# The Missing Difference Problem, and its Applications to Counter Mode Encryption

Gaëtan Leurent, Ferdinand Sibleyras

# The Missing Difference Problem, and its Applications to Counter Mode Encryption

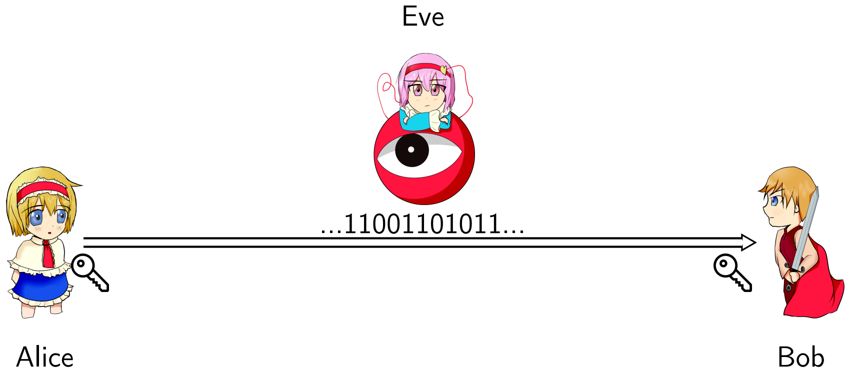
Gaëtan Leurent, Ferdinand Sibleyras

Inria, équipe SECRET

Journées Codage & Cryptographie 2018

# Introduction

- **Cryptography:** Alice encrypts then sends messages to Bob.
- **Symmetric:** Alice and Bob share the same key.
- **Public channel:** Eve (attacker) can see and/or manipulate what is being sent.



# Introduction

## Block Cipher

$$E_k : \{0,1\}^n \rightarrow \{0,1\}^n$$

A family of **permutations** indexed by a key (AES, 3DES, ...) where $n$ is the bit size of the permutation or block's size.

# Introduction

## Block Cipher

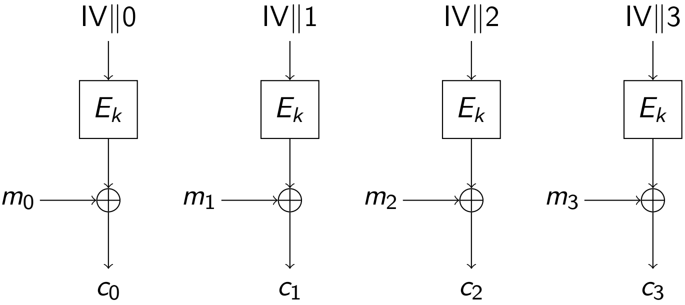
$$E_k : \{0,1\}^n \rightarrow \{0,1\}^n$$

A family of **permutations** indexed by a key (AES, 3DES, ...) where $n$ is the bit size of the permutation or block's size.

## Mode of operation

Describes how to use a **block cipher** along with a plaintext message of **arbitrary length** to achieve some concrete cryptographic goals.

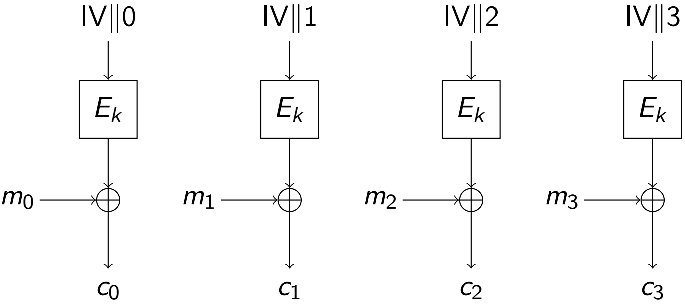
# The counter mode (CTR)




$m_i$ : The plaintext.
$E_k$ : The block cipher.
$c_i$ : The ciphertext.
IV : The Initialisation Value.

$$c_i = E_k(\mathrm{IV} \| i) \oplus m_i$$

# The counter mode (CTR)



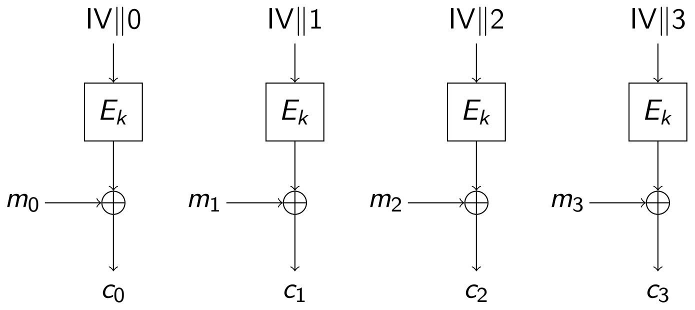

$m_i$ : The plaintext. $E_k$ : The block cipher.

$c_i$ : The ciphertext. IV : The Initialisation Value.

$$c_i = E_k(\mathrm{IV} \| i) \oplus m_i$$

Akin to a stream cipher: keystream XORed with the plaintext.

# The counter mode (CTR)




$m_i$ : The plaintext. $E_k$ : The block cipher.
$c_i$ : The ciphertext. IV : The Initialisation Value.

$$c_i = E_k(\mathrm{IV}\|i) \oplus m_i$$

Akin to a stream cipher: keystream XORed with the plaintext.
Inputs $\mathrm{IV}\|i$ to the block cipher never repeat.

# The counter mode (CTR)

Let $K_i = E_k(\mathrm{IV} \| i)$ the $i$th block of keystream.

