1st and 2nd Preimage Attacks on 7, 8 and 9 Rounds of Keccak-224,256,384,512

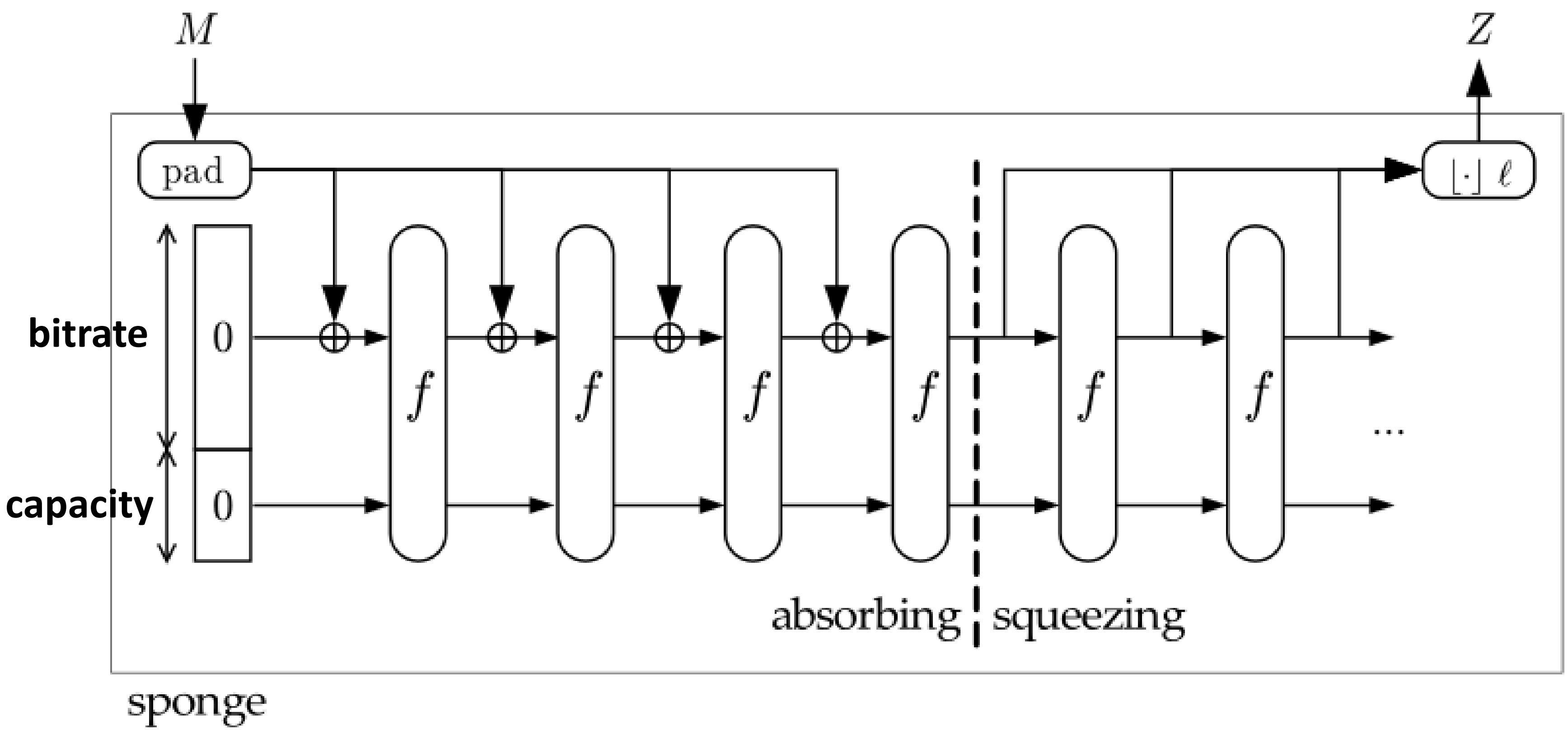
Donghoon Chang[1], Arnab Kumar[2], Pawel Morawiecki[3], Somitra Kumar Sanadhya[1]

[1]IIIT-Delhi, India [2]NSIT, India

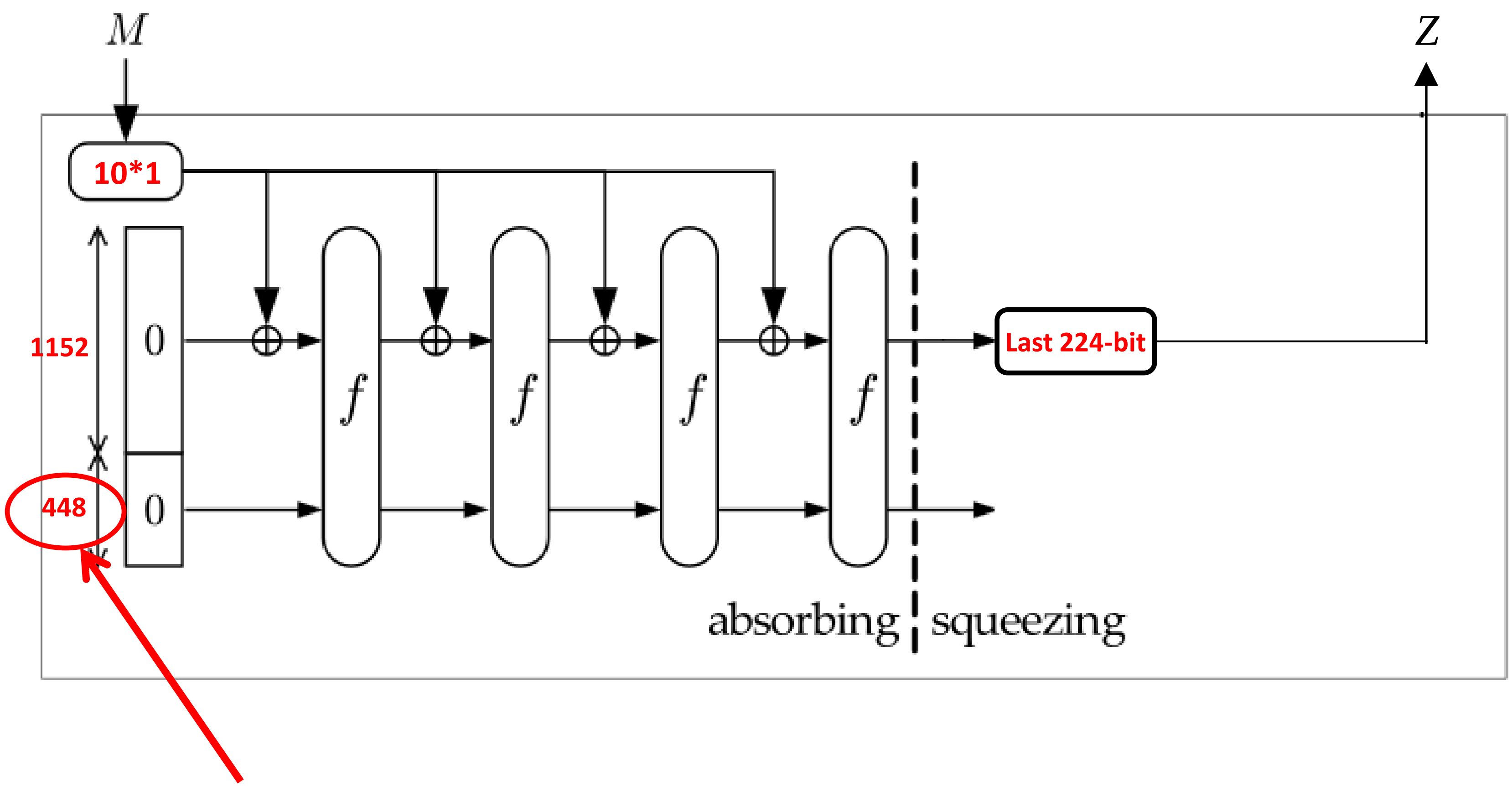
[3]Polish Academy of Sciences, Institute of Computer Science, Poland

