785 GB.
- In practice: 30.5 hours for 610 GB.

# CBC and CTR

## CBC mode

- Security proof up to $2^{n/2}$ queries
- $m_i \oplus m_j = c_{i-1} \oplus c_{j-1}$ if $c_i = c_j$
- Collisions reveals xor of two plaintext blocks

## CTR mode

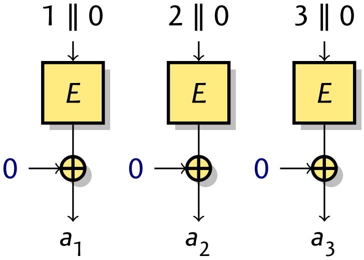
- Security proof up to $2^{n/2}$ queries
- $m_i \oplus m_j \neq c_i \oplus c_j$ $\forall i,j$
- Distinguishing attack: Key stream doesn't collide

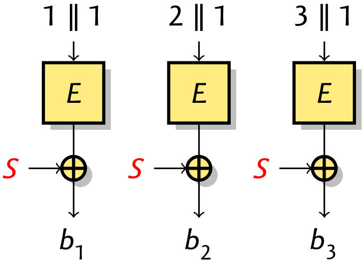
## *Recommendations* [Cryptography engineering, Ferguson, Schneier & Kohno]

*CTR leaks very little data. [...] It would be reasonable to limit the cipher mode to $2^{60}$ blocks, which allows you to encrypt $2^{64}$ bytes but restricts the leakage to a small fraction of a bit. When using CBC mode you should be a bit more restrictive. [...] We suggest limiting CBC encryption to $2^{32}$ blocks or so.*

# Plaintext recovery against CTR

- Collect two kinds of blocks




*Chosen plaintext blocks* $a_i = E(i)$

*Repeated secret* $b_j = E(j) \oplus S$

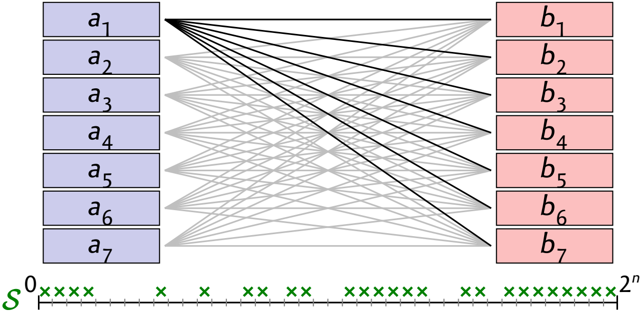
- $\forall i, j,\ a_i \neq S \oplus b_j$
- $\forall i, j,\ S \neq a_i \oplus b_j$

**Missing difference problem**

Given sets $\mathcal{A}, \mathcal{B} \subset \{0,1\}^n$
Find $S$ such that
$$\forall (a,b) \in \mathcal{A} \times \mathcal{B},\ S \neq a \oplus b$$

# Sieving algorithm

[McGrew, FSE'13]



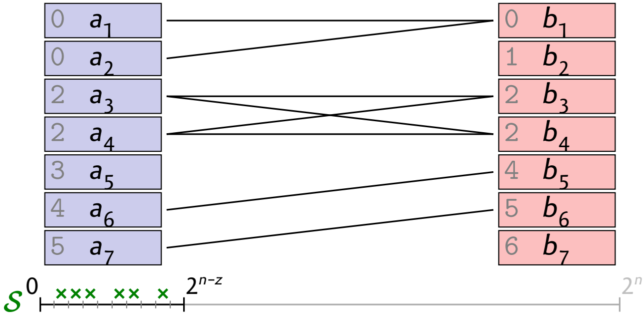

- Compute all $a_i \oplus b_j$, remove from a sieve $\mathcal{S}$

## *Analysis: Coupon collector problem*

- To exclude $2^n$ candidates $S$, we need $n \cdot 2^n$ values $a_i \oplus b_j$
  - Lists $\mathcal{A}$ and $\mathcal{B}$ of size $\sqrt{n} \cdot 2^{n/2}$. Complexity: $\tilde{\mathcal{O}}(2^n)$

# Known-prefix sieving

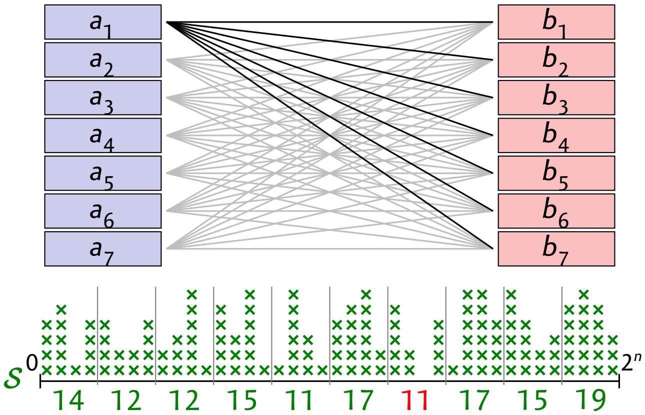
[L & Sibleyras, EC'18]




- Assume $S$ starts with $z$ zero bits
  - Smaller sieve
- Sort lists, consider $a_i$'s and $b_j$'s with matching prefix
- Complexity: $\tilde{\mathcal{O}}(2^{n/2})$ when $z \geq n/2$

# *Fast-convolution sieving*

[L & Sibleyras, EC'18]




- Use $2^{2n/3}$ queries, sieving with $2^{2n/3}$ buckets of $2^{n/3}$ elements
  - With high probability, smallest bucket corresponds to missing difference
- Sieving can be computed with Fast Walsh-Hadamard transform!
- Complexity: $\tilde{\mathcal{O}}(2^{2n/3})$ for arbitrary $S$

# Application of missing difference algorithms

## Application to CTR mode

- Assume a fixed secret encrypted repeatedly
- Assume that adversary can control the position of a fixed secret
  - Practical in the BEAST setting
- The adversary targets a block with $n/2$ secret bits and $n/2$ known bits
- Message recovery attack with birthday complexity $\tilde{O}(2^{n/2})$ using known-prefix sieving

## Applications to Wegman-Carter MAC

- Recovery of hash key is a missing difference problem
- Complexity $\tilde{O}(2^{2n/3})$ using fast-convolution sieving
- First partial key-recovery below $2^n$

# Summary: CBC and CTR

- CTR and CBC both leak plaintext data at the birthday bound
- Birthday attacks are practical against 64-bit block ciphers `Sweet32`

## Disclosure

- `Sweet32` disclosed in August 2016 `CVE-2016-2183, CVE-2016-6329`
- OpenVPN 2.4 has changed default to AES (December 2016)
- Mozilla has implemented data limits in Firefox 51 (1M records) (January 2017)
- NIST requires rekeying for 3DES after $2^{20}$ blocks rather $2^{32}$; 3DES deprecated in 2023

K. Bhargavan and G. Leurent. ACM CCS 2016
On the Practical (In-)Security of 64-bit Block Ciphers

G. Leurent and F. Sibleyras. Eurocrypt 2018
The Missing Difference Problem, and Its Applications to Counter Mode Encryption