- If $E_k$ is a good Pseudo-Random Function (PRF) then all $K_i$ are random and this is a one-time-pad.
- A block cipher is a Pseudo-Random Permutation (PRP) therefore $K_i$ are all distinct: $K_i \neq K_j \; \forall i \neq j$.

# The counter mode (CTR)

Let $K_i = E_k(\mathrm{IV}\|i)$ the $i$th block of keystream.

- If $E_k$ is a good Pseudo-Random Function (PRF) then all $K_i$ are random and this is a one-time-pad.
- A block cipher is a Pseudo-Random Permutation (PRP) therefore $K_i$ are all distinct: $K_i \neq K_j \; \forall i \neq j$.

**Security proof ($\sigma$ the number of blocks)**

$$\mathbf{Adv}^{\mathrm{IND}}_{\mathrm{CTR}\text{-}E_k}(\sigma) \leq \mathbf{Adv}^{\mathrm{PRF}}_{E_k}(\sigma) \leq \mathbf{Adv}^{\mathrm{PRP}}_{E_k}(\sigma) + \sigma^2/2^{n+1}$$

**Distinguisher**

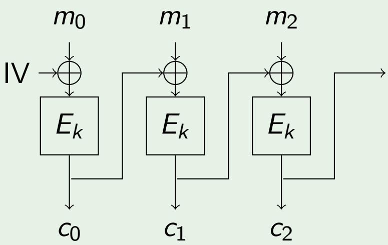
After $\sigma \simeq 2^{n/2}$ encrypted blocks we expect a collision on the $K_i$ with high probability in the case of a random ciphertext.
That is the birthday bound coming from the birthday paradox.

# CBC and CTR

**Both modes are:**

- widely deployed
- proven secure up to birthday bound ($2^{n/2}$)
- matching distinguishers at the proof's bound

## CBC mode

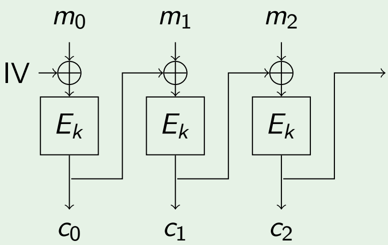


# CBC and CTR

**Both modes are:**

- widely deployed
- proven secure up to birthday bound ($2^{n/2}$)
- matching distinguishers at the proof's bound

## CBC mode




## Folklore assumptions [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.

# The counter mode (CTR)

From a distinguishing attack to a plaintext recovery attack ?

- If we know $m_i$, we recover $K_i = c_i \oplus m_i$.

# The counter mode (CTR)

From a distinguishing attack to a plaintext recovery attack ?

- If we know $m_i$, we recover $K_i = c_i \oplus m_i$.
- We can observe repeated encryptions of a secret $S$ that is $c_j = K_j \oplus S$ for many different $j$.

# The counter mode (CTR)

From a distinguishing attack to a plaintext recovery attack ?

- If we know $m_i$, we recover $K_i = c_i \oplus m_i$.
- We can observe repeated encryptions of a secret $S$ that is $c_j = K_j \oplus S$ for many different $j$.
- The distinguisher uses $K_i \oplus K_j \neq 0$ which implies $K_i \oplus c_j \neq S$ $\forall i \neq j$.

# The counter mode (CTR)

From a distinguishing attack to a plaintext recovery attack ?

- If we know $m_i$, we recover $K_i = c_i \oplus m_i$.
- We can observe repeated encryptions of a secret $S$ that is $c_j = K_j \oplus S$ for many different $j$.
- The distinguisher uses $K_i \oplus K_j \neq 0$ which implies $K_i \oplus c_j \neq S \; \forall i \neq j$.

## Main Idea

Collect many keystream blocks $K_i$ and encryptions of secret block $c_j = K_j \oplus S$; then look for a value $S$ such that $K_i \oplus c_j \neq S \; \forall i \neq j$.

# Missing difference problem

## The missing difference problem

- Given $\mathcal{A}$ and $\mathcal{B}$, and a hint $\mathcal{S}$ three sets of $n$-bit words
- Find $S \in \mathcal{S}$ such that:

$$\forall (a, b) \in \mathcal{A} \times \mathcal{B},\ S \neq a \oplus b\ .$$

# Missing difference problem

## Main Idea

Collect many keystream blocks $K_i \in \mathcal{A}$ and encryptions of secret block $c_j = K_j \oplus S \in \mathcal{B}$; then look for a value $S \in \mathcal{S}$ such that $\forall (a, b) \in \mathcal{A} \times \mathcal{B},\ S \neq a \oplus b$ .

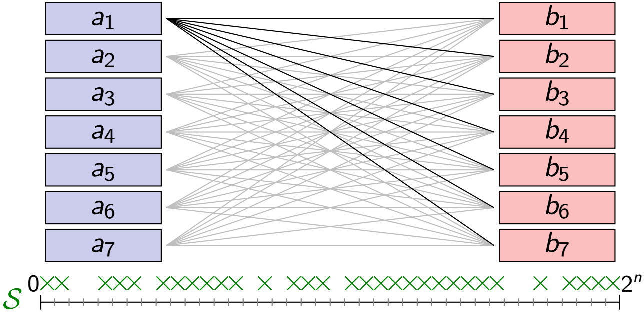
## The missing difference problem

- Given $\mathcal{A}$ and $\mathcal{B}$, and a hint $\mathcal{S}$ three sets of $n$-bit words
- Find $S \in \mathcal{S}$ such that:

$$\forall (a, b) \in \mathcal{A} \times \mathcal{B},\ S \neq a \oplus b \ .$$

# Simple Sieving Algorithm

[McGrew, FSE'13]