Presented at 2014 SHA3 Workshop, Santa Barbara USA
August 22 2014

Sponge Construction

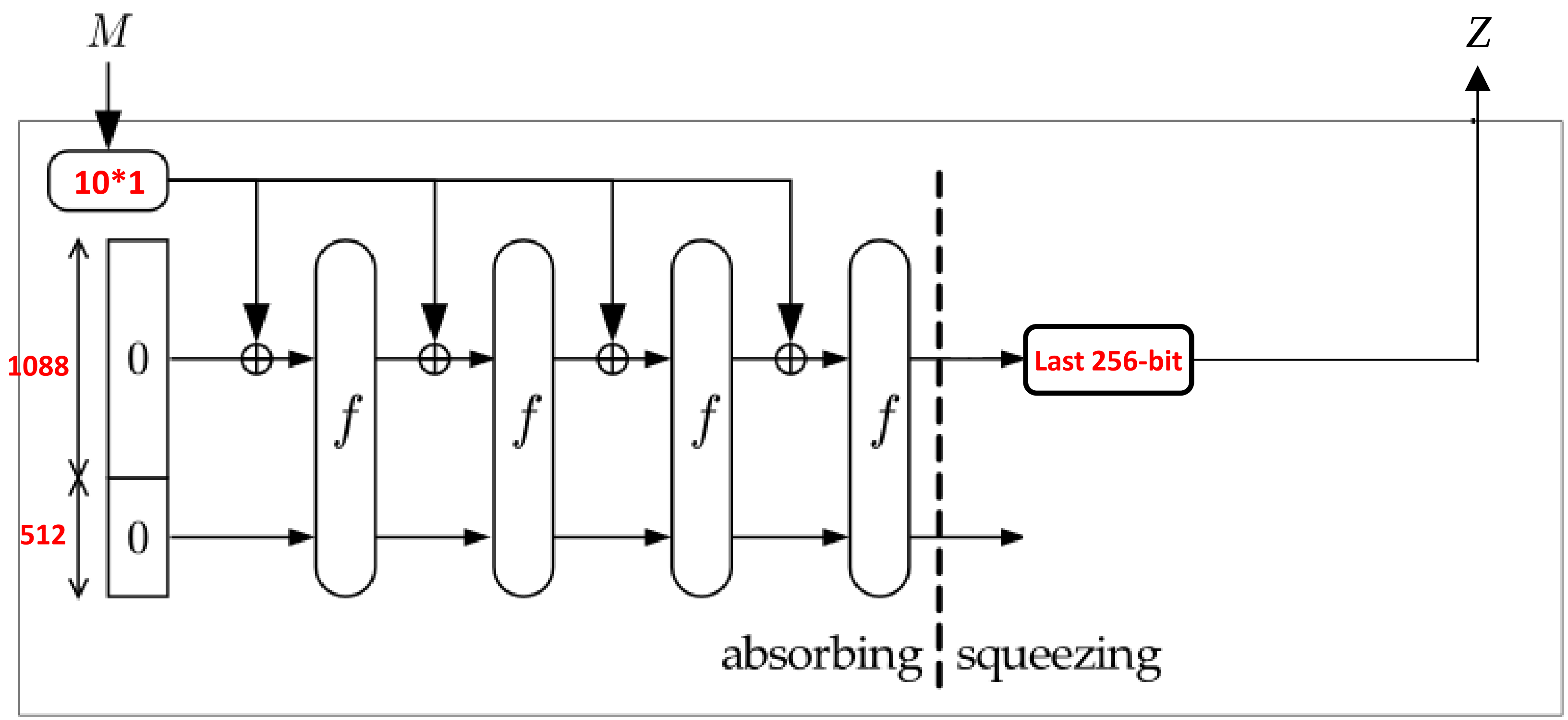


Domain Extension of Keccak-224



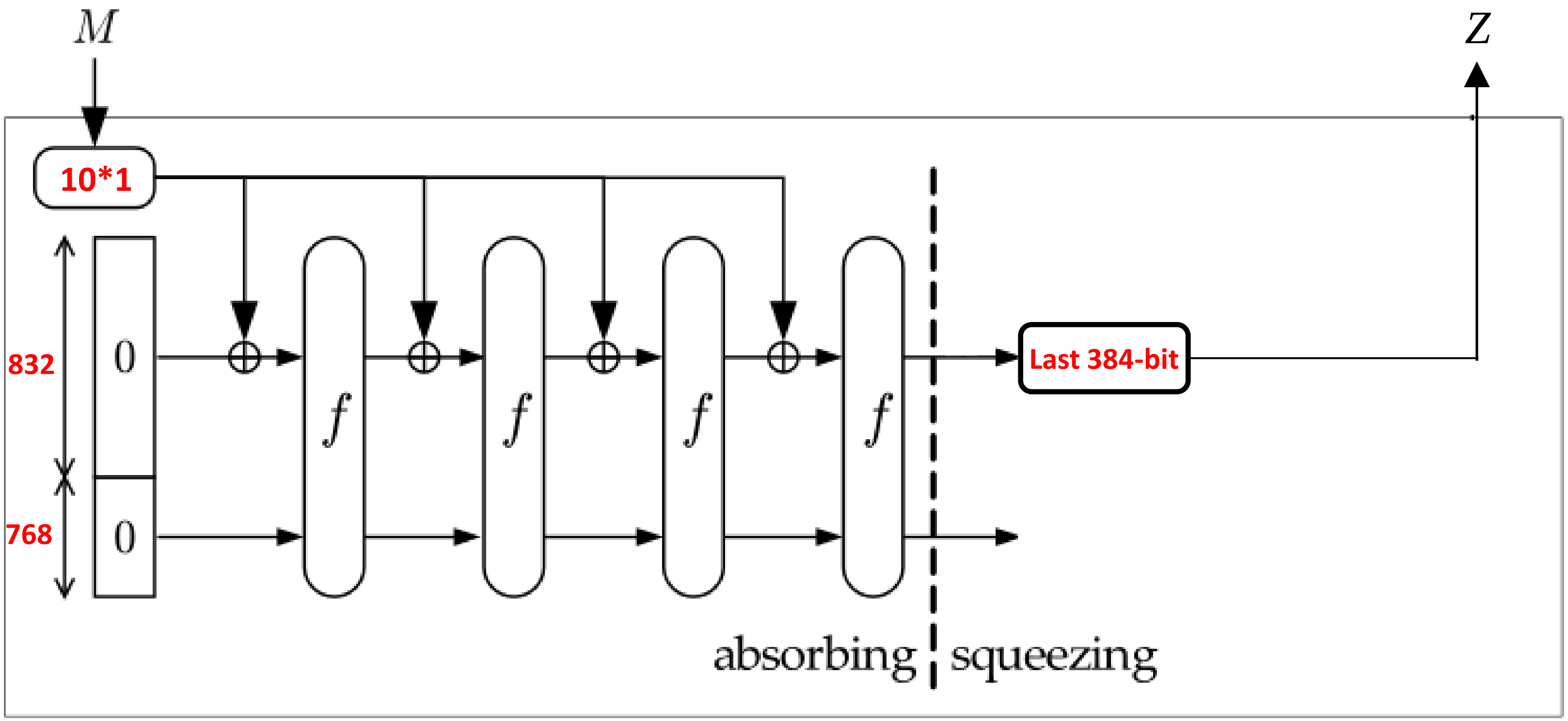