# Outline

Generic Attacks Against Encryption Modes

Generic Attacks Against MACs in the Quantum Setting

Generic Attacks Against Hash Combiners

Generic Attacks Against Hash-based MACs

Chosen-prefix Collision Attacks

# *Expected impact of quantum computers*

- Recent progress toward building a large-scale quantum computer
- Some problems can be solved much faster with quantum computers
  - Up to exponential gains
  - But we don't expect to solve all NP problems

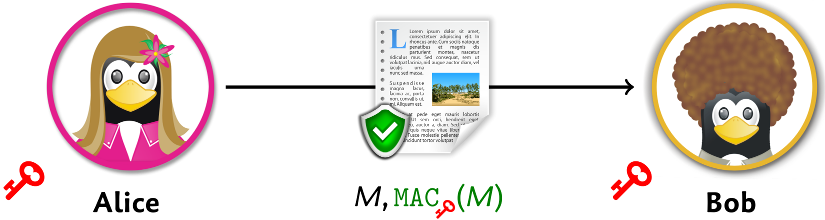
## *Impact on public-key cryptography*

- RSA, DH, ECC broken by Shor's algorithm
  - Breaks factoring and discrete log in polynomial time
  - Large effort to develop quantum-resistant algorithms (*e.g.* NIST)

## *Impact on symmetric cryptography*

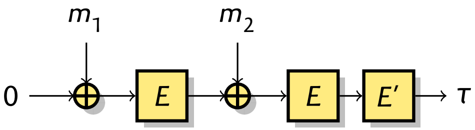
- Exhaustive search of $\kappa$-bit key in time $2^{\kappa/2}$ with Grover's algorithm
  - Common recommendation: double the key length (AES-256)
  - What is the security of modes of operation in the quantum setting?

# Message Authentication Codes (MAC)




- MAC: keyed function $\{0,1\}^* \to \{0,1\}^n$
  - Maps arbitrary-length message to fixed-length tag
- Alice uses a key 🔑 to compute a tag: $t = \mathtt{MAC}$🔑$(M)$
- Bob verifies the tag with the same key 🔑: $t \stackrel{?}{=} \mathtt{MAC}$🔑$(M)$
- Main security notion: forgery attack (hard to predict the tag of a message)

# CBC-MAC




- One of the earliest MACs, based on CBC encryption mode
- Security proof up to the birthday bound [Bellare, Kilian & Rogaway '94]

## Collision attack using two sets of $2^{n/2}$ messages

- $A_x = [0] \parallel x$
- $\mathtt{MAC}(A_x) = E'(E(x \oplus E([0])))$
- $B_y = [1] \parallel y$
- $\mathtt{MAC}(B_y) = E'(E(y \oplus E([1])))$
- $\mathtt{MAC}(A_x) = \mathtt{MAC}(B_y)$ iff $x \oplus E([0]) = y \oplus E([1])$
  - Deduce $\delta = E([0]) \oplus E([1]) = x \oplus y$
  - Produce forgeries: $\mathtt{MAC}([0] \parallel m) = \mathtt{MAC}([1] \parallel m \oplus \delta)$ for all $m$

# Simon's Algorithm

[Simon, SIAM'97]

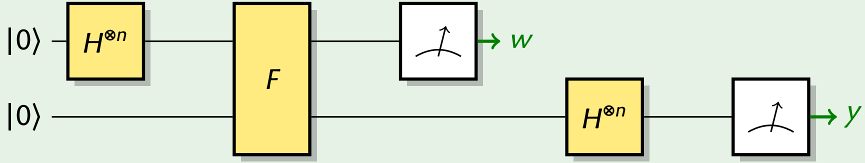
- Quantum algorithm to find collisions with extra structure

*Definition (Simon's problem)*

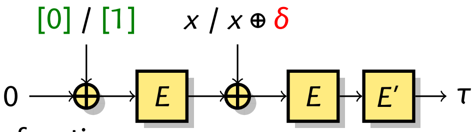
Given $f : \{0,1\}^n \to \{0,1\}^n$ such that there exists $\delta \in \{0,1\}^n$ with $f(x) = f(y) \Leftrightarrow x \oplus y = \delta$, find $\delta$.

- Classical algorithms require $\mathcal{O}(2^{n/2})$ queries (finding collisions)
- Simon's algorithm require $\mathcal{O}(n)$ quantum queries

*One step of Simon's algorithm returns $y \perp \delta$*



# Quantum attack against CBC-MAC [Kaplan, L, Leverrier, Naya-Plasencia, C'16]




1. Consider the following function:

$$f : \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$$
$$b, x \mapsto \texttt{MAC}([b] \parallel x) = E'\Big(E\big(x \oplus E([b])\big)\Big)$$

$$f(b,x) = f(b',x') \iff \begin{cases} b = b' \text{ and } x = x' & \text{or} \\ b \neq b' \text{ and } x \oplus x' = E([0]) \oplus E([1]) \end{cases}$$

- $f$ has period $1 \parallel \delta$, with $\delta = E([0]) \oplus E([1])$

2. Use Simon's algorithm to recover $1 \parallel \delta$
3. Produce forgeries: $\texttt{MAC}([0] \parallel m) = \texttt{MAC}([1] \parallel m \oplus \delta)$

# Generalization

## Simon's algorithm breaks most common MAC and AEAD modes

1. Define a function $f$ with $f(x \oplus \delta) = f(x)$ for some interesting $\delta$
   - Often corresponds to a classical collision attack
2. Build quantum circuit for $f$, use Simon's algorithm to recover $\delta$
   - $t = \mathcal{O}(n)$ quantum queries
3. Use $\delta$ to produce forgeries

- Strong assumption: superposition queries

M. Kaplan, G. Leurent, A. Leverrier, M. Naya-Plasencia — CRYPTO 2016
Breaking Symmetric Cryptosystems Using Quantum Period Finding

X. Bonnetain, G. Leurent, M. Naya-Plasencia, A. Schrottenloher — ASIACRYPT 2021
Quantum Linearization Attacks

# Quantum security of modes of operations

## *Encryption modes* [Unruh, Targhi, Tabia & Anand, PQC'16]

- Common encryption modes are quantum-secure (CBC, CTR)

## *Authentication modes (MACs)*

- Many MACs and AEAD broken with superposition queries
  - `CBC-MAC`, `PMAC`, `GMAC`, `GCM`, `OCB`, ... [KLLNP, Crypto'16]
  - `ΘCB`, `LightMAC`, `LightMAC+`... [BLNS, AC'21]
- But Cascade/HMAC is secure [Song & Yun, Crypto'17]

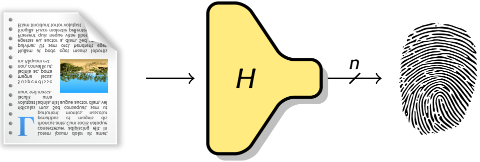
## *Authenticated-encryption modes*

- Encrypt-then-MAC is secure [Soukharev, Jao, Seshadri, PQC'16]
- New proposal with rate 1: QCB [BBCLNSS, AC'21]

# Outline

# Hash functions

- Hash function: public function $\{0,1\}^* \to \{0,1\}^n$
- Should behave like a random function
  - No structural property
  - Cryptographic properties without any key!
- Concrete security goals




**Preimage attack**

Given $F$ and $\overline{H}$, find $M$ s.t. $F(M) = \overline{H}$. Ideal security: $2^n$.