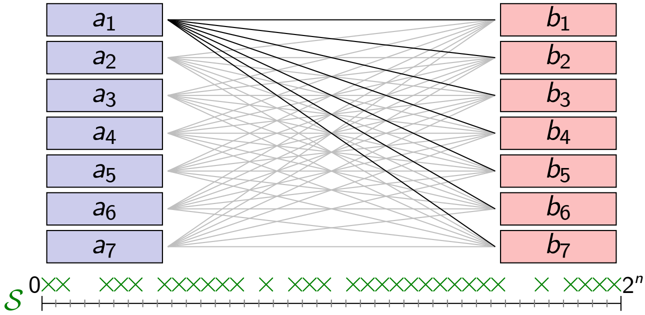



Compute all $a_i \oplus b_j$, remove results from a sieve $\mathcal{S}$.

**Analysis: case $|\mathcal{S}| = 2^n$ via coupon collector problem**

- To exclude $2^n$ candidates of $S$, 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: $\tilde{\mathcal{O}}(2^n)$

# Simple Sieving Algorithm [McGrew, FSE'13]



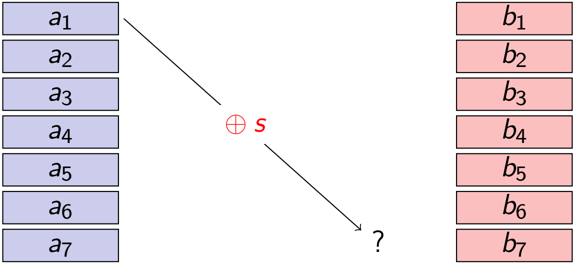

Compute all $a_i \oplus b_j$, remove results from a sieve $\mathcal{S}$.

**Analysis: case $|\mathcal{S}| = 2$**

- To exclude 1 candidate of $S$, we need $2^n$ values $a_i \oplus b_j$
  - Lists $\mathcal{A}$ and $\mathcal{B}$ of size $2^{n/2}$. Complexity: $\tilde{\mathcal{O}}(2^n)$

# Searching Algorithm

[McGrew, FSE'13]



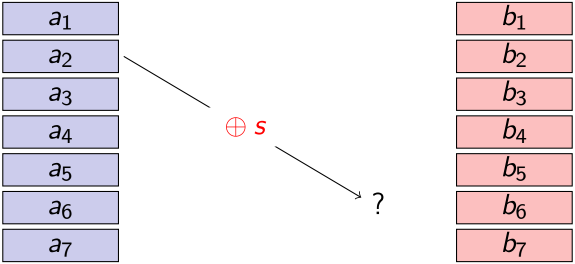

- Make a guess and verify.

**Try Guess ($s$)**

**for** $a$ **in** $\mathcal{A}$ **do**
  **if** $(s \oplus a) \in \mathcal{B}$ **then**
    **return** 0
**return** 1

# Searching Algorithm

[McGrew, FSE'13]



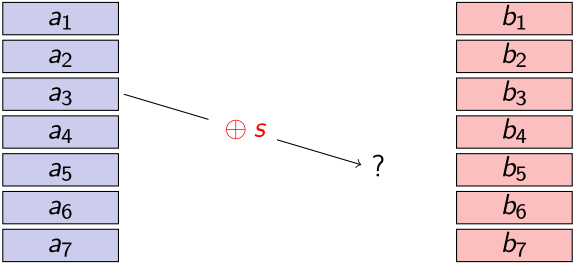

- Make a guess and verify.

**Try Guess ($s$)**

**for** $a$ **in** $\mathcal{A}$ **do**
  **if** $(s \oplus a) \in \mathcal{B}$ **then**
    **return** 0
**return** 1

# Searching Algorithm

[McGrew, FSE'13]



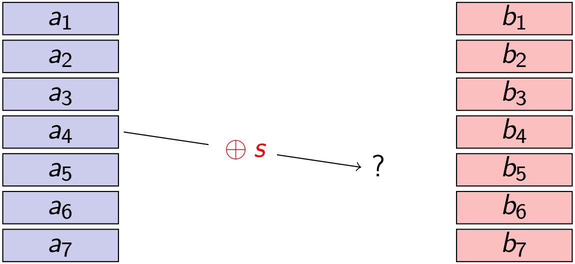

- Make a guess and verify.

**Try Guess ($s$)**

**for** $a$ **in** $\mathcal{A}$ **do**
  **if** $(s \oplus a) \in \mathcal{B}$ **then**
    **return** 0
**return** 1

# Searching Algorithm

[McGrew, FSE'13]

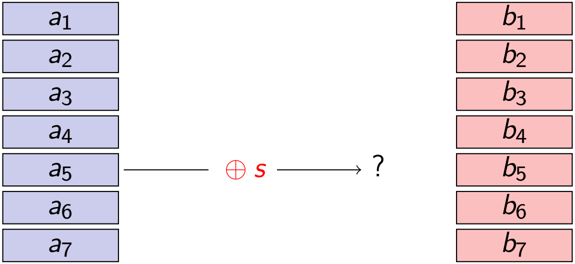



- Make a guess and verify.

