Speeding up Preimage and Key-recovery Attacks with Highly Biased Differential-linear Approximations

Zhongfeng Niu

School of Cryptology, University of Chinese Academy of Sciences

Speeding up Preimage and Key-recovery Attacks with Highly Biased Differential-linear Approximations

Zhongfeng Niu

School of Cryptology, University of Chinese Academy of Sciences

Jointed Work with

Kai Hu, Siwei Sun, Zhiyu Zhang, Meiqin Wang

Outline

- Background and Motivation
- Preliminaries
- Speeding up Preimage Attacks
- Speeding up Key-recovery Attacks
- Applications

Outline

- **Background and Motivation**
- **Preliminaries**
- **Speeding up Preimage Attacks**
- **Speeding up Key-recovery Attacks**
- **Applications**

Background and Motivation

- **Searching for preimages and secret keys are central topics in symmetric-key cryptanalysis**

- **Techniques for permutation-based primitives are limited**
  - Sponge-based hash functions and XoFs: SHA3, ASCON-HASH/XOF, Esch/XOEsch (NIST LWC finalist Sparkle suite)
  - Sponge-based AEADs: ASCON, Schwaemm (ARX constructions)

The NIST LWC report 2023

"*All of these attacks on* `Schwaemm` *variants require data beyond the data limit made by the submitters … There is no known cryptanalysis on the hash variants …*" (see [TMC+23, Page 34])

- [TMC+23] Meltem Sönmez Turan et al. *Status report on the final round of the NIST lightweight cryptography standardization process*. 2023..

Background and Motivation

- **We are good at finding distinguishers**
- **Differential-linear distinguishers are often very effective**
  - Ascon permutation: 3- and 4-round practical DL distinguishers
  - New development in DL cryptanalysis of ARX ciphers
  - Alzette: 4-round deterministic DL distinguishers
  - Sparkle permutation (the underlying permutation of Schwaemm)

- **Can we use highly biased D-L distinguishers to speed up preimage and key-recovery attack?**

Background and Motivation

- **Recall the Complementary property of DES**

$$\mathrm{DES}_{k}(m) \oplus 111\ldots111 = \mathrm{DES}_{k\oplus 111\ldots111}(m \oplus 111\ldots111)$$

$$\overline{\mathrm{DES}_k(m)} = \mathrm{DES}_{\overline{k}}(\overline{m}).$$

- **Known** (m, c) **and** $(m \oplus 111\ldots111, c^*)$**, how to speed up the search of the key by a factor of 2?**
  - **Testing k' by computing** $c' = \mathrm{DES}_{k'}(m)$, **and if** $c = c'$ **we are done**
  - **Otherwise: test whether** $c' = \overline{c^*}$

- **Note that in the above process no oracle query with related-keys are made.**

Outline

- Background and Motivation
- **Preliminaries**
- Speeding up Preimage Attacks
- Speeding up Key-recovery Attacks
- Applications

Preliminaries

- **The *x*-translate:** $x \oplus \mathbb{A}$ **is defined as** $\{x \oplus y : y \in \mathbb{A}\}$ **, where** $x \in \mathbb{F}_2^n$ **and** $\mathbb{A} \subseteq \mathbb{F}_2^n$

- **The algebraic complementary space of a linear space** $\mathbb{V} \subseteq \mathbb{F}_2^n$

Let $\mathbb{V} \subseteq \mathbb{F}_2^n$ **be a linear space with** $\dim(\mathbb{V}) = d$ **spanned by** $\{\alpha_0, \cdots, \alpha_{d-1}\}$

Let $\mathbb{V}^{\dashv} \subseteq \mathbb{F}_2^n$ **be a linear space with** $\dim(\mathbb{V}^{\dashv}) = n - d$ **spanned by** $\{\beta_0, \cdots, \beta_{n-d-1}\}$

$\{\alpha_0, \cdots, \alpha_{d-1}, \beta_0, \cdots, \beta_{n-d-1}\}$ **are linearly independent**

Preliminaries

Lemma 1. *Let $\mathbb{V} \subseteq \mathbb{F}_2^n$ be a linear space. Then $\bigcup_{x \in \mathbb{V}^{\dashv}} x \oplus \mathbb{V} = \mathbb{F}_2^n$. Moreover, For $x, y \in \mathbb{V}^{\dashv}$, $x \oplus \mathbb{V} \cap y \oplus \mathbb{V} \neq \emptyset$ if and only if $x = y$. That is, the $2^{n-\dim(\mathbb{V})}$ subsets $x \oplus \mathbb{V}$ with $x \in \mathbb{V}^{\dashv}$ form a partition of $\mathbb{F}_2^n$.*

Remark 1. For a linear space $\mathbb{V} \subseteq \mathbb{F}_2^n$, $\mathbb{V}^{\dashv}$ is not always equal to $\mathbb{V}^{\perp} = \{x \in \mathbb{F}_2^n : x \cdot y = 0 \text{ for all } y \in \mathbb{V}\}$. For example, if $\mathbb{V} = \{\mathtt{00}, \mathtt{11}\} \in \mathbb{F}_2^2$, then a choice of $\mathbb{V}^{\dashv}$ is $\{\mathtt{00}, \mathtt{01}\}$, and $\mathbb{V}^{\perp} = \{\mathtt{00}, \mathtt{11}\}$. But if $\mathbb{V}$ is spanned by unit vectors, we always have $\mathbb{V}^{\dashv} = \mathbb{V}^{\perp}$.

Preliminaries: The Correlation of Differential-linear Approximations

- **Differential-linear cryptanalysis （Langford and Hellman in 1994）**
- $f : \mathbb{F}_2^m \rightarrow \mathbb{F}_2^n$ **. The correlation** ϵ **is defined as**

$$\epsilon = \frac{1}{2^m} \sum_{x \in \mathbb{F}_2^m} (-1)^{\lambda \cdot (f(x) \oplus f(x \oplus \delta))}$$

- **Thus we have**

$$\Pr[\lambda \cdot (f(x) \oplus f(x \oplus \delta)) = 0] = \frac{1}{2} + \frac{\epsilon}{2}$$

- **When** $\epsilon \neq 0,\ \lambda \cdot (f(x) \oplus f(x \oplus \delta))$ **is biased towards**

$$\zeta_\epsilon = \frac{(-1)^{\mathtt{Sign}(\epsilon)} + 1}{2} \qquad Sign(x) = \begin{cases} 1, x > 0 \\ 0, x < 0 \end{cases}$$

Outline

- Background and Motivation
- Preliminaries
- **Speeding up Preimage Attacks**
- Speeding up Key-recovery Attacks
- Applications

Speeding up Preimage Attacks: The Naïve Case and the Basic Idea

$F : \mathbb{F}_2^m \to \mathbb{F}_2^n$ **with a deterministic DL approximation** $\lambda \cdot (F(x) \oplus F(x \oplus \delta)) = 0$

$$\boxed{F(?) = 0}$$

How to check 2 "guessed" preimages with about 1.5 evaluations of *F*?

- **Guess x and compute $y = F(x)$, if $y = O$ we are done**
- **Otherwise, test whether $\lambda \cdot (O \oplus y) = 0$, if $\lambda \cdot (O \oplus y) \neq 0$,**

  $x \oplus \delta$ can not be a preimage.

Speeding up Preimage Attacks: The Algorithmic Framework

Algorithm 1: Speed up the preimage search with DL distinguishers

Input: $O \in \mathbb{F}_2^n$; The sets of input differences $\mathbb{D} = \{\delta_0, \ldots, \delta_{s-1}\}$ and linear masks $\mathbb{M}_i = \{\lambda_{i,0}, \ldots, \lambda_{i,\ell_i-1}\}$ for $0 \leq i < s$ such that $(\delta_i, \lambda_{i,j})$ is a differential-linear approximation of F with correlation $\mathfrak{c}_{i,j}$

Output: A preimage x such that $F(x) = O$ or $\perp$

$cnt \leftarrow 0$
while $cnt < N$ **do**
  Randomly generate an input $x \in \mathbb{F}_2^m$
  $cnt \leftarrow cnt + 1$
  $y \leftarrow F(x)$
  if $y = O$ **then**
    return x ▷ x is a preimage of O
  for $0 \leq i < s$ **do**
    $reject \leftarrow$ PreTest$(y, O, \delta_i, \mathbb{M}_i)$ ▷ Perform some statistical test
    if $reject = 0$ **then**
      $y' \leftarrow F(x \oplus \delta_i)$
      if $y' = O$ **then**
        return $x \oplus \delta_i$
return $\perp$

Algorithm 2: Implement PreTest() with the strictest strategy

Input: $y = F(x)$ for some $x \in \mathbb{F}_2^m$, the preimage O, $\delta_i \in \mathbb{D}$, linear masks $\mathbb{M}_i = \{\lambda_{i,0}, \ldots, \lambda_{i,\ell_i-1}\}$ such that $(\delta_i, \lambda_{i,j})$ is a differential-linear approximation of F with correlation $\mathfrak{c}_{i,j}$

Output: 0 or 1

for $0 \leq j < \ell_i$ **do**
  if $\lambda_{i,j} \cdot (y \oplus O) \neq \zeta_{\mathfrak{c}_{i,j}}$ **then**
    return 1
return 0

We require that the masks in M_i are *linearly independent*.