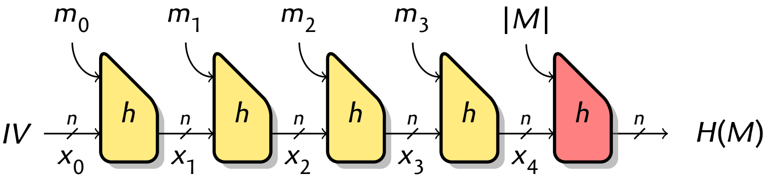
**Second-preimage attack**

Given $F$ and $M_1$, find $M_2 \neq M_1$ s.t. $F(M_1) = F(M_2)$. Ideal security: $2^n$.

**Collision attack**

Given $F$, find $M_1 \neq M_2$ s.t. $F(M_1) = F(M_2)$. Ideal security: $2^{n/2}$.

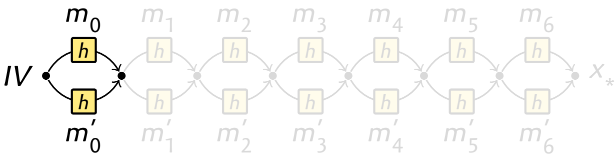
# The Merkle-Damgård construction (SHA-1, SHA-2)




- $n$-bit state, compression function $h : \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^n$
- Finalization using message length (MD strengthening)
- Notation: Iterated compression function $h^*$
  - $h^*(x, m_1 \parallel m_2 \parallel m_3) = h(h(h(x, m_1), m_2), m_3)$
- Security reduction:
  - Hash collisions imply compression function collision (generic security $2^{n/2}$)
  - Hash preimages imply finalization preimages (generic security $2^n$)
- Generic attacks above the birthday bound, exploiting collisions in smart ways
  - Second-preimage for long challenges [Kelsey & Schneier, EC'05]
  - Nostradamus attack / herding [Kelsey & Kohno, EC'06]

# Multicollisions

[Joux, Crypto '04]



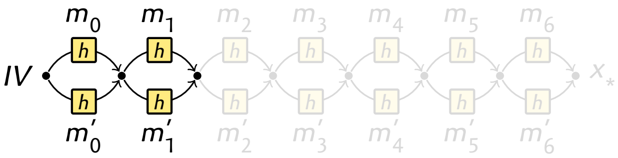

1. Find a collision pair $m_0/m'_0$ starting from $IV$
2. Find a collision pair $m_1/m'_1$ starting from $x_1 = h^*(m_0)$
3. Repeat $t$ times
4. This yields $2^t$ messages with the same hash:

| | | | |
|---|---|---|---|
| $m_0 m_1 m_2 \dots$ | $m'_0 m_1 m_2 \dots$ | $m_0 m'_1 m_2 \dots$ | $m'_0 m'_1 m_2 \dots$ |
| $m_0 m_1 m'_2 \dots$ | $m'_0 m_1 m'_2 \dots$ | $m_0 m'_1 m'_2 \dots$ | $m'_0 m'_1 m'_2 \dots$ |

- Complexity $t \cdot 2^{n/2}$ vs. $\approx 2^{\frac{2^t-1}{2^t}n}$ for a random function

# Multicollisions

[Joux, Crypto '04]



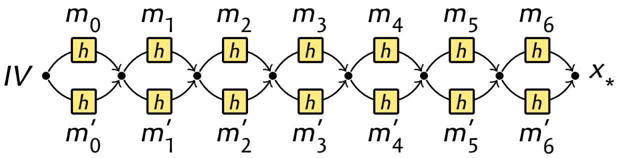

1. Find a collision pair $m_0 / m'_0$ starting from $IV$
2. Find a collision pair $m_1 / m'_1$ starting from $x_1 = h^*(m_0)$
3. Repeat $t$ times
4. This yields $2^t$ messages with the same hash:

| | | | |
|---|---|---|---|
| $m_0 m_1 m_2 \dots$ | $m'_0 m_1 m_2 \dots$ | $m_0 m'_1 m_2 \dots$ | $m'_0 m'_1 m_2 \dots$ |
| $m_0 m_1 m'_2 \dots$ | $m'_0 m_1 m'_2 \dots$ | $m_0 m'_1 m'_2 \dots$ | $m'_0 m'_1 m'_2 \dots$ |

- Complexity $t \cdot 2^{n/2}$ vs. $\approx 2^{\frac{2^t-1}{2^t} n}$ for a random function

# Multicollisions

[Joux, Crypto '04]

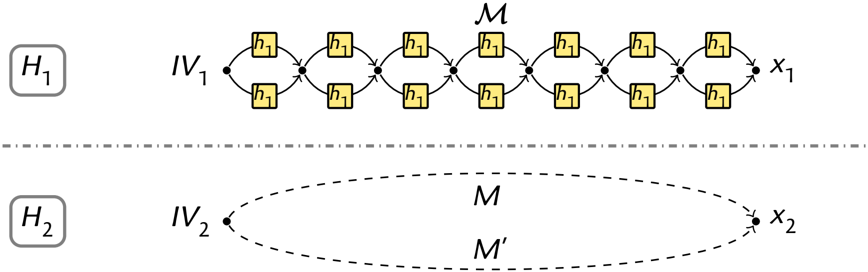



1. Find a collision pair $m_0/m'_0$ starting from $IV$
2. Find a collision pair $m_1/m'_1$ starting from $x_1 = h^*(m_0)$
3. Repeat $t$ times
4. This yields $2^t$ messages with the same hash:

$$m_0 m_1 m_2 \ldots \quad m'_0 m_1 m_2 \ldots \quad m_0 m'_1 m_2 \ldots \quad m'_0 m'_1 m_2 \ldots$$
$$m_0 m_1 m'_2 \ldots \quad m'_0 m_1 m'_2 \ldots \quad m_0 m'_1 m'_2 \ldots \quad m'_0 m'_1 m'_2 \ldots$$

- Complexity $t \cdot 2^{n/2}$ vs. $\approx 2^{\frac{2^t-1}{2^t}n}$ for a random function

# Collision attack for $H_1(M) \parallel H_2(M)$



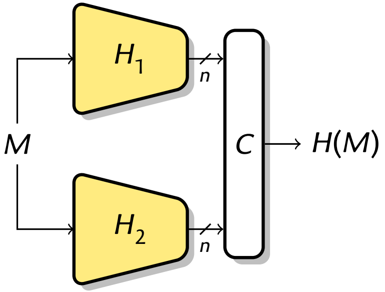

1. Build a $2^{n/2}$-multicollision for $H_1$

$$\forall M \in \mathcal{M}, H_1(M) = x_1$$

2. Find $M, M' \in \mathcal{M}$ s.t. $H_2(M) = H_2(M')$

- Complexity $\mathcal{O}(n \cdot 2^{n/2})$ vs. $2^n$ for a $2n$-bit hash function.