**Try Guess ($s$)**

**for** $a$ **in** $\mathcal{A}$ **do**
  **if** $(s \oplus a) \in \mathcal{B}$ **then**
    **return** 0
**return** 1

# Searching Algorithm [McGrew, FSE'13]



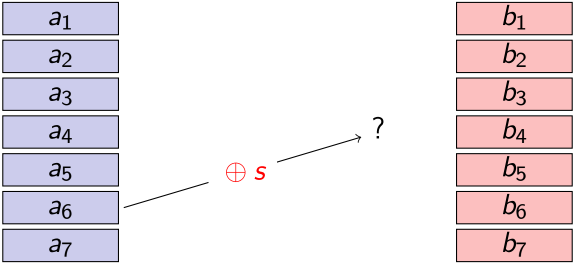

- Make a guess and verify.

**Try Guess ($s$)**

**for** $a$ **in** $\mathcal{A}$ **do**
  **if** $(s \oplus a) \in \mathcal{B}$ **then**
    **return** 0
**return** 1

# Searching Algorithm

[McGrew, FSE'13]

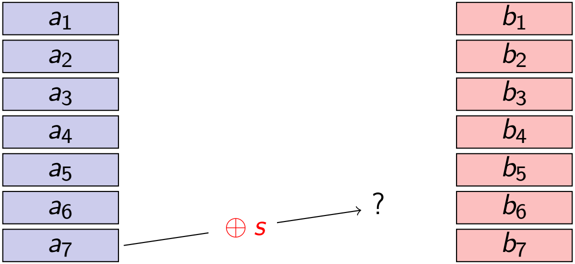



- Make a guess and verify.

**Try Guess $(s)$**

**for** $a$ **in** $\mathcal{A}$ **do**
  **if** $(s \oplus a) \in \mathcal{B}$ **then**
    **return** 0
**return** 1

# Searching Algorithm [McGrew, FSE'13]

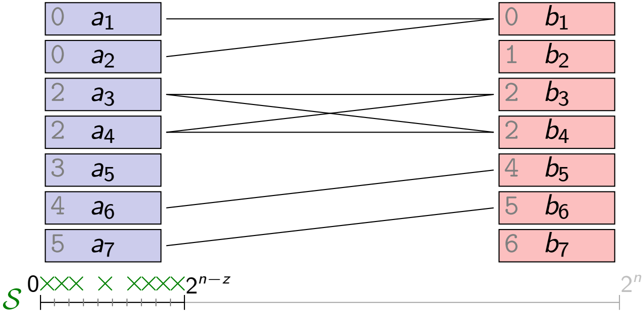



- Make a guess and verify.
- Complexity $\tilde{\mathcal{O}}(2^{n/2}\sqrt{|\mathcal{S}|})$ with unbalanced $\mathcal{A}$, $\mathcal{B}$.

**Try Guess ($s$)**

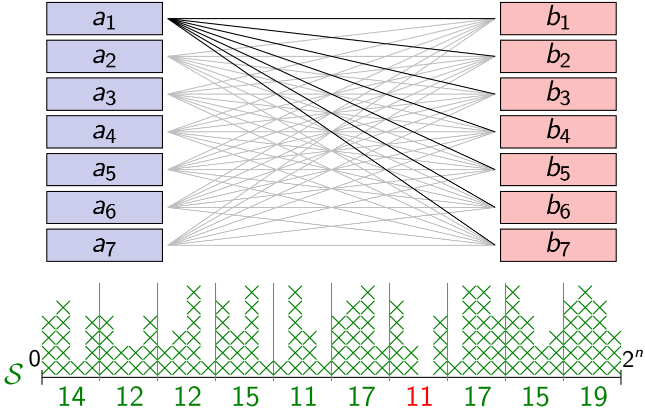
**for** $a$ **in** $\mathcal{A}$ **do**
  **if** $(s \oplus a) \in \mathcal{B}$ **then**
    **return** 0
**return** 1

# Known-prefix Sieving




- Assume $S$ starts with $z$ zero bits (more generally, linear subspace with $\dim\langle\mathcal{S}\rangle = n - z$)
- Sort lists, consider $a_i$'s and $b_j$'s with matching z-bit prefix
- Complexity: $\tilde{\mathcal{O}}(2^{n/2} + 2^{\dim\langle\mathcal{S}\rangle})$
  - Looking for collision + needed number of collisions
- Complexity: $\tilde{\mathcal{O}}(2^{n/2})$ when $\dim\langle\mathcal{S}\rangle \leq n/2$

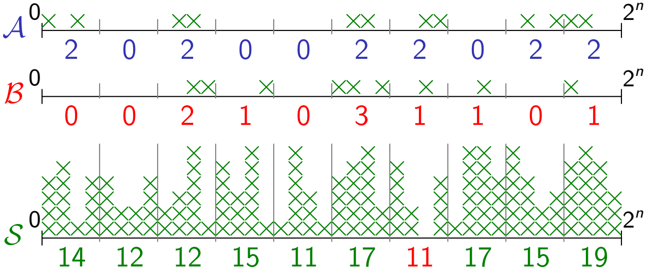
# Fast Convolution Sieving




- Instead of computing full sieve, use buckets (*ie.* truncate)
- With enough data, missing difference has smallest bucket with high probability

# Computing the sieve

- Count buckets for $\mathcal{A}$ and $\mathcal{B}$
  - $C_{\mathcal{X}}[i] = \left|\{x \in \mathcal{X} \mid T(x) = i\}\right|$