Speeding up Preimage Attacks: Complexity analysis

Elements in $\{x \oplus \delta_i : \delta_i \in \mathbb{D}, \exists j \in [\ell_i], \text{ such that } \lambda_{i,j} \cdot (y \oplus O) \neq \zeta_{\epsilon_{i,j}}\}$ **are skipped !**

Elements in $\mathbb{S}_{x,\mathbb{D}} = \{x \oplus \delta_i : \delta_i \in \mathbb{D}, \lambda_{i,j} \cdot (y \oplus O) = \zeta_{\epsilon_{i,j}}, 0 \leq j < \ell_i\}$ **are evaluated !**

The time complexity is about $N\left(1 + \sum_{i=0}^{s-1} 2^{-\ell_i}\right)$ **evaluations of F**

When the complexity of the testing procedure is not negligible compared with the evaluations of F, we can use hash tables to deal with it (see later slides).

Speeding up Preimage Attacks: Success Probability

- The successful probability is lower bounded by $\mathrm{P}_{suc} = 1-(1-\rho\cdot\tau)^N$, where

$$\tau = 2^{\log(s+1)-n} \qquad \text{and} \qquad \rho = \frac{1}{s+1}\left(1 + \sum_{i=0}^{s-1} p_i\right)$$

$$p_i = \prod_{j=0}^{\ell_i-1}\left(\frac{1}{2} + \frac{|\mathfrak{c}_{i,j}|}{2}\right)$$

- We always set $N = (\rho\cdot\tau)^{-1}$ to make the success probability to be about

$$1 - e^{-1} \approx 0.63.$$

- Note that for a random hash function with n-bit output the success probability after we try 2^n random elements is

$$1 - \left(1 - \frac{1}{2^n}\right)^{2^n} \approx 1 - e^{-1} \approx 0.63$$

Outline

- Background and Motivation
- Preliminaries
- Speeding up Preimage Attacks
- **Speeding up Key-recovery Attacks**
- Applications

Speeding up Key-recovery Attacks: The Naïve Case and the Basic Idea

$F : \mathbb{F}_2^m \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ **is a keyed function with** $F(K, P) = C$ **and** K **being the secret key**

$F_p(\cdot) = F(\cdot, P)$ **can be regarded as a one-way function parameterized with *P***

$\lambda \cdot (F(K, P) \oplus F(K \oplus \delta, P \oplus \delta')) = 0$ **be a *deterministic* related key D-L approximation**

How to check 2 "guessed" keys with about 1.5 evaluations of *F*?

- **Make single-key queries once:** $C = F(K, P)$ and $C' = F(K, P \oplus \delta')$
- **Guess** $K = k$**, compute** $Y = F(k, P)$**, and if** $Y = C$**, done.**
- **Otherwise check whether** $\lambda \cdot (C \oplus Y) = 0$**, and if it holds** $k \oplus \delta$ **is a candidate.**

Speeding up Key-recovery Attacks: The Algorithmic Framework

$F : \mathbb{F}_2^m \times \mathbb{F}_2^n \to \mathbb{F}_2^n$

Key space

Input: $\mathbb{D} = \{(\delta_0, \delta'_0), \cdots, (\delta_{s-1}, \delta'_{s-1})\} \subseteq \mathbb{F}_2^{m+n}$, and $\mathbb{M}_i = \{\lambda_{i,0}, \ldots, \lambda_{i,\ell_i - 1}\}$ for $0 \le i < s$ such that $((\delta_i, \delta'_i), \lambda_{i,j})$ is a *related-key* DL appoximation of F with correlation $\epsilon_{i,j}$, and $\hat{\mathbb{D}}_K = \{0\} \cup \{\delta_0, \cdots, \delta_{s-1}\}$ is a linear subspace of $\mathbb{F}_2^m$.

Output: The master key K

Randomly choose a plaintext P, derive $C = F(K, P)$

for $0 \le i < s$ **do**

> $C_i = F(K, P \oplus \delta'_i)$

for $k \in \hat{\mathbb{D}}_K^{\perp}$ **do**

> $c \leftarrow F(k, P)$
>
> **if** $c = C$ **then**
>
> > **if** $F(k, P \oplus \delta'_i) = C_i, 0 \le i < s$ **then**
> >
> > > **return** k $\quad \triangleright$ `a few of` $(P \oplus \delta'_i, C_i)$ `suffice`
>
> **for** $0 \le i < s$ **do**
>
> > $flag \leftarrow 0$
> >
> > **for** $0 \le j < \ell_i$ **do**
> >
> > > **if** $\lambda_{i,j} \cdot (c \oplus C_i) \neq \zeta_{\epsilon_{i,j}}$ **then**
> > >
> > > > $flag \leftarrow 1$ $\quad \triangleright$ $k \oplus \delta$ `fails to pass the filter`
> >
> > **if** $flag = 0$ **then**
> >
> > > **if** $F(k \oplus \delta_i, P \oplus \delta'_j) = C_j, 1 \le j < s$ **then**
> > >
> > > > **return** k $\quad \triangleright$ `a few of` $(P \oplus \delta'_i, C_i)$ `suffice`

Complexity analysis and Success Probability

Masks per Difference

- **The time complexity is about** $2^{m-\log(s+1)}\left(1+\sum_{i=0}^{s-1} 2^{-\ell_i}\right)$

Key size

Difference

- **Success probability** $\dfrac{1}{s+1}\left(1+\sum_{i=0}^{s-1} p_i\right)$, **where** $p_i = \prod_{j=0}^{\ell_i-1}\left(\frac{1}{2}+\frac{\mathfrak{c}_{i,j}}{2}\right)$

If we have too many D-L distinguishers, the complexity of the testing process is not negligible.

What if we have too many D-L distinguishers? A contrived example.

$$F : \mathbb{F}_2^{256} \times \mathbb{F}_2^{256} \to \mathbb{F}_2^{256} \qquad \lambda_j \cdot (F(k \oplus \delta_i, P \oplus \delta_i') \oplus F(k, P)) = 0$$

$$L = (\lambda_0, \cdots, \lambda_{127})^T$$

2^{128} times F and 2^{128} times L

Input: $D = \{0, \delta_0, \cdots, \delta_{2^{128}-2}\}$, $D' = \{\delta_0', \cdots, \delta_{2^{128}-2}'\}$, $M = \{\lambda_0, \cdots, \lambda_{127}\}$.
Output: The master key K

Randomly choose a plaintext P
$C \leftarrow F(K, P)$ // Query the oracle
for $0 \leq i < 2^{128} - 1$ **do**
  $C_{\delta_i} \leftarrow F(K, P \oplus \delta_i')$ // Query the oracle
  Insert δ_i into a hash table at address $L(C_{\delta_i})$
for $k \in D^{\dashv}$ **do**
  $c \leftarrow F(k, P)$
  if $c = C$ **then**
    return k
  Addr $\leftarrow L(c)$
  for δ *at address* Addr *of the hash table* **do**
    $C' \leftarrow F(k \oplus \delta, P)$
    if $C' = C$ **then**
      return $k \oplus \delta$
return $\perp$

In total, 3×2^{128} times F, 2×2^{128} times L, and 2^{128} times hash table lookups. Speedup factor 2^{126}.

What if we have too many D-L distinguishers? A contrived example.

$F : \mathbb{F}_2^{256} \times \mathbb{F}_2^{256} \to \mathbb{F}_2^{256} \qquad \lambda_j \cdot (F(k \oplus \delta_i, P \oplus \delta_i') \oplus F(k, P)) = 0$

$$L = (\lambda_0, \cdots, \lambda_{127})^T$$

2^{128} times F and 2^{128} times L

Dim = 128

Input: $D = \{0, \delta_0, \cdots, \delta_{2^{128}-2}\}$, $D' = \{\delta_0', \cdots, \delta_{2^{128}-2}'\}$, $M = \{\lambda_0, \cdots, \lambda_{127}\}$.
Output: The master key K

```
Randomly choose a plaintext P
C ← F(K, P)                                          // Query the oracle
for 0 ≤ i < 2^128 − 1 do
    C_{δ_i} ← F(K, P ⊕ δ'_i)                          // Query the oracle
    Insert δ_i into a hash table at address L(C_{δ_i})
for k ∈ D do
    c ← F(k, P)
    if c = C then
        return k
    Addr ← L(c)
    for δ at address Addr of the hash table do
        C' ← F(k ⊕ δ, P)
        if C' = C then
            return k ⊕ δ
return ⊥
```

In total, 3×2^{128} times F, 2×2^{128} times L, and 2^{128} times hash table lookups. Speedup factor 2^{126}.

What if we have too many D-L distinguishers? A contrived example.