# Combining two hash functions




*"In order to make the PRF as secure as possible, it uses two hash algorithms in a way which should guarantee its security if either algorithm remains secure."*
– RFC 2246 (TLS 1.0)

Classical combiners:

- Concatenation: $H_1(M) \parallel H_2(M)$
- Xor: $H_1(M) \oplus H_2(M)$

*"The whole is greater than the sum of its parts"*
– Aristotle

# Generic attacks against combiners

## Concatenation combiner

- $H(M) = H_1(M) \parallel H_2(M)$
- $2n$-bit output
- Generic security: attacks / proofs

| | attacks | proofs |
|---|---|---|
| Collisions: | $2^{n/2}$ | $2^{n/2}$ |
| Preimages: | $2^n$ | $2^n$ |
| Non-ideal: | $2^{n/2}$ | $2^{n/2}$ |

## XOR combiner

- $H(M) = H_1(M) \oplus H_2(M)$
- $n$-bit output
- Generic security: attacks / proofs

| | attacks | proofs |
|---|---|---|
| Collisions: | $2^{n/2}$ | $2^{n/2}$ |
| Preimages: | $2^n$ | $2^{n/2}$ |
| Non-ideal: | $2^{n/2}$ | $2^{n/2}$ |

## Surprising result [Joux, C'04]

If $H_1$ and $H_2$ are good MD hash functions,
$H_1 \parallel H_2$ is not stronger!

## Surprising result [L & Wang, EC'15]

If $H_1$ and $H_2$ are good MD hash functions,
$H_1 \oplus H_2$ is weaker!

# Generic attacks against combiners

## Concatenation combiner

- $H(M) = H_1(M) \parallel H_2(M)$
- $2n$-bit output
- Generic security: attacks / proofs
  - Collisions: $2^{n/2}$ $2^{n/2}$
  - Preimages: $2^n$ $2^n$
  - Non-ideal: $2^{n/2}$ $2^{n/2}$

## XOR combiner

- $H(M) = H_1(M) \oplus H_2(M)$
- $n$-bit output
- Generic security: attacks / proofs
  - Collisions: $2^{n/2}$ $2^{n/2}$
  - Preimages: $2^{3n/5}$ $2^{n/2}$
  - Non-ideal: $2^{n/2}$ $2^{n/2}$

### *Surprising result* [Joux, C'04]

If $H_1$ and $H_2$ are good MD hash functions, $H_1 \parallel H_2$ is not stronger!

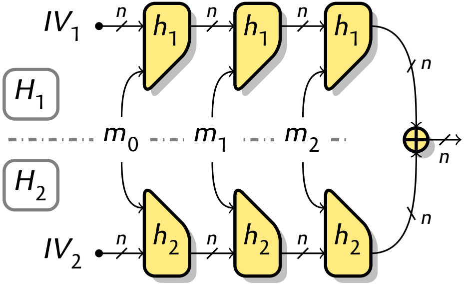
### *Surprising result* [L & Wang, EC'15]

If $H_1$ and $H_2$ are good MD hash functions, $H_1 \oplus H_2$ is weaker!

# Preimage attack against Xor combiner

[L & Wang, EC'15]

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



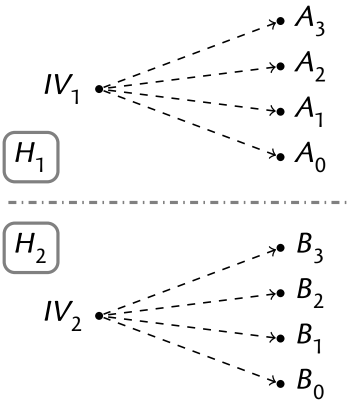

Strategy:

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 $r$, match $\{g_1(h_1(A_j, r))\}$ and $\{g_2(h_2(B_k, r)) \oplus \overline{H}\}$
   - If there is a match $(j, k)$:
     Get $\mathbf{M}_{jk}$, preimage is $M = \mathbf{M}_{jk} \parallel r$
   - Complexity $\mathcal{O}(2^n / \min\{|\mathcal{A}|, |\mathcal{B}|\})$

# Preimage attack against Xor combiner

[L & Wang, EC'15]

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



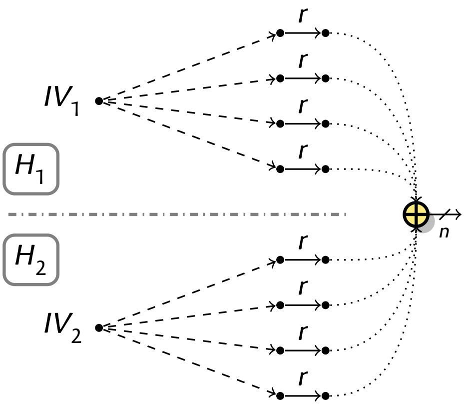

Strategy:

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 $r$, match $\{g_1(h_1(A_j, r))\}$ and $\{g_2(h_2(B_k, r)) \oplus \overline{H}\}$
   - If there is a match $(j, k)$: Get $\mathbf{M}_{jk}$, preimage is $M = \mathbf{M}_{jk} \parallel r$
   - Complexity $\mathcal{O}(2^n / \min\{|\mathcal{A}|, |\mathcal{B}|\})$

# Preimage attack against Xor combiner

[L & Wang, EC'15]

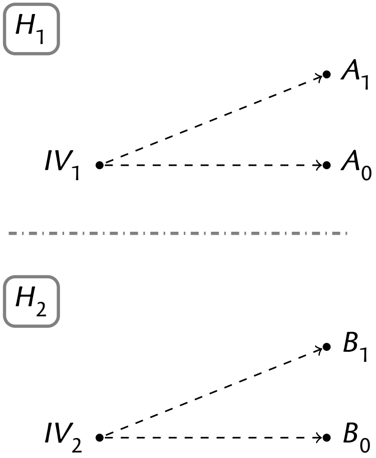
$H(M) = H_1(M) \oplus H_2(M)$




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 $r$, match $\{g_1(h_1(A_j, r))\}$ and $\{g_2(h_2(B_k, r)) \oplus \overline{H}\}$
   - If there is a match $(j, k)$: Get $\mathbf{M}_{jk}$, preimage is $M = \mathbf{M}_{jk} \parallel r$
   - Complexity $\mathcal{O}(2^n / \min\{|\mathcal{A}|, |\mathcal{B}|\})$