The size of capacity is double of the hash output size.

Domain Extension of Keccak-256



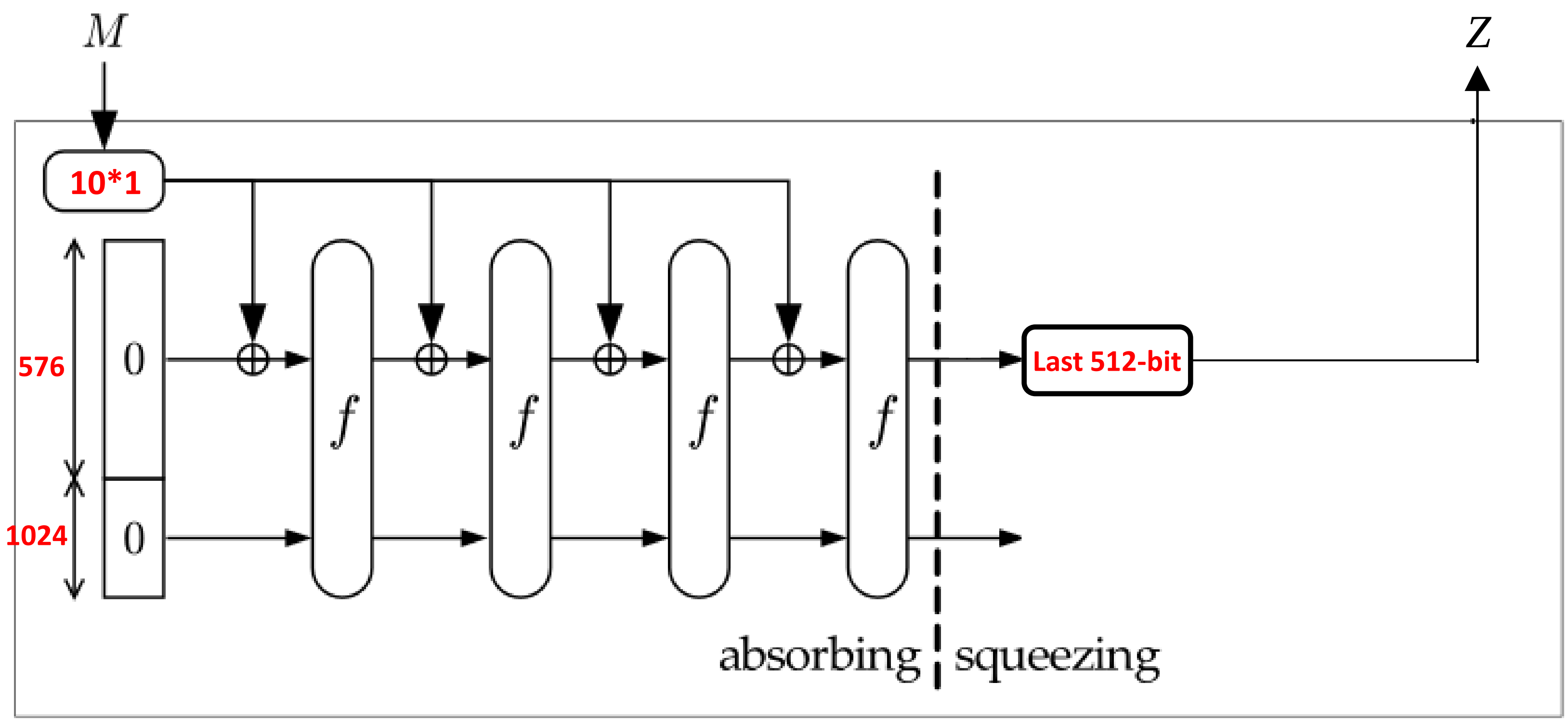
The size of capacity is double of the hash output size.