$F : \mathbb{F}_2^{256} \times \mathbb{F}_2^{256} \to \mathbb{F}_2^{256}$ $\qquad \lambda_j \cdot (F(k \oplus \delta_i, P \oplus \delta_i') \oplus F(k, P)) = 0$

$$L = (\lambda_0, \cdots, \lambda_{127})^T$$

2^{128} times F and 2^{128} times L

Input: $D = \{0, \delta_0, \cdots, \delta_{2^{128}-2}\}$, $D' = \{\delta_0', \cdots, \delta_{2^{128}-2}'\}$, $M = \{\lambda_0, \cdots, \lambda_{127}\}$.
Output: The master key K

Randomly choose a plaintext P
$C \leftarrow F(K, P)$ // Query the oracle

Dim = 128

for $0 \le i < 2^{128} - 1$ **do**
  $C_{\delta_i} \leftarrow F(K, P \oplus \delta_i')$ // Query the oracle
  Insert δ_i into a hash table at address $L(C_{\delta_i})$
for $k \in \boxed{D^{\perp}}$ **do**
  $\boxed{c \leftarrow F(k, P)}$ 1 time / loop
  if $c = C$ **then**
    return k
  $\boxed{\texttt{Addr} \leftarrow L(c)}$ 1 time / loop
  for δ *at address* `Addr` *of the hash table* **do**
    $C' \leftarrow F(k \oplus \delta, P)$
    if $C' = C$ **then**
      return $k \oplus \delta$
return $\perp$

In total, 3×2^{128} times F, 2×2^{128} times L, and 2^{128} times hash table lookups. Speedup factor 2^{126}.

What if we have too many D-L distinguishers? A contrived example.

$F : \mathbb{F}_2^{256} \times \mathbb{F}_2^{256} \to \mathbb{F}_2^{256}$ $\qquad \lambda_j \cdot (F(k \oplus \delta_i, P \oplus \delta_i') \oplus F(k, P)) = 0$

$$L = (\lambda_0, \cdots, \lambda_{127})^T$$

2^{128} times F and 2^{128} times L

Dim = 128

Input: $D = \{0, \delta_0, \cdots, \delta_{2^{128}-2}\}$, $D' = \{\delta_0', \cdots, \delta_{2^{128}-2}'\}$, $M = \{\lambda_0, \cdots, \lambda_{127}\}$.
Output: The master key K

```
Randomly choose a plaintext P
C ← F(K, P)                                   // Query the oracle
for 0 ≤ i < 2^128 − 1 do
    C_{δ_i} ← F(K, P ⊕ δ'_i)                  // Query the oracle
    Insert δ_i into a hash table at address L(C_{δ_i})
for k ∈ D^⊥ do
    c ← F(k, P)
    if c = C then
        return k
    Addr ← L(c)
    for δ at address Addr of the hash table do
        C' ← F(k ⊕ δ, P)
        if C' = C then
            return k ⊕ δ
return ⊥
```

1 time / loop

1 time / loop

1 hash table lookup / loop

In total, 3×2^{128} times F, 2×2^{128} times L, and 2^{128} times hash table lookups. Speedup factor 2^{126}.

What if we have too many D-L distinguishers? A contrived example.

$$F : \mathbb{F}_2^{256} \times \mathbb{F}_2^{256} \to \mathbb{F}_2^{256} \qquad \lambda_j \cdot (F(k \oplus \delta_i, P \oplus \delta_i') \oplus F(k, P)) = 0$$

$$L = (\lambda_0, \cdots, \lambda_{127})^T$$

2^{128} times F and 2^{128} times L

Input: $D = \{0, \delta_0, \cdots, \delta_{2^{128}-2}\}$, $D' = \{\delta_0', \cdots, \delta_{2^{128}-2}'\}$, $M = \{\lambda_0, \cdots, \lambda_{127}\}$.
Output: The master key K

```
Randomly choose a plaintext P
C ← F(K, P)                                        // Query the oracle
for 0 ≤ i < 2^128 − 1 do
    C_δi ← F(K, P ⊕ δ'_i)                          // Query the oracle
    Insert δ_i into a hash table at address L(C_δi)
for k ∈ D^⊣ do
    c ← F(k, P)
    if c = C then
        return k
    Addr ← L(c)
    for δ at address Addr of the hash table do
        C' ← F(k ⊕ δ, P)
        if C' = C then
            return k ⊕ δ
return ⊥
```

Dim = 128

1 time / loop

1 time / loop

1 hash table lookup / loop

1 time / loop

In total, 3×2^{128} times F, 2×2^{128} times L, and 2^{128} times hash table lookups. Speedup factor 2^{126}.

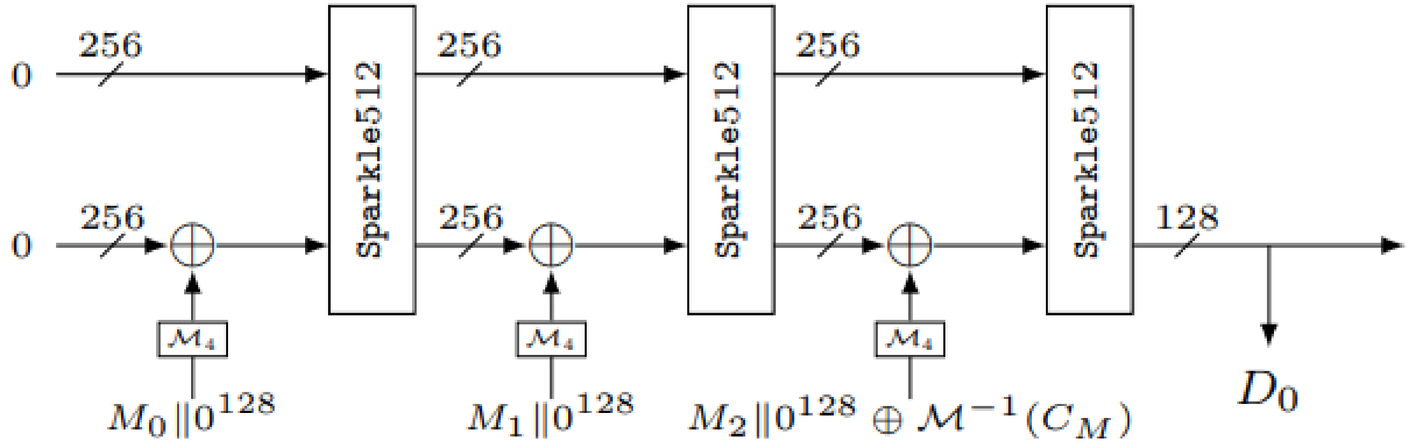
Outline

- Background and Motivation
- Preliminaries
- Speeding up Preimage Attacks
- Speeding up Key-recovery Attacks
- **Applications**

Application 1: Preimage Attacks on XOEsch (XoFs)

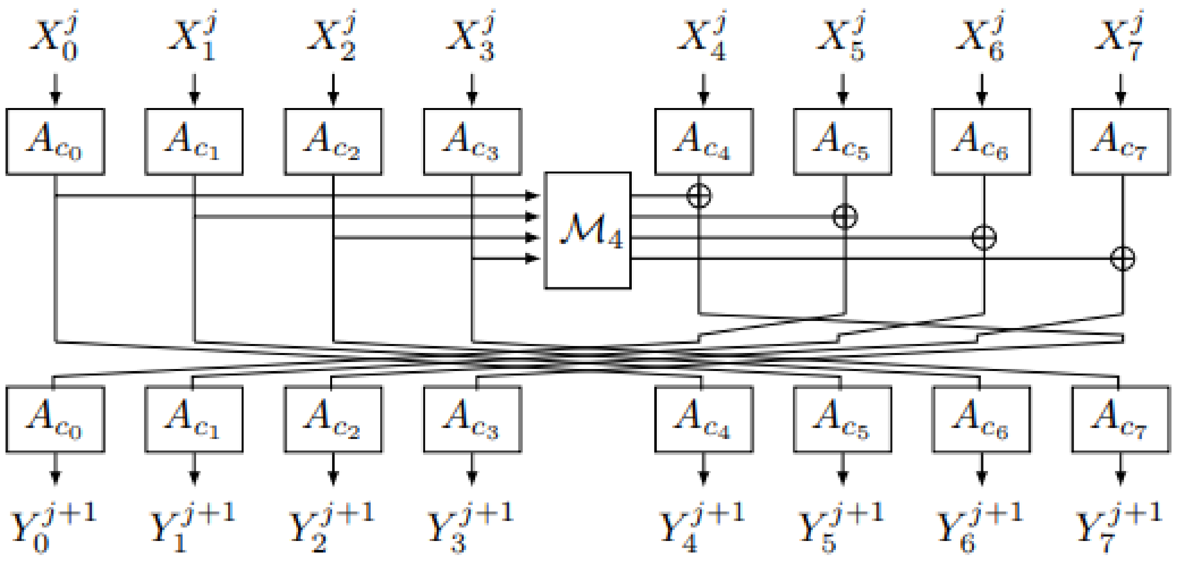
Table : Parameters used by XOEsch256 and XOEsch384 with the digest length being $t > 0$. Our attacks are applied to the cases with $t = 128$ and $t = 192$.

Instance	Size			Security Claim	
	Permutation	Rate	Capacity	Collision	(2nd) Preimage
XOEsch256	384	128	256	$\min\{128, t/2\}$	$\min\{128, t\}$
XOEsch384	512	128	384	$\min\{192, t/2\}$	$\min\{192, t\}$

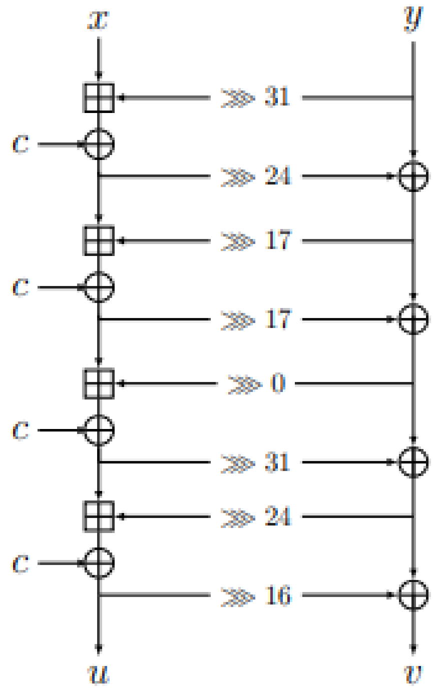


According to the specification of XOEsch, only when necessary, the message is padded.