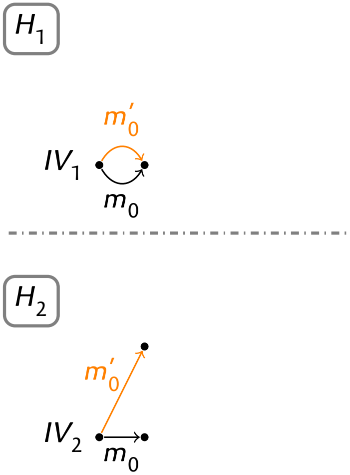
# Example: $2 \times 2$ structure




1. Find a collision $(m_0, m_0')$ in $H_1$
2. Find a collision $(m_1, m_1')$ in $h_2^*(h_2^*(m_0))$
3. Find a multicollision starting from $h_2^*(m_0' \parallel m_1)$
4. Select messages $(m_2, m_2')$ from the multicollision such that: $h_1^*(m_0 \parallel m_1 \parallel m_2) = h_1^*(m_0 \parallel m_1' \parallel m_2')$

- $H(m_0 \parallel m_1 \parallel m_2) = (A_0, B_0)$
- $H(m_0' \parallel m_1 \parallel m_2) = (A_0, B_1)$
- $H(m_0 \parallel m_1' \parallel m_2) = (A_1, B_0)$
- $H(m_0' \parallel m_1 \parallel m_2') = (A_1, B_1)$

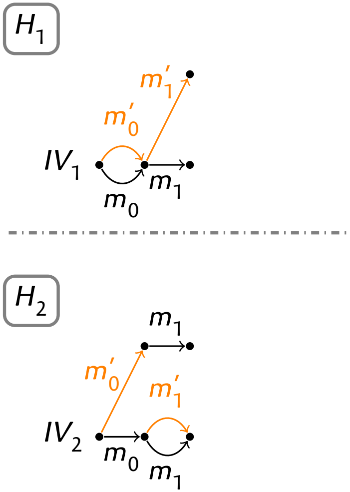
# Example: 2 × 2 structure




1. Find a collision $(m_0, m'_0)$ in $H_1$
2. Find a collision $(m_1, m'_1)$ in $h_2^*(h_2^*(m_0))$
3. Find a multicollision starting from $h_2^*(m'_0 \| m_1)$
4. Select messages $(m_2, m'_2)$ from the multicollision such that: $h_1^*(m_0 \| m_1 \| m_2) = h_1^*(m_0 \| m'_1 \| m'_2)$

- $H(m_0 \| m_1 \| m_2) = (A_0, B_0)$
- $H(m'_0 \| m_1 \| m_2) = (A_0, B_1)$
- $H(m_0 \| m'_1 \| m_2) = (A_1, B_0)$
- $H(m'_0 \| m_1 \| m'_2) = (A_1, B_1)$

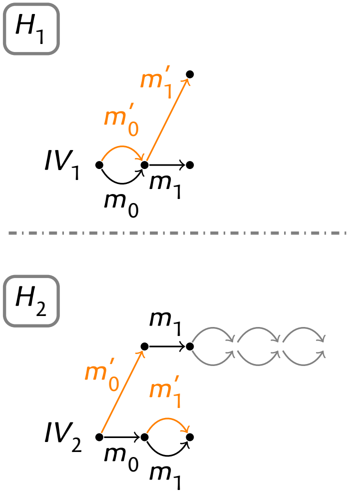
# Example: 2 × 2 structure




1. Find a collision $(m_0, m_0')$ in $H_1$
2. Find a collision $(m_1, m_1')$ in $h_2^*(h_2^*(m_0))$
3. Find a multicollision starting from $h_2^*(m_0' \| m_1)$
4. Select messages $(m_2, m_2')$ from the multicollision such that: $h_1^*(m_0 \| m_1 \| m_2) = h_1^*(m_0 \| m_1' \| m_2')$

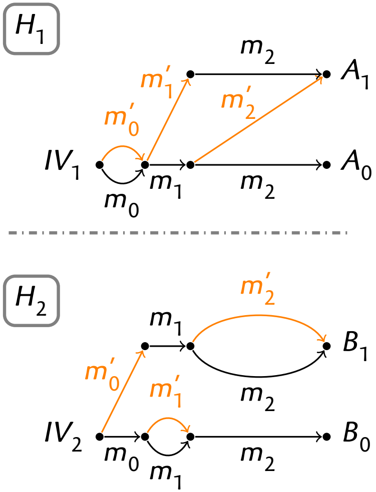
- $H(m_0 \| m_1 \| m_2) = (A_0, B_0)$
- $H(m_0' \| m_1 \| m_2) = (A_0, B_1)$
- $H(m_0 \| m_1' \| m_2) = (A_1, B_0)$
- $H(m_0' \| m_1 \| m_2') = (A_1, B_1)$

# *Example: 2 × 2 structure*




1. Find a collision $(m_0, m'_0)$ in $H_1$
2. Find a collision $(m_1, m'_1)$ in $h^*_2(h^*_2(m_0))$
3. Find a multicollision starting from $h^*_2(m'_0 \parallel m_1)$
4. Select messages $(m_2, m'_2)$ from the multicollision such that: $h^*_1(m_0 \parallel m_1 \parallel m_2) = h^*_1(m_0 \parallel m'_1 \parallel m'_2)$
   - $H(m_0 \parallel m_1 \parallel m_2) = (A_0, B_0)$
   - $H(m'_0 \parallel m_1 \parallel m_2) = (A_0, B_1)$
   - $H(m_0 \parallel m'_1 \parallel m_2) = (A_1, B_0)$
   - $H(m'_0 \parallel m_1 \parallel m'_2) = (A_1, B_1)$

# Example: 2 × 2 structure



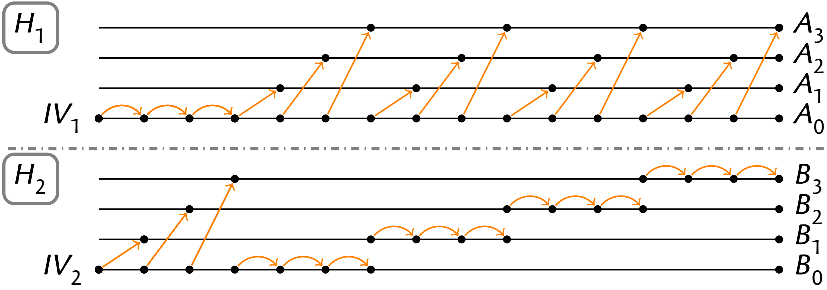

1. Find a collision $(m_0, m'_0)$ in $H_1$
2. Find a collision $(m_1, m'_1)$ in $h_2^*(h_2^*(m_0))$
3. Find a multicollision starting from $h_2^*(m'_0 \parallel m_1)$
4. Select messages $(m_2, m'_2)$ from the multicollision such that: $h_1^*(m_0 \parallel m_1 \parallel m_2) = h_1^*(m_0 \parallel m'_1 \parallel m'_2)$

- $H(m_0 \parallel m_1 \parallel m_2) = (A_0, B_0)$
- $H(m'_0 \parallel m_1 \parallel m_2) = (A_0, B_1)$
- $H(m_0 \parallel m'_1 \parallel m_2) = (A_1, B_0)$
- $H(m'_0 \parallel m_1 \parallel m'_2) = (A_1, B_1)$

# Interchange structure

[L & Wang, EC'15]