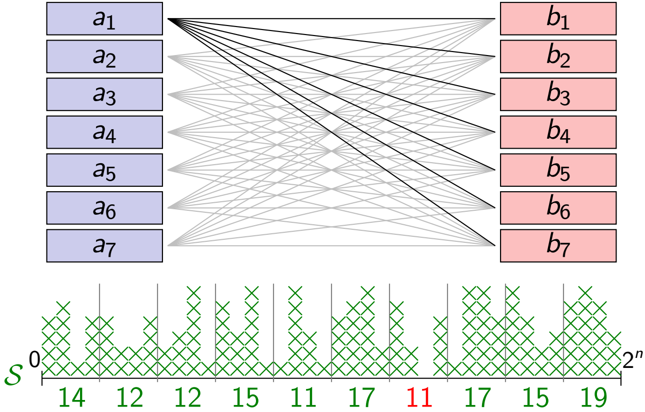
# Computing the sieve

- Count buckets for $\mathcal{A}$ and $\mathcal{B}$
  - $C_{\mathcal{X}}[i] = \left|\left\{x \in \mathcal{X} \mid T(x) = i\right\}\right|$
  - $$\begin{aligned} C_{\mathcal{S}}[i] &= |\{(a,b) \in \mathcal{A} \times \mathcal{B} \mid T(a \oplus b) = i\}| \\ &= \sum_{a \in \mathcal{A}} |\{b \in \mathcal{B} \mid T(a \oplus b) = i\}| \\ &= \sum_{a \in \mathcal{A}} C_{\mathcal{B}}[i \oplus T(a)] \\ &= \sum_{j \in \{0,1\}^{n-t}} C_{\mathcal{A}}[j] \cdot C_{\mathcal{B}}[i \oplus j] \end{aligned}$$

# Computing the sieve

- Count buckets for $\mathcal{A}$ and $\mathcal{B}$
  - $C_{\mathcal{X}}[i] = \left|\left\{x \in \mathcal{X} \mid T(x) = i\right\}\right|$
  - $$\begin{aligned} C_{\mathcal{S}}[i] &= |\{(a,b) \in \mathcal{A} \times \mathcal{B} \mid T(a \oplus b) = i\}| \\ &= \sum_{a \in \mathcal{A}} |\{b \in \mathcal{B} \mid T(a \oplus b) = i\}| \\ &= \sum_{a \in \mathcal{A}} C_{\mathcal{B}}[i \oplus T(a)] \\ &= \sum_{j \in \{0,1\}^{n-t}} C_{\mathcal{A}}[j] \cdot C_{\mathcal{B}}[i \oplus j] \end{aligned}$$
- Discrete convolution can be computed efficiently with the Fast Walsh-Hadamard transform!
  - Complexity: $\tilde{\mathcal{O}}(|C_{\mathcal{S}}|)$ for arbitrary $\mathcal{S}$

# Fast Convolution Sieving




$$T(S) \stackrel{?}{=} \operatorname{argmin} C_{\mathcal{S}}[i]$$

And we can finish with Known-prefix Sieving to recover the rest.

- $2^{2n/3}$ queries, sieving with $2^{2n/3}$ buckets of $2^{n/3}$ elements

# Missing difference problem algorithms

## Algorithms for the missing difference problem

| | | |
|---|---|---|
| Simple Sieving | Complexity $\tilde{\mathcal{O}}(2^n)$ | [McGrew] |
| Searching | Complexity $\tilde{\mathcal{O}}(2^{n/2}\sqrt{\lvert\mathcal{S}\rvert})$ | [McGrew] |
| Known-prefix Sieving | Complexity $\tilde{\mathcal{O}}(2^{n/2} + 2^{\dim\langle\mathcal{S}\rangle})$ | |
| Fast Convolution Sieving | Complexity $\tilde{\mathcal{O}}(2^{2n/3})$ | |

# Missing difference problem algorithms

## Algorithms for the missing difference problem

| | | |
|---|---|---|
| **Simple Sieving** | Complexity $\tilde{\mathcal{O}}(2^n)$ | [McGrew] |
| **Searching** | Complexity $\tilde{\mathcal{O}}(2^{n/2}\sqrt{\lvert\mathcal{S}\rvert})$ | [McGrew] |
| **Known-prefix Sieving** | Complexity $\tilde{\mathcal{O}}(2^{n/2} + 2^{\dim\langle\mathcal{S}\rangle})$ | |
| **Fast Convolution Sieving** | Complexity $\tilde{\mathcal{O}}(2^{2n/3})$ | |

- Improved algorithm if $\mathcal{S}$ is a linear subspace
  - In particular still near optimal when $\dim\langle\mathcal{S}\rangle = n/2$
- Improved algorithm for arbitrary $\mathcal{S}$ at the cost of data
  - First algorithm with complexity below $2^n$ in that case

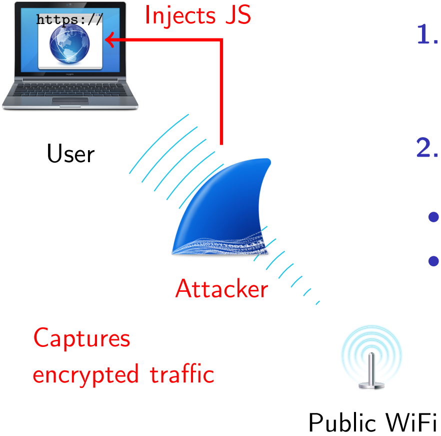
# Back to Cryptanalysis

**New Tools, New Attacks**

**Known-prefix** $\rightarrow$ plaintext recovery on CTR mode

**Fast Convolution** $\rightarrow$ forgery on GMAC and Poly1305

# BEAST Attack Setting [Duong & Rizzo 2011]



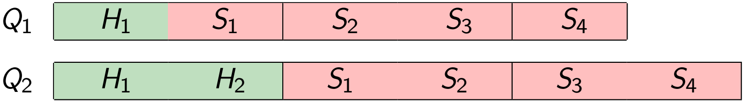

- Attacker has access to the network (*eg.* public WiFi)