Application 1: Preimage Attacks on XOEsch (XoFs)



(a) The structure of 1.5-step of `Sparkle512` permutation. In this instance, there are 8 64-bit branches.



(b) `Alzette` parameterized by c.

Differential-Linear Distinguishers for the Alzette Box

Table : The nontrivial DL Distinguishers of A_c with their *absolute* correlations. Note that these input differences consitute $\mathbb{D}_{\mathtt{Alzette}}$. Together with 0, they form a linear space denoted by $\hat{\mathbb{D}}_{\mathsf{Alzette}}$. All or the first five linear masks in the table head form $\mathbb{M}_i$ for each $\delta_i \in \mathbb{D}_{\mathtt{Alzette}}$.

Diff.\Mask	$(17,1)_\lambda$	$(18,2)_\lambda$	$(19,3)_\lambda$	$(5,21)_\lambda$	$(4,20)_\lambda$	$(14,30)_\lambda$	$(28,12)_\lambda$
$(0010)_\delta$	1	1	1	≥ 0.96	≥ 0.94	≥ 0.96	≥ 0.90
$(0100)_\delta$	1	1	1	≥ 0.96	≥ 0.94	≥ 0.94	≥ 0.86
$(1000)_\delta$	1	1	1	≥ 0.96	≥ 0.92	≥ 0.92	≥ 0.82
$(0110)_\delta$	1	1	1	≥ 0.96	≥ 0.94	≥ 0.94	≥ 0.88
$(1010)_\delta$	1	1	1	≥ 0.96	≥ 0.94	≥ 0.94	≥ 0.84
$(1100)_\delta$	1	1	1	≥ 0.96	≥ 0.94	≥ 0.94	≥ 0.86
$(1110)_\delta$	1	1	1	≥ 0.96	≥ 0.94	≥ 0.94	≥ 0.88
$(0001)_\delta$	≥ 0.92	≥ 0.92	≥ 0.94	≥ 0.916	≥ 0.84		
$(0011)_\delta$	≥ 0.92	≥ 0.92	≥ 0.94	≥ 0.92	≥ 0.84		
$(0101)_\delta$	≥ 0.92	≥ 0.92	≥ 0.94	≥ 0.92	≥ 0.84		
$(0111)_\delta$	≥ 0.92	≥ 0.92	≥ 0.94	≥ 0.92	≥ 0.84		
$(1011)_\delta$	≥ 0.92	≥ 0.92	≥ 0.94	≥ 0.92	≥ 0.84		
$(1101)_\delta$	≥ 0.92	≥ 0.92	≥ 0.94	≥ 0.92	≥ 0.84		
$(1111)_\delta$	≥ 0.92	≥ 0.92	≥ 0.94	≥ 0.92	≥ 0.84		
$(1001)_\delta$	≥ 0.92	≥ 0.92	≥ 0.94	≥ 0.92	≥ 0.86		

$$b_0 = (\mathtt{0x80000000}, \mathtt{0x0})$$

$$b_1 = (\mathtt{0x40000000}, \mathtt{0x0})$$

$$b_2 = (\mathtt{0x20000000}, \mathtt{0x0})$$

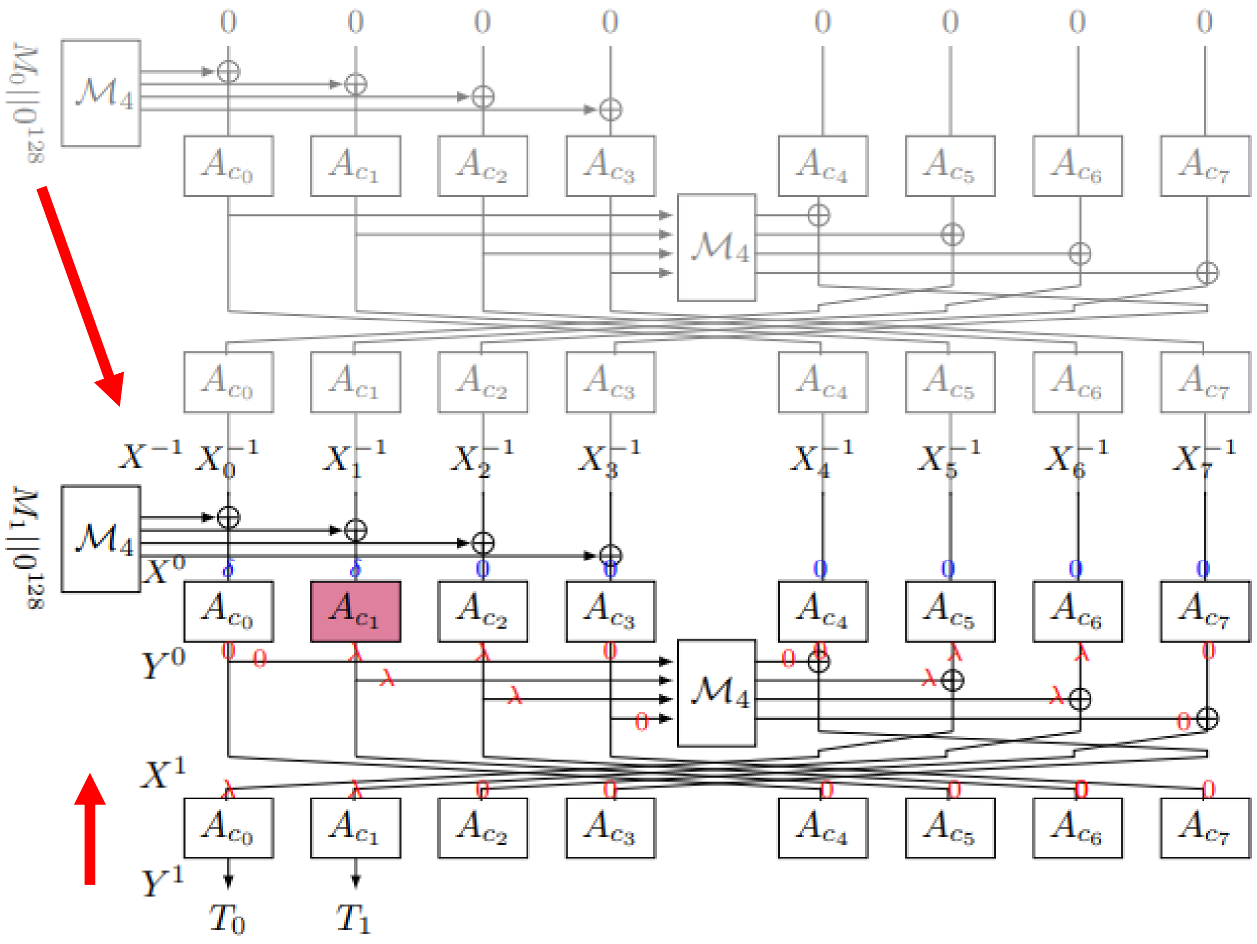
$$b_3 = (\mathtt{0x0}, \mathtt{0x40000000})$$

The difference space is spanned by unit vectors.

$$(\mathtt{0xa0000000}, \mathtt{0x40000000}) = b_0 \oplus b_1 \oplus b_3.$$

$$(1101)_\delta$$

Preimage Attack on 1.5-Step XOEsch384 with 128-bit Digest



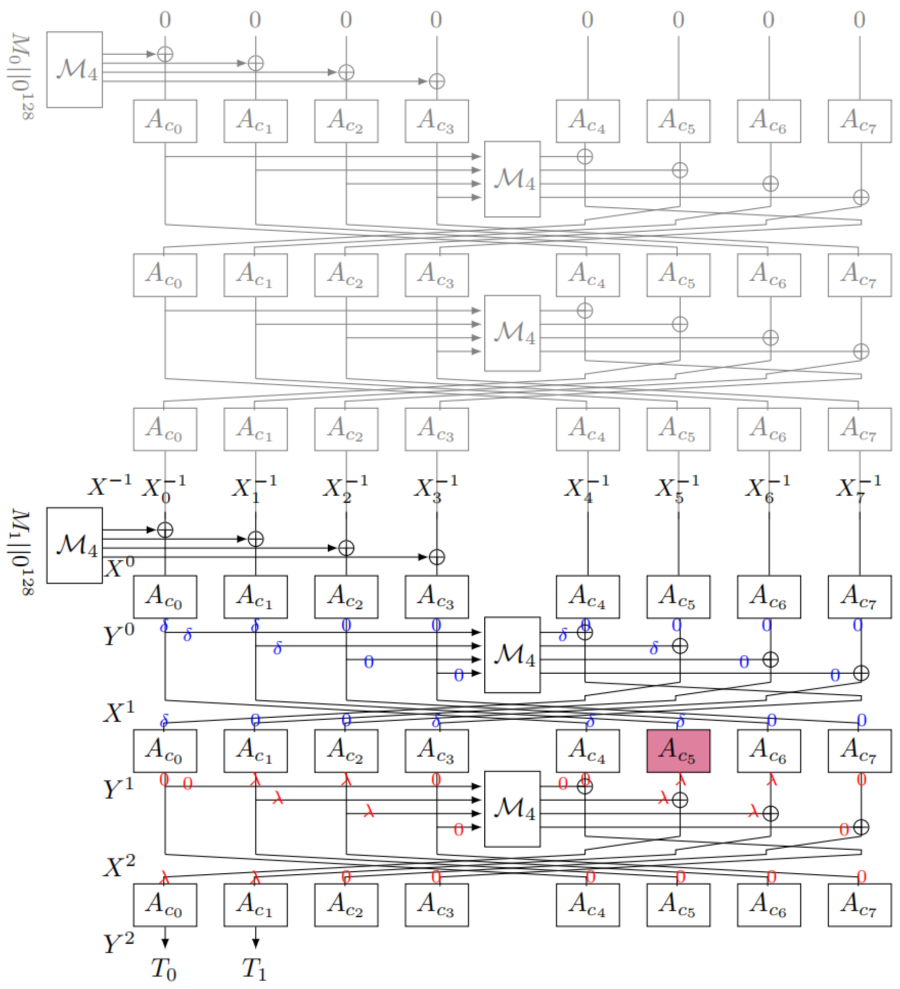
$F_{LSM}: \mathbb{F}_2^{128} \to \mathbb{F}_2^{128}$ mapping M_1 to (X_0^1, X_1^1).