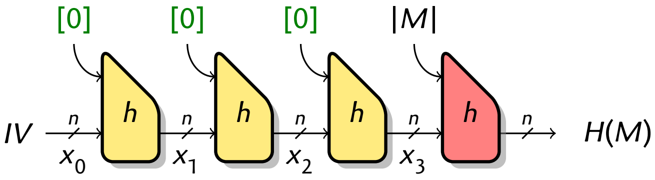
- We generalize this construction to a larger set of output states




- Complexity $\tilde{\mathcal{O}}(2^{n/2+2t})$ to build a structure with $|\mathcal{A}| = |\mathcal{B}| = 2^t$
- Complexity $\tilde{\mathcal{O}}(2^{5n/6})$ for preimages (tradeoff)

# Alternative structure using cycles

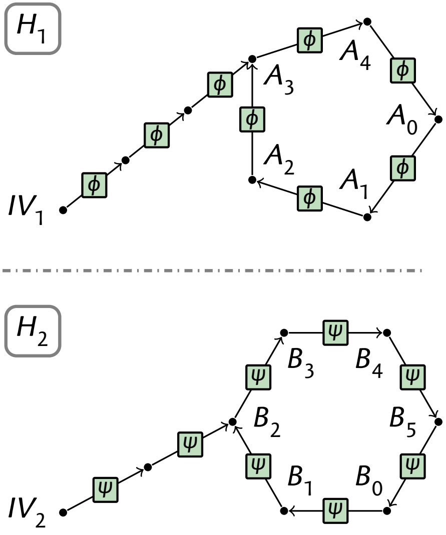
- New presentation of "multicycles" [Bao, Wang, Guo, Gu, C'17]




- Using a long message repeating a fixed block $M = [0]^\lambda$, we iterate fixed functions:

$$\phi : x \mapsto h_1(x, [0])$$
$$\psi : x \mapsto h_2(x, [0])$$

# Alternative structure using cycles




- Use cyclic nodes as end-point:
  - $\mathcal{A} = H_1$ cycle, length $\ell_1$
  - $\mathcal{B} = H_2$ cycle, length $\ell_2$
- With suitable naming, for $\lambda$ large enough:

$$h_1^*([0]^\lambda) = A_{\lambda \bmod \ell_1} \qquad h_2^*([0]^\lambda) = B_{\lambda \bmod \ell_2}$$

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

$$\begin{cases} h_1^*([0]^\lambda) = A_j \\ h_2^*([0]^\lambda) = B_k \end{cases} \iff \begin{cases} \lambda \bmod \ell_1 = i \\ \lambda \bmod \ell_2 = j \end{cases}$$

  - $\lambda$ uniformly distributed in range of size $\ell_1 \ell_2$
  - $\Pr[\lambda < 2^t] \approx 2^{n-t}$
- Complexity $\tilde{\mathcal{O}}(2^{3n/4})$ for preimages (tradeoff)

# Summary: Preimage attack for $H_1(M) \oplus H_2(M)$

## Interchange structure

- Complexity $\tilde{\mathcal{O}}(2^{5n/6})$ [LW15]
- Works for Merkle-Damgård and HAIFA
  - Finalization function, block counter at each round
- Short messages: length $\tilde{\mathcal{O}}(2^{n/3})$

## Using cycles

- Complexity $\tilde{\mathcal{O}}(2^{3n/4})$ (simple)
- Complexity $\tilde{\mathcal{O}}(2^{5n/8})$ [BWGG17]
- Complexity $\tilde{\mathcal{O}}(2^{11n/18})$ [BDGLW20]
- Complexity $\tilde{\mathcal{O}}(2^{3n/5})$ (new)
- Works only for Merkle-Damgård mode
  - Finalization function, same function at each step
- Long messages: length $\tilde{\mathcal{O}}(2^{3n/5})$

G. Leurent, L. Wang — Eurocrypt 2015
The Sum Can Be Weaker Than Each Part

Z. Bao, I. Dinur, J. Guo, G. Leurent, L. Wang — Journal of Cryptology 2020
Generic Attacks on Hash Combiners

# Outline

Generic Attacks Against Encryption Modes

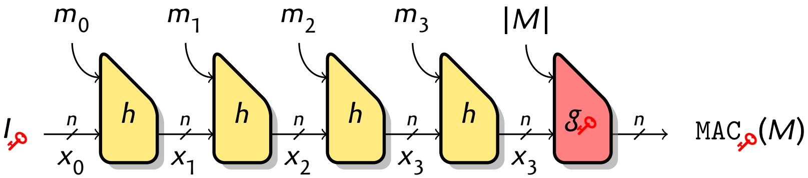
Generic Attacks Against MACs in the Quantum Setting

Generic Attacks Against Hash Combiners

**Generic Attacks Against Hash-based MACs**

Chosen-prefix Collision Attacks

# Hash-based MACs




- $n$-bit chaining value, $n$-bit MAC
- $\kappa$-bit key

we focus on $n = \kappa$

- Key-dependent initial value $I$
- Unkeyed compression function $h$
- Key-dependent finalization, with message length $g$
- Examples: HMAC, envelope MAC, sandwich MAC
- Security proofs up to the birthday bound

# Summary: Cryptanalysis of hash-based MACs

- Attacks using properties of functional graphs, and entropy loss of iteration
- Generic state-recovery attacks
  - Complexity $\tilde{O}(2^{n/2})$ for Merkle-Damgård (tight)
  - Complexity $\tilde{O}(2^{4n/5})$ for HAIFA (not tight)
- Generic key-recovery attack against HMAC with a checksum (HMAC-GOST)
  - Complexity $\tilde{O}(2^{3n/4})$ for Merkle-Damgård (not tight)
  - Complexity $\tilde{O}(2^{4n/5})$ for HAIFA (not tight)
  - The checksum actually makes the hash function weaker!

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

I. Dinur, G. Leurent — Crypto 2014 & Algorithmica
Improved Generic Attacks against Hash-Based MACs and HAIFA

# Outline

Generic Attacks Against Encryption Modes

Generic Attacks Against MACs in the Quantum Setting

Generic Attacks Against Hash Combiners

Generic Attacks Against Hash-based MACs

*Chosen-prefix Collision Attacks*

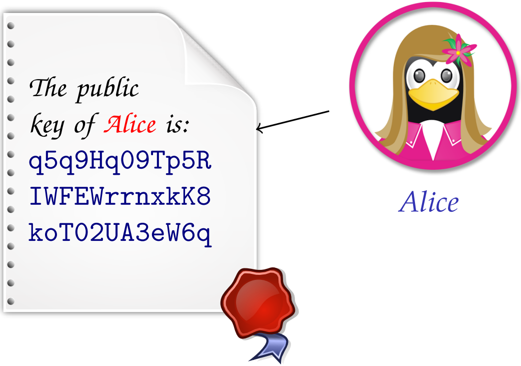
- Chosen-prefix collisions
- Application to SHA-1

# Hash function security

- Many collision attacks in the 2000s
  - MD4 : 3 [NSKO05]
  - MD5 : $2^{16}$ [WY05,SSA+09]
  - SHA-1: $2^{65}$ [WYY05,SBK+17]
- Hash functions are used in many constructions/protocols
  - Signatures (hash-and-sign)
  - HMAC
  - TLS
  - ...
- Impact of collisions on these constructions is not clear

- What is the practical impact of collision attacks?
  - Some constructions are secure without assuming collision resistance (*e.g.* HMAC)
  - Can we extend attacks to break applications?