1. Attacker uses JS to generate traffic
   - Tricks victim to malicious site
   - JS makes *cross-origin* requests
2. Attacker captures encrypted data

- Chosen plaintext attack
- Chosen-Prefix Secret-Suffix model $M \to \mathcal{E}(M \| S)$ [Hoang &al., Crypto'15]

# Application to CTR (CPSS queries)

- Plaintext recovery using the known-prefix sieving algorithm
- Two kind of queries; half-block and full-block headers:




1. Recover $S_1$ using the first block of each query:

$$\left.\begin{array}{l}\mathcal{A} = \{\mathcal{E}(H_1 \| H_2)\} \\ \mathcal{B} = \{\mathcal{E}(H_1 \| S_1)\}\end{array}\right\} \rightarrow \text{Missing difference:} \quad 0 \| (S_1 \oplus H_2).$$

# Application to CTR (CPSS queries)

- Plaintext recovery using the known-prefix sieving algorithm
- Two kind of queries; half-block and full-block headers:

| | | | | | | |
|---|---|---|---|---|---|---|
| $Q_1$ | $H_1$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | |
| $Q_2$ | $H_1$ | $H_2$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ |

1. Recover $S_1$ using the first block of each query:
   $$\left.\begin{array}{l}\mathcal{A} = \{\mathcal{E}(H_1 \| H_2)\} \\ \mathcal{B} = \{\mathcal{E}(H_1 \| S_1)\}\end{array}\right\} \rightarrow \text{Missing difference: } 0 \| (S_1 \oplus H_2).$$
2. When $S_1$ is known, recover $S_2$, with $Q_2$ queries:
   $$\left.\begin{array}{l}\mathcal{A} = \{\mathcal{E}(H_1 \| H_2)\} \\ \mathcal{B} = \{\mathcal{E}(S_1 \| S_2)\}\end{array}\right\} \rightarrow \text{Missing difference: } (S_1 \oplus H_1) \| (S_2 \oplus H_2).$$

# Application to CTR (CPSS queries)

- Plaintext recovery using the known-prefix sieving algorithm
- Two kind of queries; half-block and full-block headers:

| | | | | | | |
|---|---|---|---|---|---|---|
| $Q_1$ | $H_1$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ | |
| $Q_2$ | $H_1$ | $H_2$ | $S_1$ | $S_2$ | $S_3$ | $S_4$ |

1. Recover $S_1$ using the first block of each query:
   $$\left.\begin{array}{l}\mathcal{A} = \{\mathcal{E}(H_1 \| H_2)\} \\ \mathcal{B} = \{\mathcal{E}(H_1 \| S_1)\}\end{array}\right\} \to \text{Missing difference: } 0 \| (S_1 \oplus H_2).$$
2. When $S_1$ is known, recover $S_2$, with $Q_2$ queries:
   $$\left.\begin{array}{l}\mathcal{A} = \{\mathcal{E}(H_1 \| H_2)\} \\ \mathcal{B} = \{\mathcal{E}(S_1 \| S_2)\}\end{array}\right\} \to \text{Missing difference: } (S_1 \oplus H_1) \| (S_2 \oplus H_2).$$
3. When $S_2$ is known, recover $S_3$:
   $$\left.\begin{array}{l}\mathcal{A} = \{\mathcal{E}(H_1 \| H_2)\} \\ \mathcal{B} = \{\mathcal{E}(S_2 \| S_3)\}\end{array}\right\} \to \text{Missing difference: } (S_2 \oplus H_1) \| (S_3 \oplus H_2).$$
4. ...

# Application to CTR (CPSS queries)

## Full Asymptotic Complexity

| | |
|---|---|
| **Queries** | $\mathcal{O}(\sqrt{n} \cdot 2^{n/2})$ |
| **Memory** | $\mathcal{O}(\sqrt{n} \cdot 2^{n/2})$ |
| **Time** | $\mathcal{O}(n \cdot 2^{n/2})$ |

# Impacts

**How practical** can be the plaintext recovery attack on CTR ?

- Mostly used with AES, famous 128-bit block cipher, as part of GCM. 90% of Firefox HTTPS traffic uses AES-GCM.
  - Requires $128 \times 2^{64}$ bits $=$ 256 exbibytes over one session
  - 2016 global IP traffic is 82.3 exbibytes per month [Cisco]

# Impacts

**How practical** can be the plaintext recovery attack on CTR ?

- Mostly used with AES, famous 128-bit block cipher, as part of GCM. 90% of Firefox HTTPS traffic uses AES-GCM.
  - Requires $128 \times 2^{64}$ bits = 256 exbibytes over one session
  - 2016 global IP traffic is 82.3 exbibytes per month [Cisco]
- SSHv2 includes CTR with 3DES, a 64-bit block cipher.
  - Requires $64 \times 2^{32}$ bits = 32 GiB
  - Quickly attainable with modern internet speed

# Impacts

**How practical** can be the plaintext recovery attack on CTR ?

- Mostly used with AES, famous 128-bit block cipher, as part of GCM. 90% of Firefox HTTPS traffic uses AES-GCM.
  - Requires $128 \times 2^{64}$ bits = 256 exbibytes over one session
  - 2016 global IP traffic is 82.3 exbibytes per month [Cisco]
- SSHv2 includes CTR with 3DES, a 64-bit block cipher.
  - Requires $64 \times 2^{32}$ bits = 32 GiB
  - Quickly attainable with modern internet speed

**Sweet32 attack by Bhargavan and Leurent**

Attack in the **BEAST setting with birthday bound complexity** already shown to be a threat over the web in recent work.
This is the **Sweet32** attack on CBC mode, more commonly used with 64-bit block ciphers.