2 message blocks (M_0, M_1)

$2^{123.64}$ 1.5 step XOEsch384 evaluations

Speedup factor: $2^{4.36}$

Preimage Attack on 2.5-Step XOEsch384 with 128-bit Digest



$$\delta = (1001)_\delta, \mathbb{M} = \begin{Bmatrix} (25,9)_\lambda, (26,10)_\lambda, (27,11)_\lambda, (28,12)_\lambda, \\ (29,13)_\lambda, (30,14)_\lambda, (11,27)_\lambda, (12,28)_\lambda, \end{Bmatrix}$$

correlation $\mathfrak{c}_j \geq 0.998$

$(\gamma_0, \gamma_1, \delta)$ satisfying

$$A_{c_0}^{-1}(\gamma_0) \oplus A_{c_0}^{-1}(\gamma_0 \oplus \delta) = A_{c_1}^{-1}(\gamma_1) \oplus A_{c_1}^{-1}(\gamma_1 \oplus \delta).$$

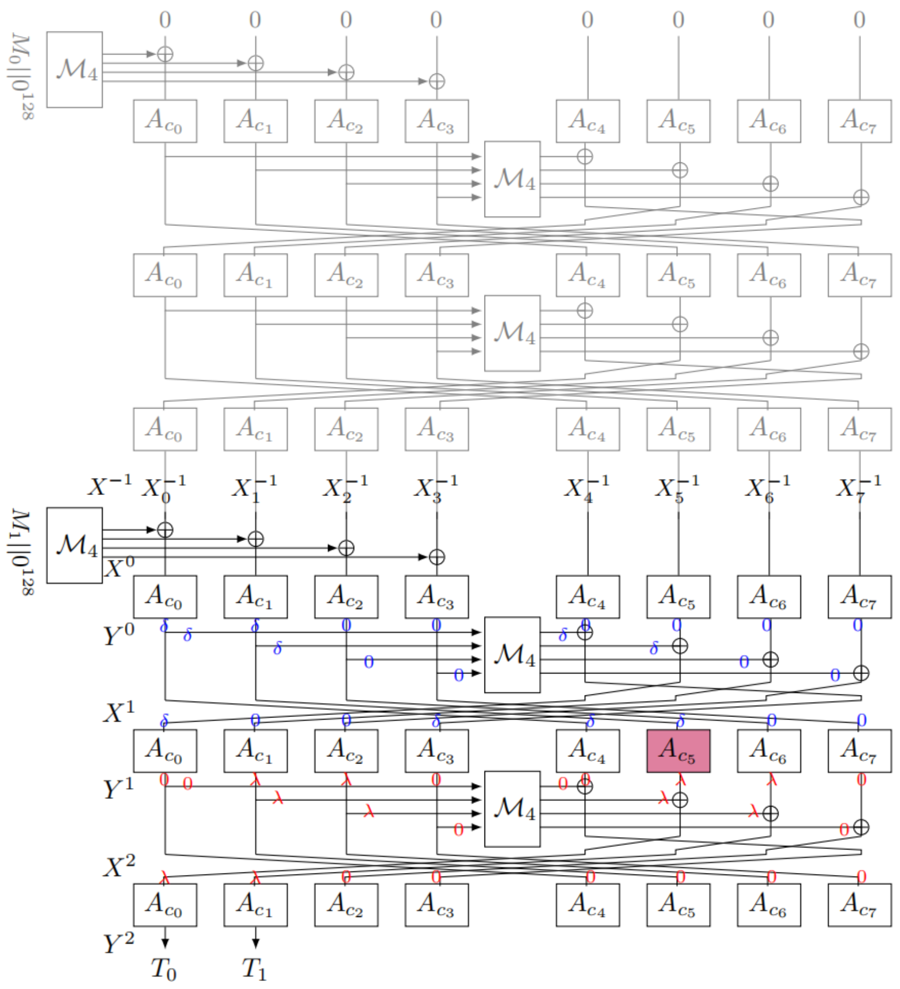
holds with a probability of about 2^{-64}

trying 2^{74} different (γ_0, γ_1) for one δ

$F_{LSL} : \mathbb{F}_2^{256} \to \mathbb{F}_2^{128}$ mapping $(Y_0^0, Y_1^0, Y_2^0, Y_3^0)$ to (X_0^2, X_1^2)

2 message blocks (M_0, M_1)

Preimage Attack on 2.5-Step XOEsch384 with 128-bit Digest



$$\delta = (1001)_\delta, \mathbb{M} = \begin{cases} (25,9)_\lambda, (26,10)_\lambda, (27,11)_\lambda, (28,12)_\lambda, \\ (29,13)_\lambda, (30,14)_\lambda, (11,27)_\lambda, (12,28)_\lambda, \end{cases}$$

correlation $\mathfrak{c}_j \geq 0.998$

Speedup factor: $2^{2.24}$

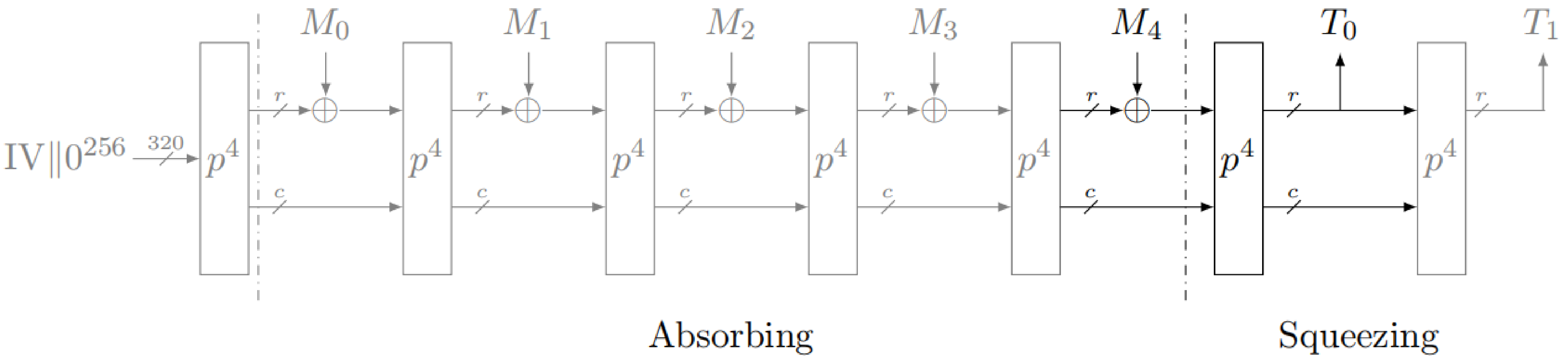
$2^{125.76}$ 2.5 step XOEsch384 evaluations

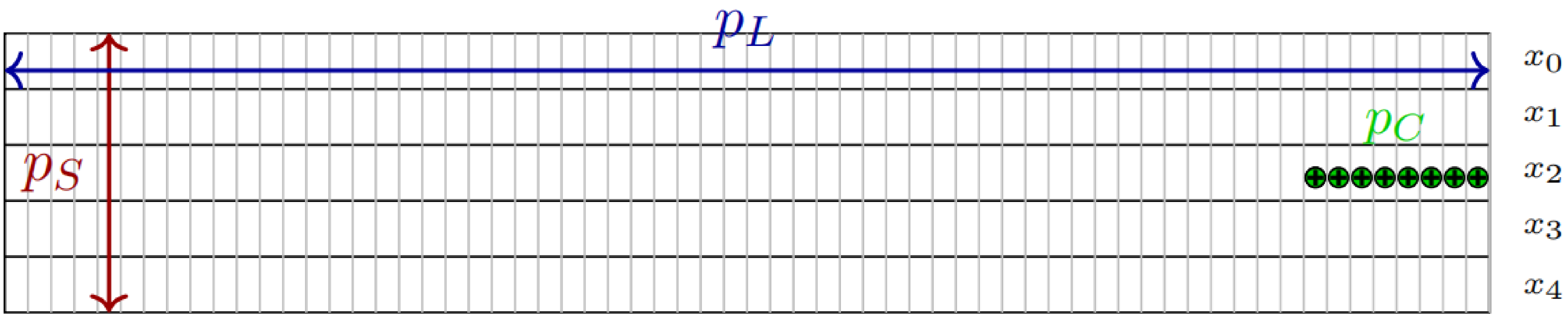
$F_{LSL} : \mathbb{F}_2^{256} \rightarrow \mathbb{F}_2^{128}$ mapping $(Y_0^0, Y_1^0, Y_2^0, Y_3^0)$ to (X_0^2, X_1^2)