# *Attacking key certification* [Stevens, Lenstra & de Weger, EC'07]




### *PKI Infrastructure*

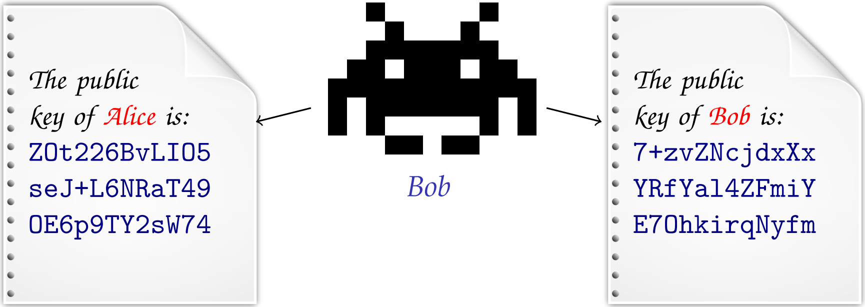
- Alice generates key
- Asks CA to sign
- Certificate proves ID

### *Impersonation attack*

1. Bob creates keys s.t. $H(\texttt{Alice}||k_A) = H(\texttt{Bob}||k_B)$
2. Bob asks CA to certify his key $k_B$
3. Bob copies the signature to $k_A$, impersonates Alice

# *Attacking key certification*

[Stevens, Lenstra & de Weger, EC'07]




## *PKI Infrastructure*

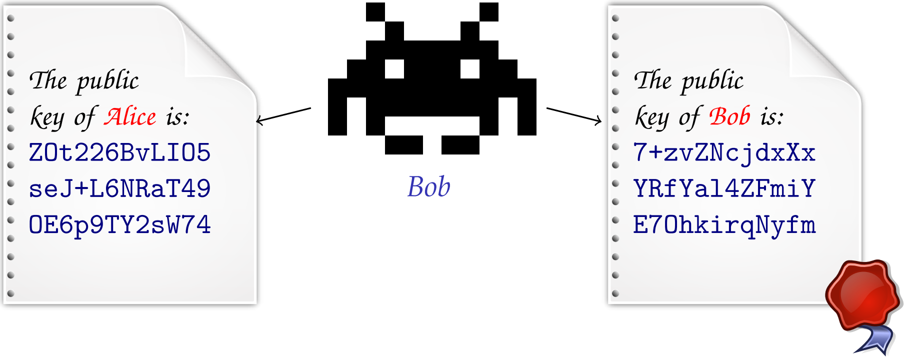
- Alice generates key
- Asks CA to sign
- Certificate proves ID

## *Impersonation attack*

1. Bob creates keys s.t. $H(\texttt{Alice}\|🔑_A) = H(\texttt{Bob}\|🔑_B)$
2. Bob asks CA to certify his key $🔑_B$
3. Bob copies the signature to $🔑_A$, impersonates Alice

# *Attacking key certification*

[Stevens, Lenstra & de Weger, EC'07]




## *PKI Infrastructure*

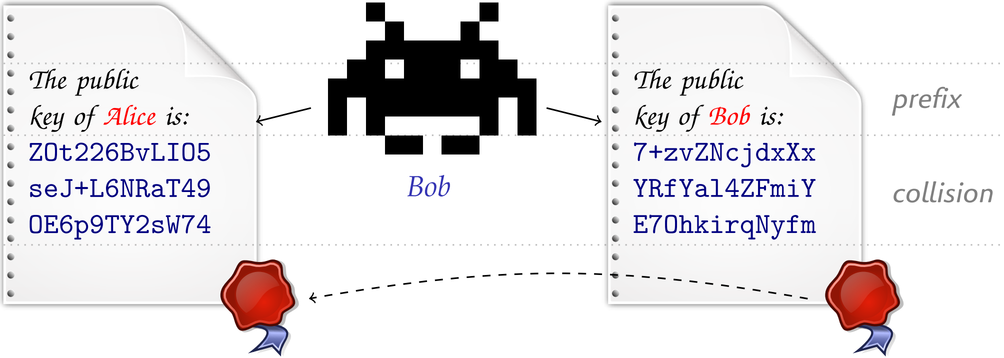
- Alice generates key
- Asks CA to sign
- Certificate proves ID

## *Impersonation attack*

1. Bob creates keys s.t. $H(\texttt{Alice}||🔑_A) = H(\texttt{Bob}||🔑_B)$
2. Bob asks CA to certify his key $🔑_B$
3. Bob copies the signature to $🔑_A$, impersonates Alice

# Attacking key certification

[Stevens, Lenstra & de Weger, EC'07]




## PKI Infrastructure

- Alice generates key
- Asks CA to sign
- Certificate proves ID

## Impersonation attack

1. Bob creates keys s.t. $H$(Alice||🔑$_A$) = $H$(Bob||🔑$_B$)
2. Bob asks CA to certify his key 🔑$_B$
3. Bob copies the signature to 🔑$_A$, impersonates Alice

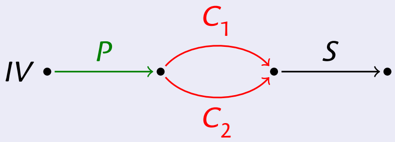
# Chosen-Prefix Collisions

[Stevens, Lenstra & de Weger, EC'07]

- Collisions are hard to exploit: garbage collision blocks $C_i$

## Identical-prefix collision

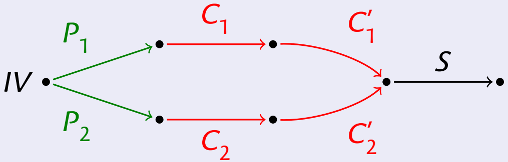
- Given IV, find $M_1 \neq M_2$ s. t. $H(M_1) = H(M_2)$




- Arbitrary common prefix/suffix, random collision blocks
- Breaks integrity verification
- Colliding PDFs (breaks signature?)