# Wegman-Carter Authentication Modes

- Wegman-Carter: build a MAC from a universal hash function and a PRF

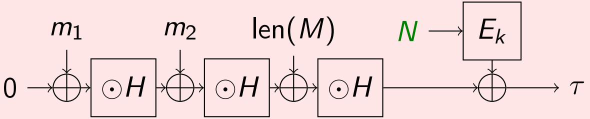
$$\mathrm{WC}(N, M) = H_{k_1}(M) \oplus F_{k_2}(N).$$

$$\mathbf{Adv}^{\mathrm{MAC}}_{\mathrm{WC}[H,F]} \leq \mathbf{Adv}^{\mathrm{PRF}}_{F} + \varepsilon + 2^{-n}$$

- Wegman-Carter-Shoup: use a block cipher as a PRF

$$\mathrm{WCS}(N, M) = H_{k_1}(M) \oplus E_{k_2}(N),$$

**Example: Polynomial-based hashing (GMAC, Poly1305-AES)**



# Key recovery as a missing difference problem

- Fix two messages $M \neq M'$, capture MACs
  - $a_i = \mathrm{MAC}(i, M) = H_{K_1}(M) \oplus K_i$
  - $b_j = \mathrm{MAC}(j, M') = H_{K_1}(M') \oplus K_j$
  - $a_i \oplus b_j \neq H_{K_1}(M) \oplus H_{K_1}(M')$
- For polynomial hashing, easy to recover universal hash key from $H_{K_1}(M) \oplus H_{K_1}(M')$

# Key recovery as a missing difference problem

- Fix two messages $M \neq M'$, capture MACs
  - $a_i = \mathrm{MAC}(i, M) = H_{K_1}(M) \oplus K_i$
  - $b_j = \mathrm{MAC}(j, M') = H_{K_1}(M') \oplus K_j$
  - $a_i \oplus b_j \neq H_{K_1}(M) \oplus H_{K_1}(M')$
- For polynomial hashing, easy to recover universal hash key from $H_{K_1}(M) \oplus H_{K_1}(M')$
- Sieving algorithm recovers $H(M) \oplus H(M')$ with $\tilde{\mathcal{O}}(2^{n/2})$ queries and $\tilde{\mathcal{O}}(2^n)$ computations
  - Independently done in another Eurocrypt paper!

Optimal Forgeries Against Polynomial-Based MACs and GCM
Atul Luykx, Bart Preneel [Eurocrypt '18]

# Key recovery as a missing difference problem

- Fix two messages $M \neq M'$, capture MACs
  - $a_i = \mathrm{MAC}(i, M) = H_{K_1}(M) \oplus K_i$
  - $b_j = \mathrm{MAC}(j, M') = H_{K_1}(M') \oplus K_j$
  - $a_i \oplus b_j \neq H_{K_1}(M) \oplus H_{K_1}(M')$
- For polynomial hashing, easy to recover universal hash key from $H_{K_1}(M) \oplus H_{K_1}(M')$
- Sieving algorithm recovers $H(M) \oplus H(M')$ with $\tilde{\mathcal{O}}(2^{n/2})$ queries and $\tilde{\mathcal{O}}(2^n)$ computations
  - Independently done in another Eurocrypt paper!

    Optimal Forgeries Against Polynomial-Based MACs and GCM
    Atul Luykx, Bart Preneel [Eurocrypt '18]
- Fast convolution sieving recovers $H(M) \oplus H(M')$ with $\tilde{\mathcal{O}}(2^{2n/3})$ queries and computations
  - First universal forgery attack with less than $2^n$ operations

# Bonus algorithm

**Citation** [Luykx & Preneel, Eurocrypt'18]

... implementing the attacks seems to require a large amount of storage to achieve significant success probability. It is unclear whether there is a compact way of representing the set of false keys.

# Bonus algorithm

**Citation** [Luykx & Preneel, Eurocrypt'18]

... implementing the attacks seems to require a large amount of storage to achieve significant success probability. It is unclear whether there is a compact way of representing the set of false keys.

**Optimal queries and memory complete sieving**

**Guess** first half of difference.

  **Run** Known-prefix sieving over second half.

**Repeat** until found.

# Bonus algorithm

**Citation** [Luykx & Preneel, Eurocrypt'18]

... implementing the attacks seems to require a large amount of storage to achieve significant success probability. It is unclear whether there is a compact way of representing the set of false keys.

**Optimal queries and memory complete sieving**

**Guess** first half of difference.

**Run** Known-prefix sieving over second half.

**Repeat** until found.

Time is still $\tilde{\mathcal{O}}(2^n)$ but memory reduced to $\mathcal{O}(2^{n/2})$ in the nonce-respecting CPA model.