2 message blocks (M_0, M_1)

Preimage Attack on 3-ROUND ASCON-XOF WITH 128 BITS

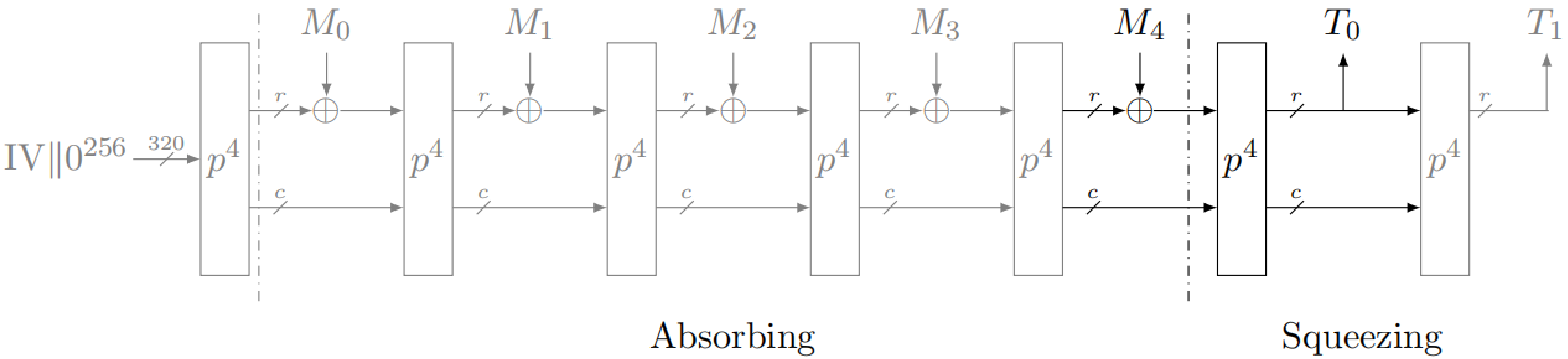
The rate is 64-bit.





Preimage Attack on 3-ROUND ASCON-XOF WITH 128 BITS

The rate is 64-bit.



$$\delta_i = (i), \mathbb{M}_i = \left\{ \begin{array}{l} (i+2),(i+9),(i+12),(i+15),(i+21), \\ (i+22),(i+30),(i+31),(i+32),(i+33), \\ (i+37),(i+43),(i+44),(i+50),(i+52), \\ (i+53),(i+54),(i+55),(i+57),(i+59), \\ (i+63) \end{array} \right\} \quad 0 \leq i < 64$$

$$\Delta_{i+64} = (i, i+22), \mathbb{M}_{i+64} = \left\{ \begin{array}{l} (i+2),(i+12),(i+15),(i+21), \\ (i+31),(i+37),(i+43),(i+44), \\ (i+52),(i+53),(i+54),(i+55), \\ (i+59),(i+62) \end{array} \right\} \quad 0 \leq i < 64$$

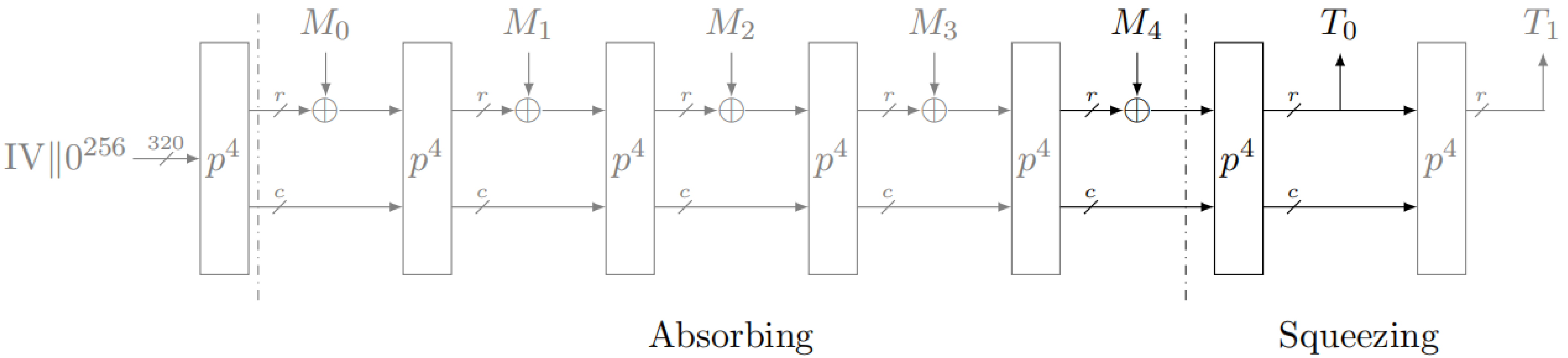
$$\Delta_i, i \in \{63, 105, 127\}$$

$2^{120.02}$ ASCON-XOF

Speedup factor: 2^8

Preimage Attack on 4-ROUND ASCON-XOF WITH 128 BITS

The rate is 64-bit.



$$(i) \xrightarrow[0.25]{4R} (i+8), (i) \xrightarrow[0.25]{4R} (i+30), (i) \xrightarrow[0.44]{4R} (i+50), (i) \xrightarrow[0.50]{4R} (i+54),$$

$$(i) \xrightarrow[0.14]{4R} (i+27), (i) \xrightarrow[0.16]{4R} (i+47).$$

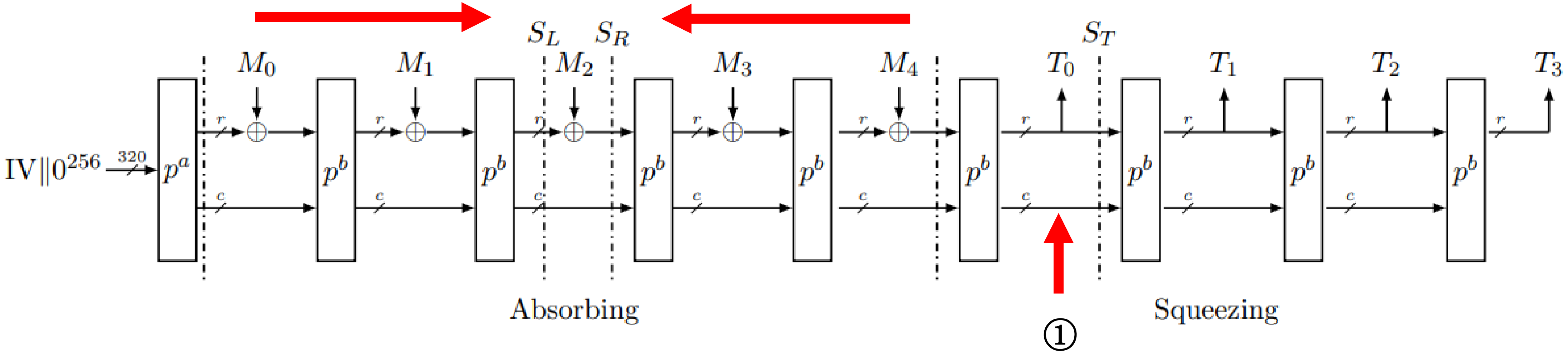
$2^{125.47}$ ASCON-XOF

Speedup factor: $2^{2.53}$

$$0 \leq i < 63$$

Preimage Attack on ASCON-Hash with State Recovery and MITM

The rate is 64-bit.



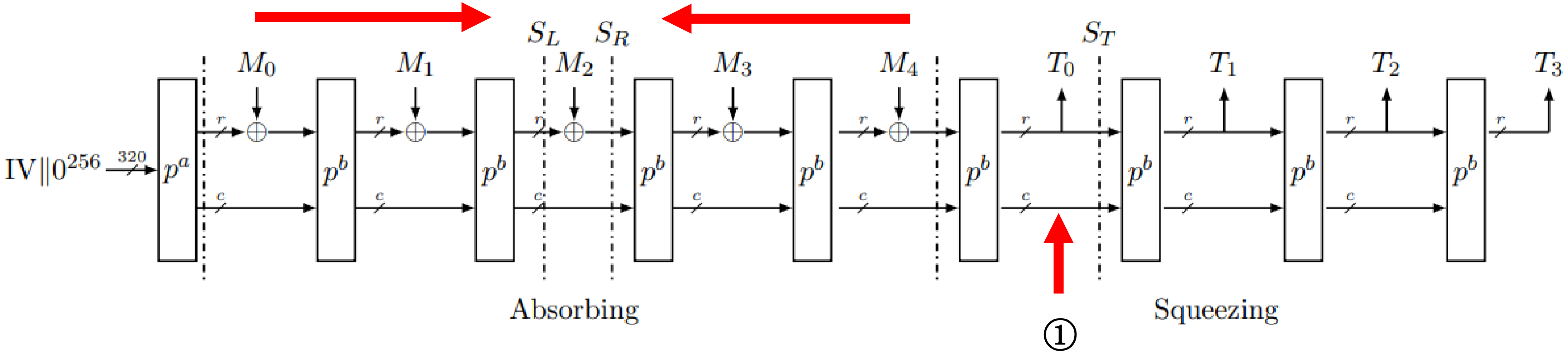
Remark. The designers claimed that `Ascon-HASH` provides 128-bit security with respect to preimage attacks [DEMS21]. However, at CRYPTO 2022 [LM22], Lefevre and Mennink proved that the preimage security bound of a sponge built on an ideal permutation is around $\min\{\max\{n - r', c/2\}, n\}$-bit, where n is the digest size, c the capacity of the sponge (during absorption), and r' the rate (during squeezing). Considering this proof, the preimage security bound of `Ascon-HASH` can be updated to 2^{192} from 2^{128}.