## Chosen-prefix (CP) collision

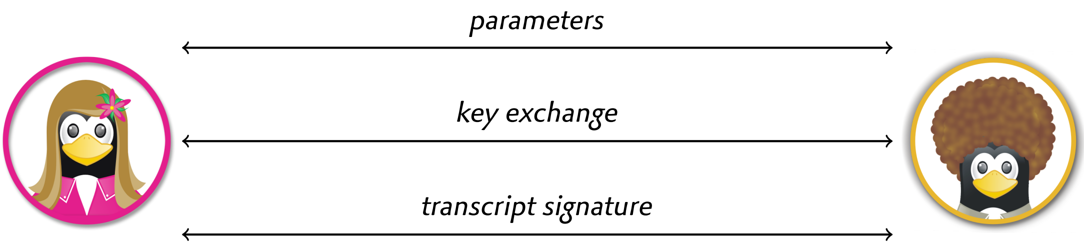
- Given $P_1, P_2$, find $M_1 \neq M_2$ s. t. $H(P_1 \parallel M_1) = H(P_2 \parallel M_2)$




- Two arbitrary prefixes, common suffix random collision blocks
- Attack more difficult
- Breaks certificates

# Transcript-collision attacks: SLOTH

[Bhargavan & L, NDSS'16]



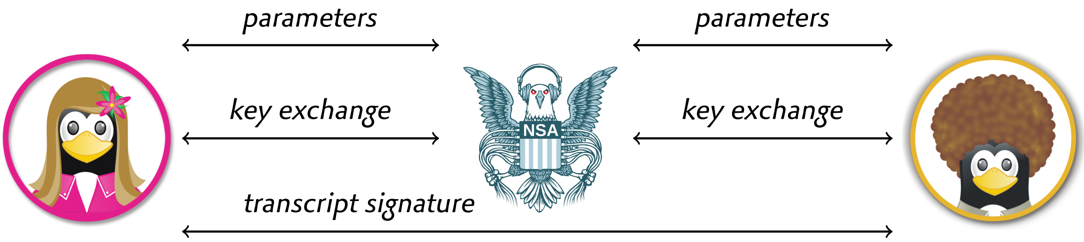

### Opening a secure channel (e.g. TLS/SSH/IKE)

1. Negotiate parameters
2. Key exchange (Diffie-Hellman)
3. Sign the transcript to authenticate the parties

### Man-in-the-middle attack

- Make the transcripts collide
- Transfer the signature
- Applicable to TLS/SSH/IKE

# *Transcript-collision attacks: SLOTH* [Bhargavan & L, NDSS'16]




*Opening a secure channel (e.g. TLS/SSH/IKE)*

1. Negotiate parameters
2. Key exchange (Diffie-Hellman)
3. Sign the transcript to authenticate the parties

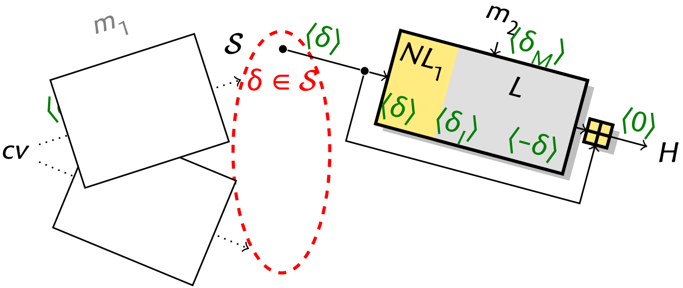
*Man-in-the-middle attack*

- Make the transcripts collide
- Transfer the signature
- Applicable to TLS/SSH/IKE

# *Chosen-prefix collision attack* [Stevens, Lenstra & de Weger, EC'07]

*Main idea*

Find a set of "nice" chaining value differences $\mathcal{S}$



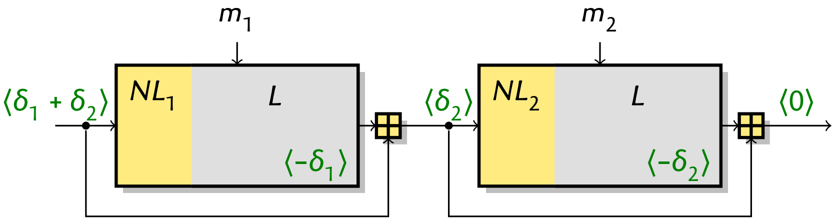

1. Birthday phase
   - Find $m_1, m_1'$ such that $H(P_1 \parallel m_1) - H(P_2 \parallel m_1') \in \mathcal{S}$
   - Complexity about $\sqrt{2^n / |\mathcal{S}|}$
2. Near-collision phase
   - Adjust non-linear trail
   - Erase the state difference, using near-collision blocks

*Improvement: building a larger set $\mathcal{S}$*

- The bottleneck of the SHA-1 attack is the birthday phase

# Multi-block technique

[L & Peyrin, EC'19]




- Assume we reach a set of output differences $\mathcal{D}$ with one block
- Assume we can build trails from any input difference

- With two blocks, we can reach a set of output differences:
  $\mathcal{S}_2 := \{\delta_1 + \delta_2 : \delta_1, \delta_2 \in \mathcal{D}\}$
- With $n$ blocks:
  $\mathcal{S}_n := \{\delta_1 + \delta_2 + \cdots \delta_n : \delta_1, \delta_2, \ldots \delta_n \in \mathcal{D}\}$

- Build a graph with all differences in $\mathcal{S}$
- Use graph algorithms to select the trail for each block

## Implementing Chosen-prefix Collisions [L & Peyrin, UX'20]

- Our method reuses the cryptanalysis results on SHA-1
  - Turning a collision attack into a chosen-prefix collision
- We implemented the full CPC attack
  - 2 months using 900 GPU (GTX 1060)
  - Complexity improvements to near-collision blocks search (factor 8 ~ 10)

| | | |
|---|---|---|
| *identical-prefix collision* | from $2^{64.7}$ to $2^{61.6}$ | (11 kUS$ in GPU rental) |
| *chosen-prefix collision* | from $2^{77.1}$ to $2^{63.5}$ | (45 kUS$ in GPU rental) |

- Application to PGP Web-of-Trust
  - Impersonation attack using colliding certificates
  - Implemented in practice

# SHA-1 Summary

- SHA-1 must be deprecated: signatures can now be abused in practice

- SHA-1 certificates deprecated by web browsers (early 2017)
- GnuPGv2 stopped trusting SHA-1 certificates (2019-11) `CVE-2019-14855`
- TLS 1.0 and 1.1 have been deprecated (RFC8996 – 2021-04) `CVE-2015-7575`
  - Transcript signed with MD5 ‖ SHA-1
- SHA-1 deprecated for TLS in-protocol signatures (RFC9155 – 2021-12)
- OpenSSH has disabled RSA-SHA1 signatures by default (2021-09)

K. Bhargavan, G. Leurent — NDSS 2016
Transcript collision attacks: Breaking authentication in TLS, IKE and SSH.

G. Leurent, T. Peyrin — Eurocrypt 2019
From Collisions to Chosen-Prefix Collisions – Application to Full SHA-1

G. Leurent, T. Peyrin — Usenix 2020
SHA-1 is a Shambles: First CP Collision on SHA-1, Application to the PGP Web of Trust

# Conclusion: Cryptanalysis beyond primitives

*Fun research area*

- Interesting algorithmic problems for generic attacks
- Concrete attacks with practical impact

- Modes and protocols usually studied with proofs but cryptanalysis is useful
  - Mistakes in proofs
  - Gap between proofs and attacks
  - Different security degradation after the birthday bound
  - Usage when the proof does not apply
  - Security proof becomes invalid in different model

*Take away*

Don't assume security above the birthday bound without a proof