Domain Extension of Keccak-384



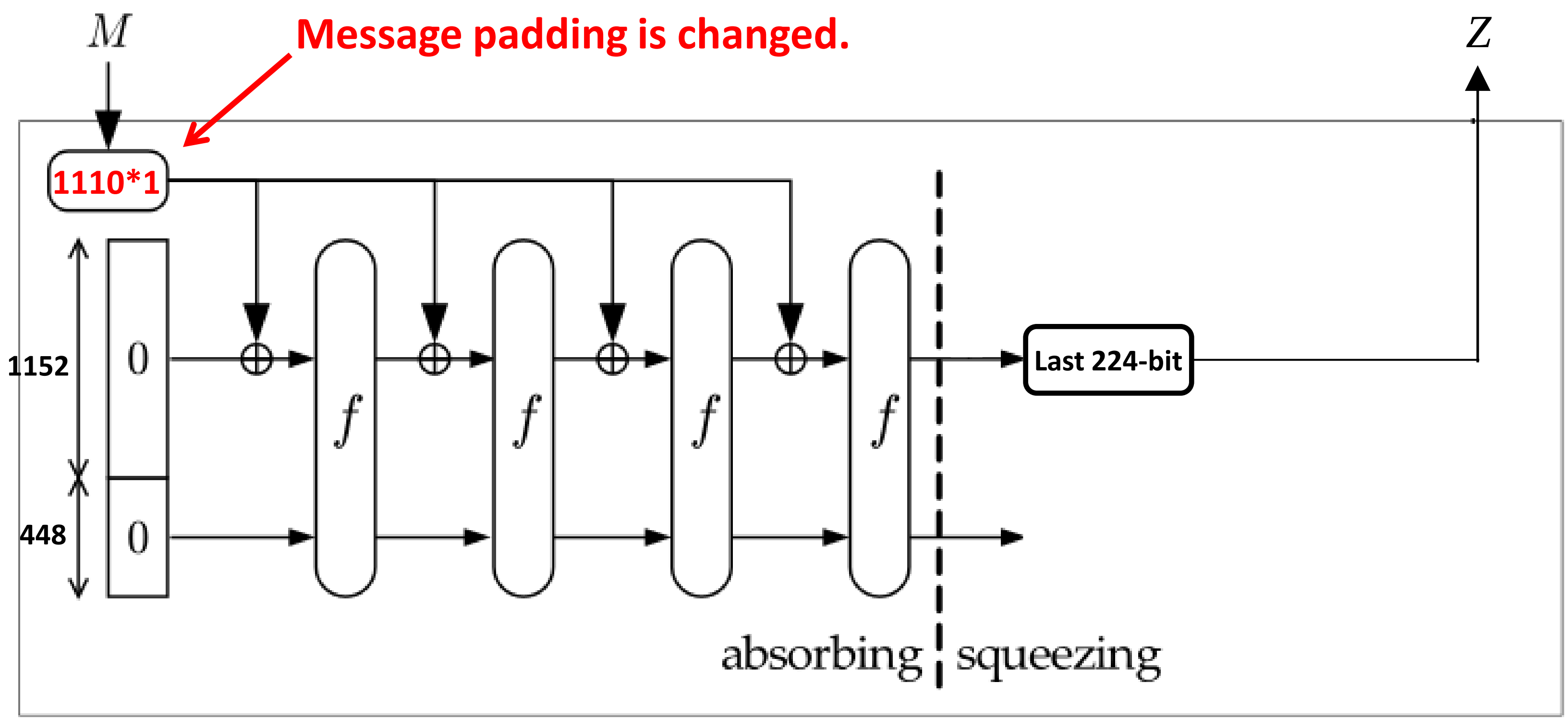
The size of capacity is double of the hash output size.

Domain Extension of Keccak-512



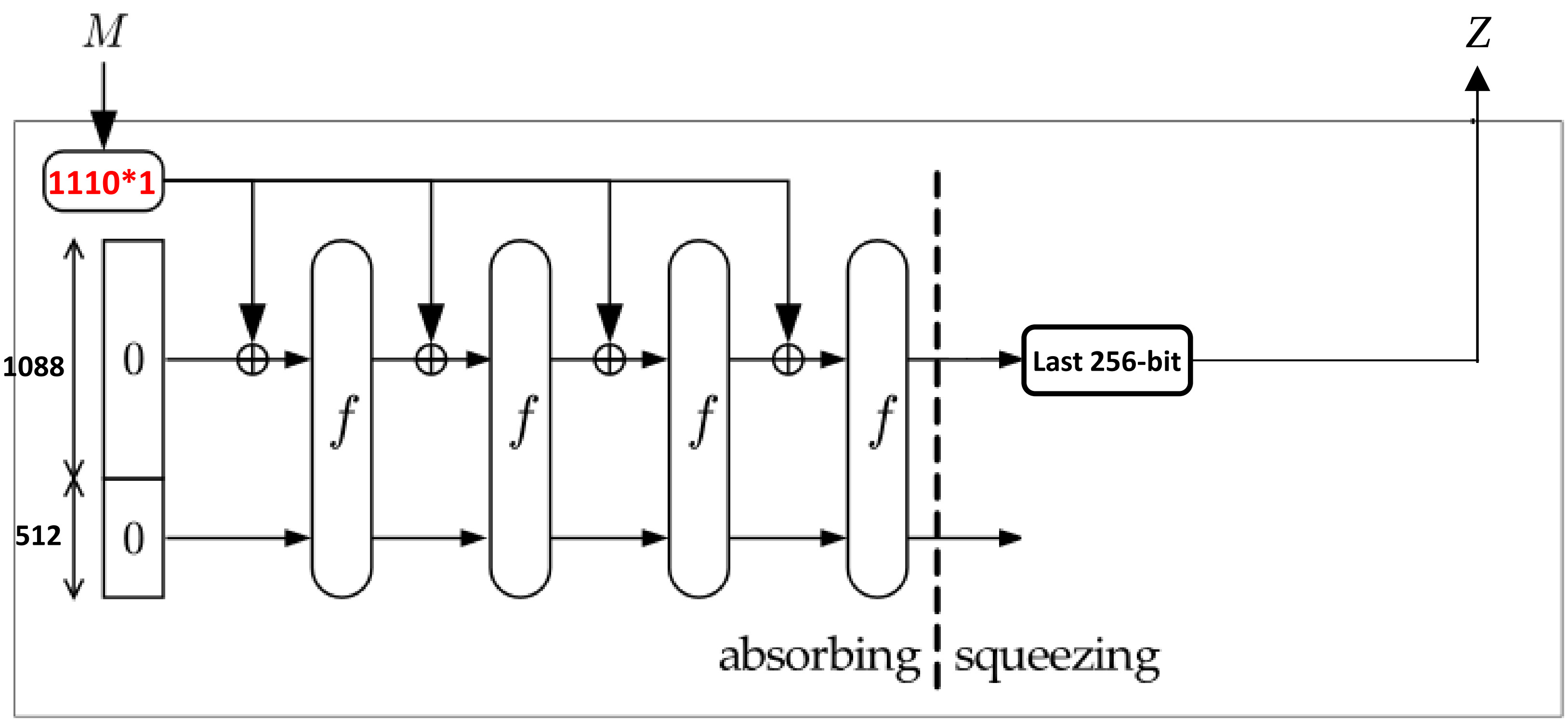
The size of capacity is double of the hash output size.