Yu Sasaki. Memoryless Unbalanced Meet-in-the-Middle Attacks: Impossible Results, and applications. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *Applied Cryptography and Network Security - 12th International Conference, ACNS 2014*, volume 8479 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2014.

Floyd's cycle-finding algorithm

Preimage Attack on 3-ROUND ASCON-Hash with State Recovery and MITM

The rate is 64-bit.



$$\delta_{64,i} = (64+i), \mathbb{M}_{64,i} = \begin{Bmatrix} (i+9),(i+15),(i+21),(i+22),(i+30), \\ (i+31),(i+32),(i+33),(i+37),(i+43), \\ (i+44),(i+50),(i+52),(i+53),(i+54), \\ (i+55),(i+57),(i+59),(i+63) \end{Bmatrix}$$

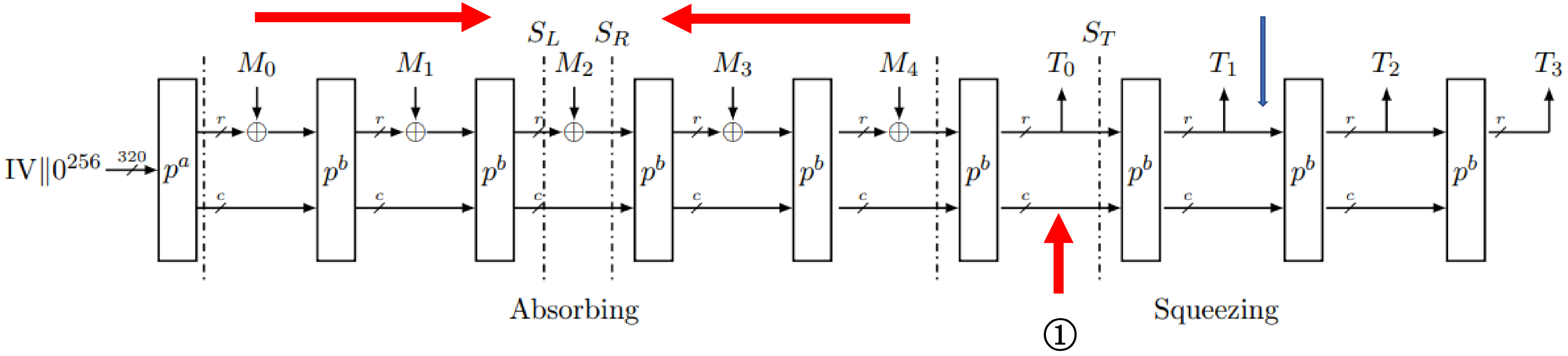
The correlation of all DL distinguishers $(\delta_{64,i}, \lambda_{i,j})$ and $\lambda_{i,j} \in \mathbb{M}_{64,i}$ are 1

$2^{183.98}$ ASCON Hash

Speedup factor: 2^8

Preimage Attack on 3-ROUND ASCON-Hash with State Recovery and MITM

The rate is 64-bit.



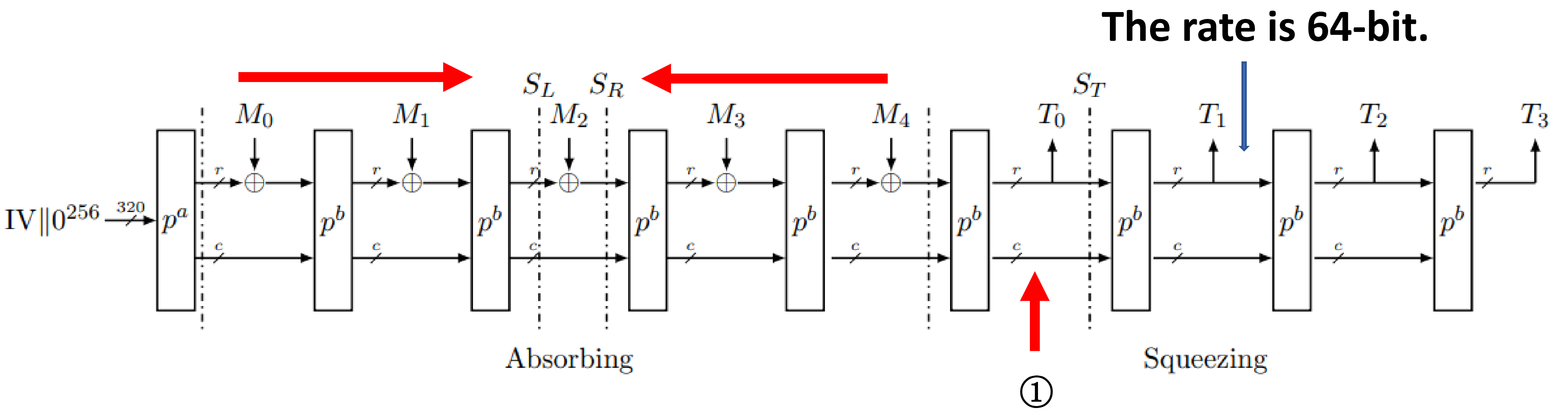
$$\delta_{64,i} = (64+i), \mathbb{M}_{64,i} = \left\{ \begin{matrix} (i+9),(i+15),(i+21),(i+22),(i+30), \\ (i+31),(i+32),(i+33),(i+37),(i+43), \\ (i+44),(i+50),(i+52),(i+53),(i+54), \\ (i+55),(i+57),(i+59),(i+63) \end{matrix} \right\}$$

The correlation of all DL distinguishers $(\delta_{64,i}, \lambda_{i,j})$ and $\lambda_{i,j} \in \mathbb{M}_{64,i}$ are 1

$2^{183.98}$ ASCON Hash

Speedup factor: 2^8

Preimage Attack on 4-ROUND ASCON-Hash with State Recovery and MITM

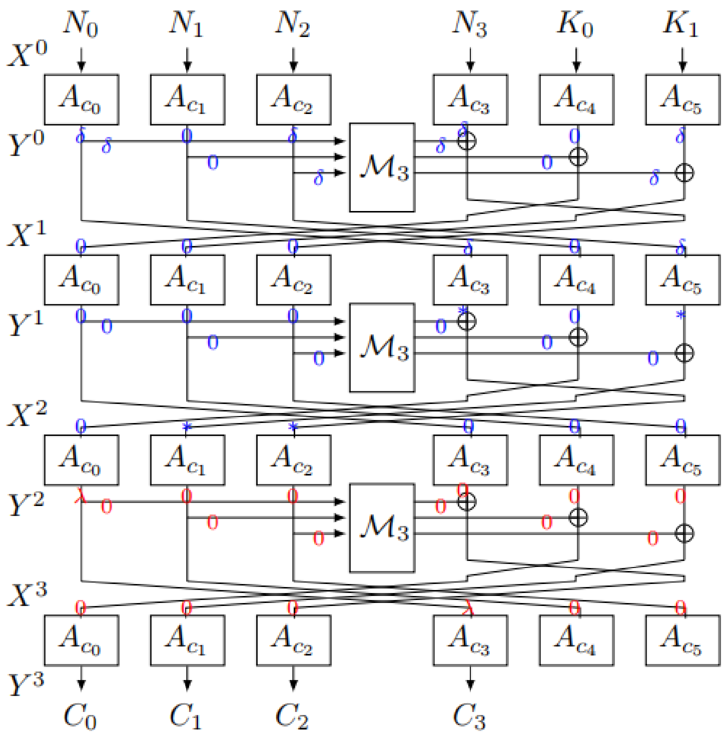


$$(128+i) \xrightarrow[0.36]{4R} (i+32), (128+i) \xrightarrow[0.68]{4R} (i+54), (128+i) \xrightarrow[0.24]{4R} (i+60).$$

$2^{188.61}$ ASCON Hash

Speedup factor: $2^{3.39}$

Key Recovery Attack on 3.5-Step Schwaemm 256-128



2^{65.3} Schwaemm 256-128

Speedup factor： 2^{63}
compared with exhaustive search

Memory: 2^{64}

Input difference space:

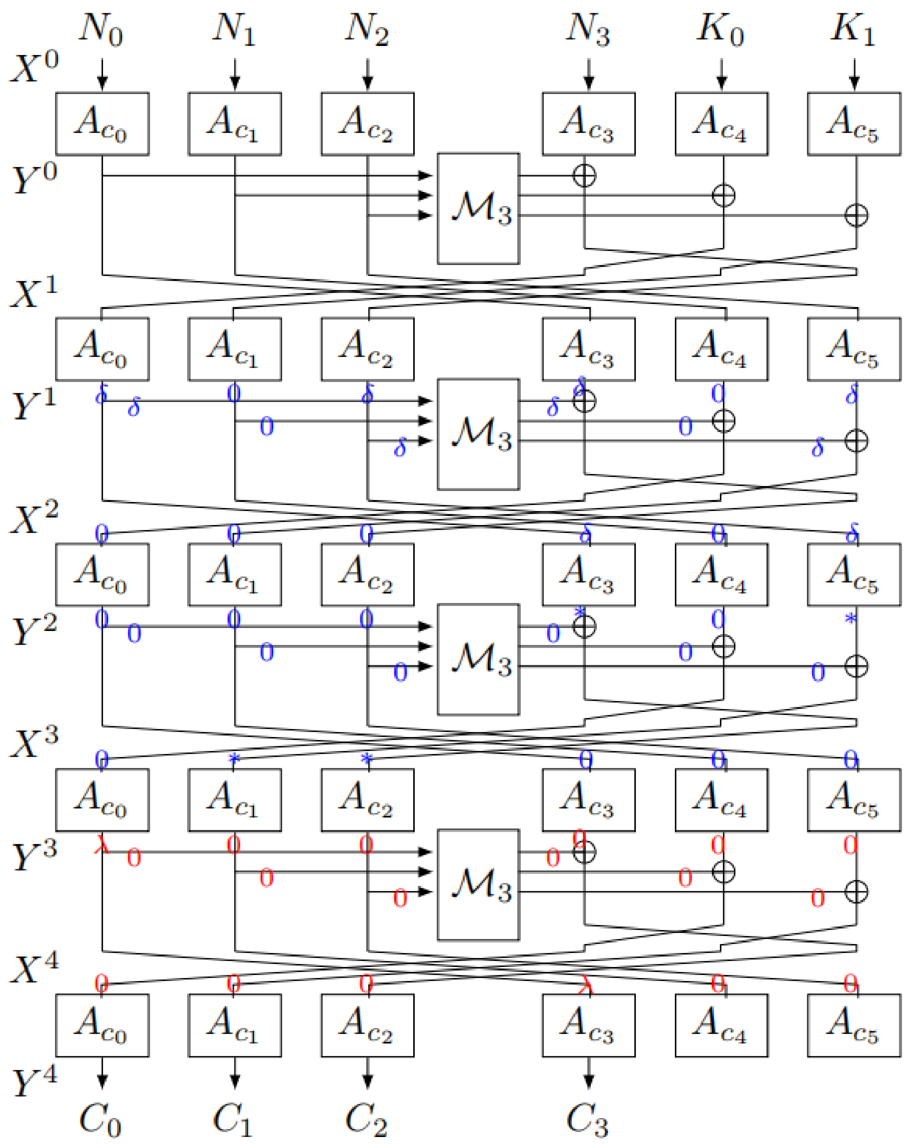
$$\hat{\mathbb{D}}_K = \{(0, \delta) : \delta \in \mathbb{F}_2^{64}\}$$

Output mask set for each input difference:

$$\mathbb{M} = \{(0, 0, 0, e_i, 0, 0) : 0 \leq i < 64\}$$

$$\mathbb{D} = \{(\delta, 0, \delta, \delta, 0, \delta) : \delta \in \mathbb{F}_2^{64} \backslash \{0\}\}$$

Key Recovery Attack on 4.5-Step Schwaemm 256-256-Extend one Step



2^{65.4} Schwaemm 256-128

Speedup factor： 2^{63} compared with exhaustive search

Success Probability： 0.63

Memory: 2^{64}

A Summary of Applications

Table 1: The preimage and collision attacks on XOEsch, Ascon-XOF and Ascon-HASH. Except for the 6-round preimage attack on Ascon-XOF, the success probability of all preimage attacks in this table are approximately 0.63.

Target	Attack type	Round (Step)	Time	Mem.	Output length	Security claim	Meth.	Ref.
XOEsch384	Preimage	1.5	$2^{123.64}$	Neg.	128	2^{128}	DL	Sect. 5.2
		1.5	$2^{186.64}$	Neg.	192	2^{192}	DL	Sect. 5.2
		2.5	$2^{125.76}$	2^{11}	128	2^{128}	DL	Sect. 5.3
		2.5	$2^{188.76}$	2^{11}	192	2^{192}	DL	Sect. 5.3
XOEsch256	Preimage	1.5	$2^{123.64}$	Neg.	128	2^{128}	DL	Sect. F.1
		2.5	$2^{125.66}$	2^{11}	128	2^{128}	DL	Sect. F.2
Ascon-XOF	Preimage	2	2^{103}	Neg.	128	2^{128}	Cube-like	[ASC]
		3	$2^{120.58}$	2^{39}	128	2^{128}	MitM	[QHD+23]
		3	$2^{114.53}$	2^{30}	128	2^{128}	MitM	[QZH+23]
		3	$2^{112.21}$	Neg.	128	2^{128}	Lin.	[LHC+23]
		3	$2^{120.02}$	Neg.	128	2^{128}	DL	Sect. K
		4	$2^{124.67}$	2^{50}	128	2^{128}	MitM	[QHD+23]
		4	$2^{124.49}$	Neg.	128	2^{128}	Lin.	[LHC+23]
		4	$2^{125.47}$	Neg.	128	2^{128}	DL	Sect. 6
		6†	$2^{127.3}$	Neg.	128	2^{128}	Algebraic	[DEMS21]
Ascon-HASH	Preimage	3	$2^{183.98}$	Neg.	256	2^{192}	MitM-DL	Sect. L
		4	$2^{188.61}$	Neg.	256	2^{192}	MitM-DL	Sect. 7
	Collision	2	2^{125}	Neg.	128	2^{128}	Diff.	[ZDW19]
		2	2^{103}	Neg.	128	2^{128}	Diff.	[GPT21]
		3	$2^{121.85}$	2^{121}	128	2^{128}	MitM	[QZH+23]
		4	$2^{126.77}$	2^{126}	128	2^{128}	MitM	[QZH+23]

DL: Differential-linear, Lin.: Linearization, †: No padding bits

Table 2: Results on AEADs and block ciphers. Note that all previous state-recovery attacks on Schwaemm AEADs either omit the whitening (labeled by ⊖) or surpass the data limit set by designers (labeled by ⊘). The success probability of all our key-recovery attacks for 4.5-step Schwaemm is 0.63.

Target	Attack type	Step	Time	Data	Mem.	Security claim	Method	Ref.
Schwaemm 256-128	Key-rec.	3.5	$2^{65.3}$	2^{64}	2^{64}	2^{120}	DL	Sect. 8.1
		3.5	2^{64}	1	Neg.	2^{120}	Structural	Sect. M
		4.5	$2^{65.4}$	2^{64}	2^{64}	2^{120}	DL	Sect. 8.2
Schwaemm 192-192	State-rec.⊖	3.5	2^{128}	2^{64}	2^{128}	2^{184}	Data T-O	[BBdS+21]
	Key-rec.	3.5	2^{129}	2^{64}	2^{64}	2^{184}	DL	Sect. N.1
	State-rec.⊖	4.5	$2^{128+\tau}$	$2^{128-\tau}$	$2^{128+\tau}$	2^{184}	Bir. Diff.	[BBdS+21]
	Key-rec.	4.5	2^{129}	2^{64}	2^{64}	2^{184}	DL	Sect. N.1
Schwaemm 256-256	State-rec.⊖	3.5	2^{192}	2^{64}	2^{192}	2^{248}	Data T-O	[BBdS+21]
	State-rec.⊖	3.5	2^{192}	1	Neg.	2^{248}	Bir. Diff.	[BBdS+21]
	State-rec.⊘	3.5	$2^{224+\tau}$	$2^{224-\tau}$	$2^{224+\tau}$	2^{248}	Bir. Diff.	[BBdS+21]
	Key-rec.	3.5	$2^{129.32}$	2^{128}	2^{128}	2^{248}	DL	Sect. N.2
	State-rec.⊖	4.5	$2^{192}+2^{160+\tau}$	$2^{160-\tau}$	2^{192}	2^{248}	Bir. Diff.	[BBdS+21]
	Key-rec.	4.5	$2^{129.37}$	2^{128}	2^{128}	2^{248}	DL	Sect. N.2
Schwaemm 128-128	State-rec.⊖	3.5	2^{64}	2^{64}	2^{64}	2^{120}	Data T-O	[BBdS+21]
	Key-rec.	3.5	$2^{65.32}$	2^{64}	2^{64}	2^{120}	DL	Sect. N.3
	State-rec.⊖	4.5	$2^{96+\tau}$	$2^{96-\tau}$	$2^{96+\tau}$	2^{120}	Guess Det.	[BBdS+21]
	Key-rec.	4.5	$2^{65.37}$	2^{64}	2^{64}	2^{120}	DL	Sect. N.3
Crax-S-10	Key-rec.	10	$2^{127.53}$	2	Neg.	2^{128}	DL	Sect. O

DL: Differential-linear, Data. T-O: Data trade-off, Bir. Diff.: Birthday differential

Thanks!