# Conclusion

We defined the **missing difference problem** and **improved** the algorithms to solve it in particular for some cases:

| Case | Previous | This work | Improved attacks |
|---|---|---|---|
| $\mathcal{S}$ affine subspace of dim $n/2$ | $\tilde{\mathcal{O}}(2^{3n/4})$ | $\tilde{\mathcal{O}}(2^{n/2})$ | CTR plaintext recovery |
| No prior info *ie.* $\lvert\mathcal{S}\rvert = 2^n$ | $\tilde{\mathcal{O}}(2^{n})$ | $\tilde{\mathcal{O}}(2^{2n/3})$ | GMAC, Poly1305 universal forgery |

# Conclusion

We defined the **missing difference problem** and **improved** the algorithms to solve it in particular for some cases:

| Case | Previous | This work | Improved attacks |
| --- | --- | --- | --- |
| $\mathcal{S}$ affine subspace of dim $n/2$ | $\tilde{\mathcal{O}}(2^{3n/4})$ | $\tilde{\mathcal{O}}(2^{n/2})$ | CTR plaintext recovery |
| No prior info *ie.* $\lvert\mathcal{S}\rvert = 2^n$ | $\tilde{\mathcal{O}}(2^n)$ | $\tilde{\mathcal{O}}(2^{2n/3})$ | GMAC, Poly1305 universal forgery |

Main take away :

- CTR mode not more secure than CBC (Sweet32).
- Frequent rekeying away from birthday bound will prevent these attacks.

# Known-prefix Sieving Simulation

We challenge the heuristic assumptions we made (independence of the XORs $\{a \oplus b\}$). Approximations seem good enough.

Ran simulations with $n = 64$ bits and $z = n/2 = 32$ zeros.

- Each round we compare two lists of $2^{n/2}$ elements.
- Each round we expect $2^{n/2}$ partial collisions.
- Coupon collector predicts $n/2 \cdot \ln(2) \cdot 2^{n/2}$ partial collisions to recover $S$, that is 23 rounds on expectation.
- Simulation gives an idea of what is hidden in the $\mathcal{O}$ notations.

**Consistent speed of leaking**

In every runs, after 16 rounds the sieve was left between 419 and 560 candidates of $S$ only.

# Known-prefix Sieving Simulation

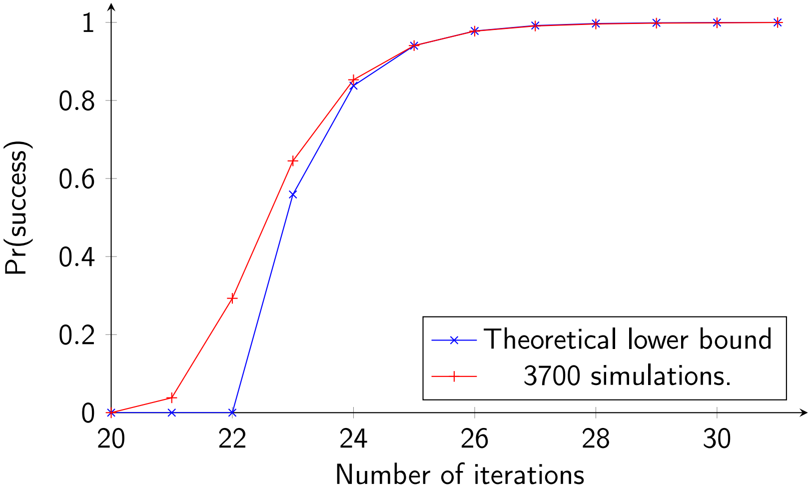
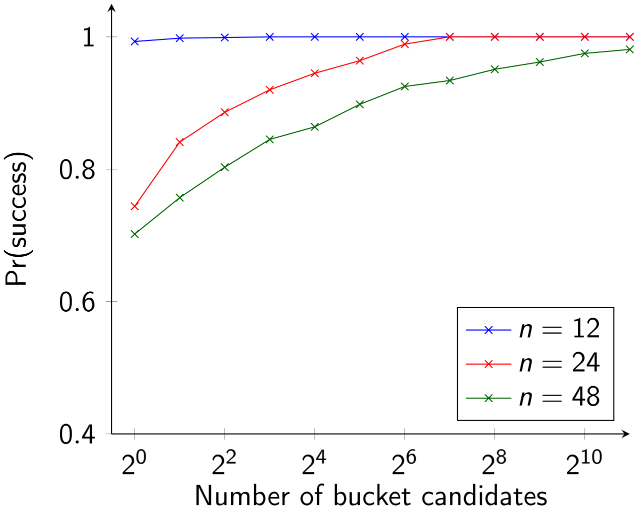



**Figure:** Probability of success of the known prefix sieving knowing $2^{32}$ encryptions of a 32-bit secret against the number of chunks of $2^{32}$ keystream blocks of size $n = 64$ bits used.

# Fast Convolution Simulation

**Figure:** Results for $\sqrt{n}2^{2n/3}$ data; counting over $2n/3$ bits.



# Works comparison

We independently described roughly the same attack on GCM, yet luckily our works complete each others:

**Leurent & Sibleyras, EC'18**

- Computational model
- Focus on **algorithms**
- Run simulations
- Provide a range of novel techniques and trade-offs
- Approach extendable to forgery on CWC mode

**Luykx & Preneel, EC'18**

- Information theoretic model
- Focus on **proofs**
- More rigorous analysis
- Show optimality w.r.t the best proofs
- Approach extendable to the KPA setting