Domain Extension of SHA3-224



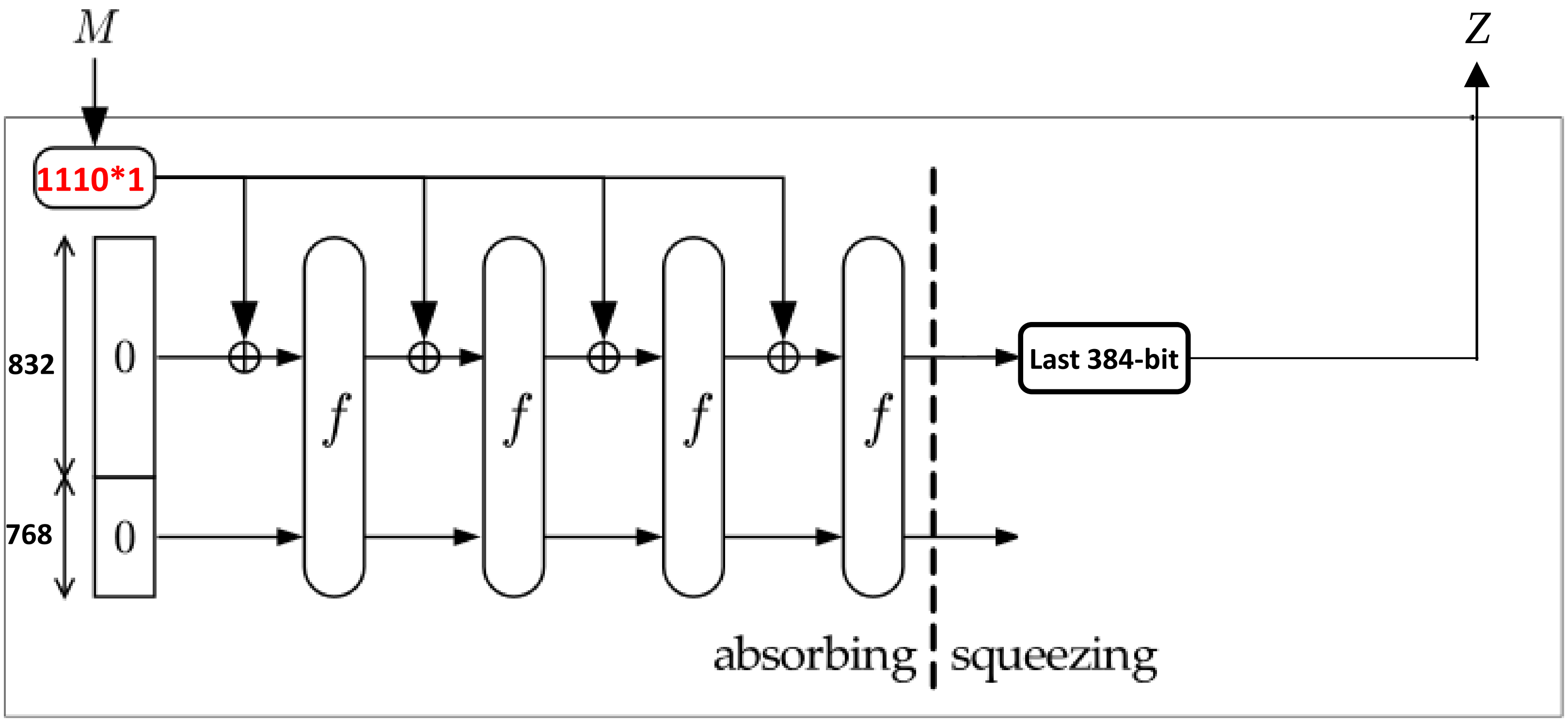
The size of capacity is double of the hash output size.

Domain Extension of SHA3-256



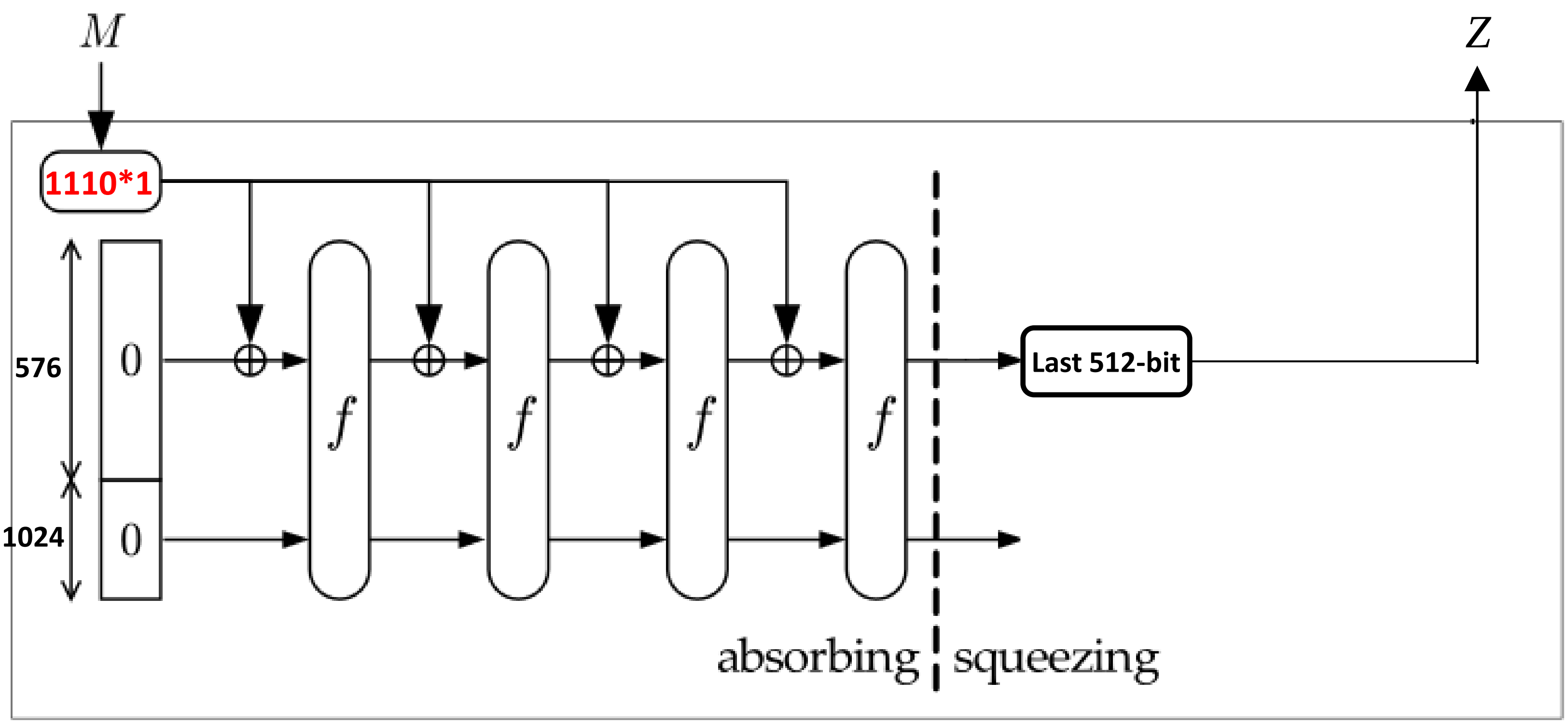
The size of capacity is double of the hash output size.

Domain Extension of SHA3-384



The size of capacity is double of the hash output size.

Domain Extension of SHA3-512



The size of capacity is double of the hash output size.

1600-bit Permutation f

- A 1600-bit state is described by $a[x][y][z]$ for $0 \leq x \leq 4$, $0 \leq y \leq 4$, $0 \leq z \leq 63$.
- f consists of 24 rounds. Each round is defined by $\mathbf{R} = \iota \circ \chi \circ \pi \circ \rho \circ \theta$.

degree 1
(0.5 round)

$$\theta : a[x][y][z] \leftarrow a[x][y][z] \oplus \bigoplus_{y'=0}^{4} a[x-1][y'][z] \oplus \bigoplus_{y'=0}^{4} a[x+1][y'][z-1]$$

$$\rho : a[x][y][z] \leftarrow a[x][y][z-(t+1)(t+2)/2],$$

$$\text{with } t \text{ satisfying } 0 \leq t < 24 \text{ and } \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^t \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \text{ in } \mathbf{GF}(5)^{2\times 2},$$

$$\text{or } t = -1 \text{ if } x = y = 0,$$

$$\pi : a[x][y] \leftarrow a[x'][y'], \text{ with } \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix},$$

degree 2
(0.5 round)

$$\chi : a[x] \leftarrow a[x] \oplus (a[x+1] \oplus 1) a[x+2],$$

$$\iota : a \leftarrow a \oplus \mathbf{RC}[i_r],$$

Number of Bit-operations of each Round

For $0 \leq x \leq 4$, $0 \leq y \leq 4$, $0 \leq z \leq 63$.

320 bit-operations

1600 bit-operations

1280 bit-operations

$$\theta : a[x][y][z] \leftarrow a[x][y][z] \oplus \boxed{\bigoplus_{y'=0}^{4} a[x-1][y'][z]} \oplus \bigoplus_{y'=0}^{4} a[x+1][y'][z-1]$$

$$\rho : a[x][y][z] \leftarrow a[x][y][z-(t+1)(t+2)/2],$$

$$\text{with } t \text{ satisfying } 0 \leq t < 24 \text{ and } \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^{t} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \text{ in } \mathbf{GF}(5)^{2\times 2},$$

$$\text{or } t = -1 \text{ if } x = y = 0,$$

$$\pi : a[x][y] \leftarrow a[x'][y'], \text{ with } \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix},$$

$$\chi : a[x] \leftarrow \boxed{a[x] \oplus (a[x+1] \oplus 1) a[x+2],}$$

$$\iota : a \leftarrow a \oplus \mathbf{RC}[i_r],$$

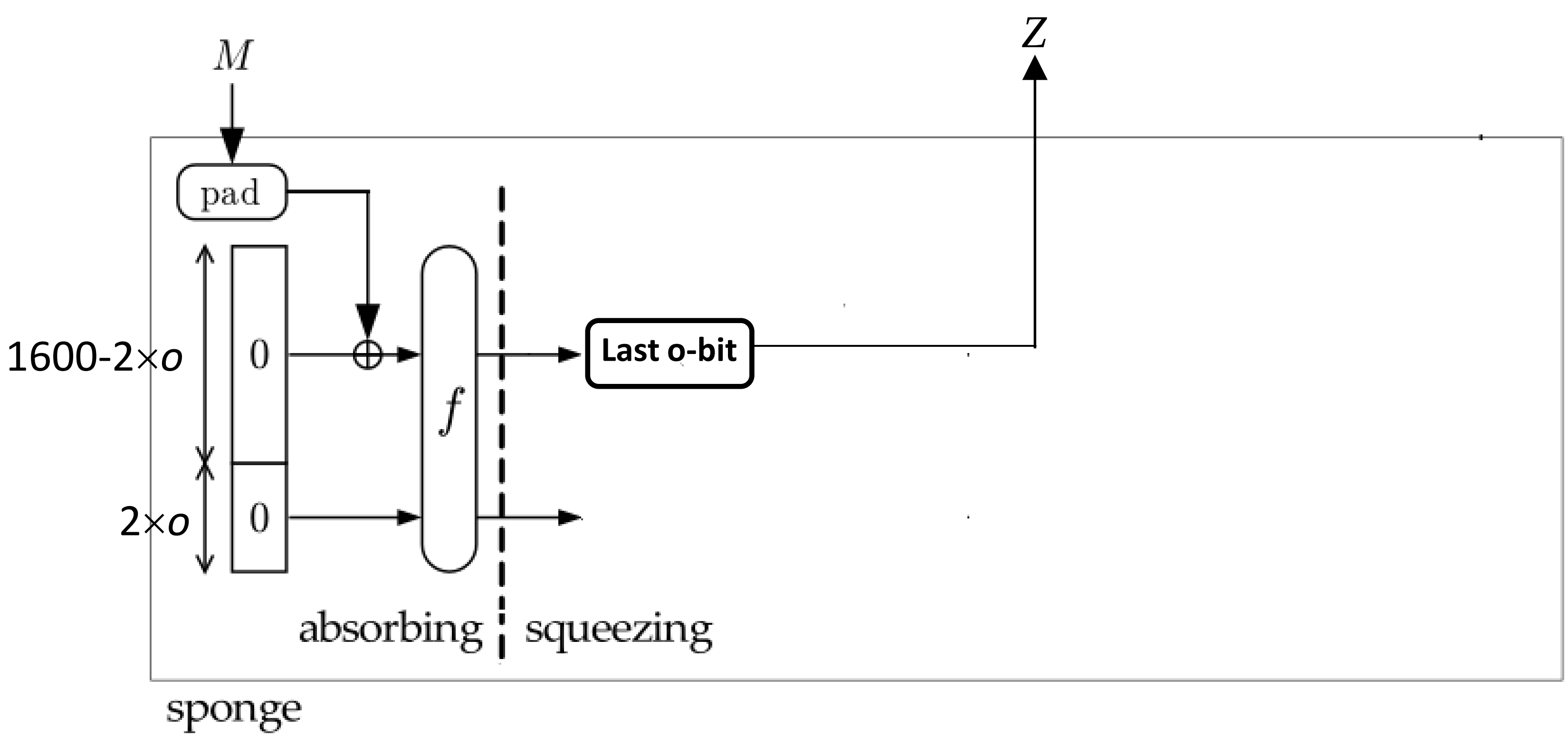
4800 bit-operations

64 bit-operations

In total, <u>at least **8064** (=1600+1280+320+4800+64) bit-operations</u> are required to compute one round.

General Preimage Attack Complexity for Keccak-n and SHA3-n based on r-round *f*

- So, given a *o*-bit hash value *Z*, we need **r×8064** × **2^o** bit-operations to find its preimage with high probability.



Polynomial Enumeration (used by Dinur and Shamir [FSE 2011])

- Given a boolean function f_i ($1 \le i \le b$) with n-bit input and degree d, where f_i is the i-th output bit of f,
- polynomial enumeration algorithm is a way of constructing the truth table of f_i by the following two steps.
  - **Step 1:** Compute coefficients of f_i,
    - **Time complexity:** $\sum_{0 \le j \le d}(2^j \times {}_nC_j)$.
  - **Step 2:** Construct the truth table of f_i using the fast Moebius transformation.
    - **Time complexity:** $n \times 2^{n-1}$.

The Fast Moebius Transformation

- transforms the coefficient array of a boolean function to its truth table array.

For example, $f(x_1,x_2,x_3)=x_1 \oplus x_1x_2x_3 \oplus x_1x_2 \oplus x_3$

Coefficient Array

$$\left(\begin{array}{c}0\\1\\0\\0\\1\\0\\1\\1\end{array}\right)\begin{array}{l} \\ \leftarrow \text{coefficient of } x_3\\ \\ \\ \leftarrow \text{coefficient of } x_1\\ \\ \leftarrow \text{coefficient of } x_1x_2\\ \leftarrow \text{coefficient of } x_1x_2x_3\end{array}$$

The Fast Moebius Transformation

- transforms the coefficient array of a boolean function to its truth table array.

For example, $f(x_1,x_2,x_3)=x_1 \oplus x_1x_2x_3 \oplus x_1x_2 \oplus x_3$

Coefficient Array

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

The Fast Moebius Transformation

- transforms the coefficient array of a boolean function to its truth table array.

For example, $f(x_1,x_2,x_3)=x_1\oplus x_1x_2x_3\oplus x_1x_2\oplus x_3$

Coefficient Array

$$\begin{pmatrix}0\\1\\0\\0\\1\\0\\1\\1\end{pmatrix} \longrightarrow \begin{pmatrix}0\\1\\0\\0\\1\\1\\1\\0\end{pmatrix}$$

The Fast Moebius Transformation

- transforms the coefficient array of a boolean function to its truth table array.

For example, $f(x_1,x_2,x_3)=x_1 \oplus x_1x_2x_3 \oplus x_1x_2 \oplus x_3$

Coefficient Array

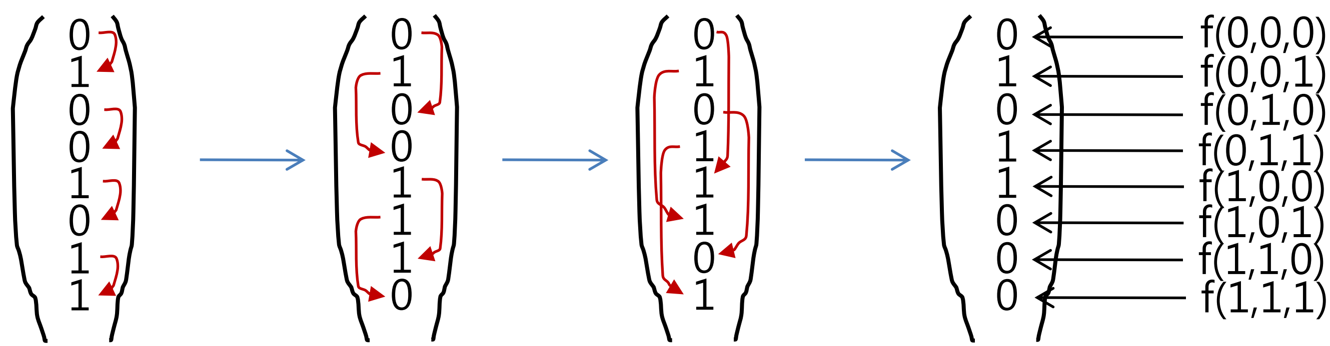
$$\begin{pmatrix} 0\\1\\0\\0\\1\\0\\1\\1 \end{pmatrix} \rightarrow \begin{pmatrix} 0\\1\\0\\0\\1\\1\\1\\0 \end{pmatrix} \rightarrow \begin{pmatrix} 0\\1\\0\\1\\1\\1\\0\\1 \end{pmatrix}$$

The Fast Moebius Transformation

- transforms the coefficient array of a boolean function to its truth table array.

For example, $f(x_1,x_2,x_3)=x_1 \oplus x_1x_2x_3 \oplus x_1x_2 \oplus x_3$

Coefficient Array Truth Table Array



Complexity : for n variables, $n \times 2^{n-1}$ 1-bit XOR operations.

Preimage Attack on H using Polynomial Enumeration (by Dinur and Shamir)

- Given a o-bit hash output *Z*,
  - **Step 1:** By polynomial enumeration algorithm, efficiently find messages M's which partially match over b bits of the given *o*-bit hash value.
  - **Step 2:** if there is M s.t. H(M)=Z, then return M else goes to Step 1.

Improving Polynomial Enumeration (by Bernstein [NIST mailing list 2013])

- Given a boolean function f_i ($1 \le i \le b$) with n-bit input and degree d, where f_i is the i-th output bit of f.
- polynomial enumeration algorithm is a way of constructing the truth table of f_i by the following two steps.
  - **Step 1:** Compute coefficients of f_i,
    - **Time complexity:** $\cancel{\sum_{0 \le j \le d}(2^j \times {}_nC_j)} \Rightarrow \sum_{0 \le j \le d}(j \times {}_nC_j)$.
  - **Step 2:** Construct the truth table of f_i using the fast Moebius transformation.
    - **Time complexity:** $n \times 2^{n-1}$.

Improving Polynomial Enumeration (by Bernstein [NIST mailing list 2013])

- Given a boolean function f_i ($1 \leq i \leq b$) with n-bit input and degree d, where f_i is the i-th output bit of f.
- polynomial enumeration algorithm is a way of constructing the truth table of f_i by the following two steps.
  - **Step 1:** Compute coefficients of f_i,
    - **Time complexity:** $\cancel{\sum_{0 \leq j \leq d} (2^j \times {}_nC_j)}$ $\Rightarrow$ $\sum_{0 \leq j \leq d} (j \times {}_nC_j)$.
  - **Step 2:** Construct the truth table of f_i using the fast Moebius transformation.
    - **Time complexity:** $n \times 2^{n-1}$.

But this time complexity improvement requires big memory cost.

Application to 6, 7, 8 rounds of Keccak-512 (by Bernstein)

- **6 rounds:** 2^{176} bits of memory give a workload reduction by a factor 50 (~6 bits)
- **7 rounds:** 2^{320} bits of memory give a workload reduction by a factor 37 (~5 bits)
- **8 rounds:** 2^{508} bits of memory give a workload reduction by a factor 1.4 (half a bit)

Our Results

- Bernstein only described the idea of improving Step 1 complexity. However, overall time and memory complexity of his attack is not clear.
- **Result 1:** Based on Bernstein's idea, we made **Algorithm 1** for generating the coefficient array of a boolean function with detailed time and memory complexity.
- **Result 2:** We provide a general preimage attack methodology on hash functions using Result 1 and meet-in-the-middle-matching technique.
- **Result 3:** Using Result 2, as an example, we further improve Bernstein's result upto 9 rounds of Keccak.

Algorithm 1 for Generating the Coefficient Array of a Boolean Function (Result 1)

Algorithm 1: Computing the Coefficient Static Array of a Boolean Function

Input: Boolean function f with n-bit input and having algebraic degree at most d
Result: Coeffient static array C of size 2^n, which is initialized with all zeros in the beginning

begin
    l=0;
    while $l \leq d$ **do**
        for $A \in \alpha$ AND $|A| = l$ **do**
            y=0;
            i=0;
            $y=f(S_A)$;
            $\mathrm{Sum}_0[S_A] = y$;
            while $i < l$ **do**
                $y = y \oplus \mathrm{Sum}_i[S_{A,i+1}]$;
                $i=i+1$;
                $\mathrm{Sum}_i[S_A] = y$;
            $C[S_A] = y$, where C_A is also same as $C[S_A]$;
        $l=l+1$;

Time Complexity: $5 \times (\sum_{l=0}^{d} l \times \binom{n}{l}) + T \times \sum_{l=0}^{d} \binom{n}{l}$

Memory Complexity: $(2d+1) \times 2^n + 2^n$

Algorithm 1 for Generating the Coefficient Array of a Boolean Function (Result 1)

Algorithm 1: Computing the Coefficient Static Array of a Boolean Function

Input: Boolean function f with n-bit input and having algebraic degree at most d

Result: Coeffient static array C of size 2^n, which is initialized with all zeros in the beginning

$\alpha = \{A : |A| \leq d \text{ and } A \subset \{1,2,\ldots..,n\}\}$

begin
- l=0;
- **while** $l \leq d$ **do**
  - **for** $A \in \alpha$ AND $|A| = l$ **do**
    - y=0;
    - i=0;
    - y=$f(S_A)$;
    - $\mathrm{Sum}_0[S_A] = y$;
    - **while** $i < l$ **do**
      - $y = y \oplus \mathrm{Sum}_i[S_{A,i+1}]$;
      - i=i+1;
      - $\mathrm{Sum}_i[S_A] = y$;
    - $C[S_A] = y$, where C_A is also same as $C[S_A]$;
  - l=l+1;

Time Complexity: $5 \times (\sum_{l=0}^{d} l \times \binom{n}{l}) + T \times \sum_{l=0}^{d} \binom{n}{l}$

Memory Complexity: $(2d+1) \times 2^n + 2^n$

Algorithm 1 for Generating the Coefficient Array of a Boolean Function (Result 1)

Algorithm 1: Computing the Coefficient Static Array of a Boolean Function

Input: Boolean function f with n-bit input and having algebraic degree at most d

Result: Coeffient static array C of size 2^n, which is initialized with all zeros in the beginning

1 **begin**
2   l=0;
3   **while** $l \leq d$ **do**
4   **for** $A \in \alpha$ AND $|A| = l$ **do**
5   y=0;
6   i=0;
7       $y=f(S_A)$;
8       $\mathrm{Sum}_0[S_A] = y$;
9   **while** $i < l$ **do**
10         $y = y \oplus \mathrm{Sum}_i[S_{A,i+1}]$;
11         $i=i+1$;
12         $\mathrm{Sum}_i[S_A] = y$;
13       $C[S_A] = y$, where C_A is also same as $C[S_A]$;
14     $l=l+1$;

The time complexity of f is **T**.
(in terms of number of bit-operations)

Step 7

Time Complexity: $5 \times (\sum_{l=0}^{d} l \times \binom{n}{l}) + T \times \sum_{l=0}^{d} \binom{n}{l}$

Memory Complexity: $(2d+1) \times 2^n + 2^n$

Algorithm 1 for Generating the Coefficient Array of a Boolean Function (Result 1)

Algorithm 1: Computing the Coefficient Static Array of a Boolean Function

Input: Boolean function f with n-bit input and having algebraic degree at most d
Result: Coeffient static array C of size 2^n, which is initialized with all zeros in the beginning

1 **begin**
2 l=0;
3 **while** $l \leq d$ **do**
4 **for** $A \in \alpha\ AND\ |A| = l$ **do**
5 y=0;
6 i=0;
7 y=$f(S_A)$;
8 $\mathrm{Sum}_0[S_A] = y$;
9 **while** $i < l$ **do**
10 $y = y \oplus \mathrm{Sum}_i[S_{A,i+1}]$;
11 i=i+1;
12 $\mathrm{Sum}_i[S_A] = y$;
13 $C[S_A] = y$, where C_A is also same as $C[S_A]$;
14 l=l+1;

2 bit-operations (1 XOR, 1-bit memory access of static array Sum)

2 bit-operations are needed on average

1 bit -operation (1-bit update of static array Sum)

Step 10,11,12

Time Complexity: $5 \times (\sum_{l=0}^{d} l \times \binom{n}{l}) + T \times \sum_{l=0}^{d} \binom{n}{l}$

Memory Complexity: $(2d+1) \times 2^n + 2^n$

Algorithm 1 for Generating the Coefficient Array of a Boolean Function (Result 1)

Algorithm 1: Computing the Coefficient Static Array of a Boolean Function

Input: Boolean function f with n-bit input and having algebraic degree at most d

Result: Coeffient static array C of size 2^n, which is initialized with all zeros in the beginning

```
begin
  l=0;
  while l ≤ d do
    for A ∈ α AND |A| = l do
      y=0;
      i=0;
      y=f(S_A);
      Sum_0[S_A] = y;
      while i < l do
        y = y ⊕ Sum_i[S_{A,i+1}];
        i=i+1;
        Sum_i[S_A] = y;
      C[S_A] = y, where C_A is also same as C[S_A];
    l=l+1;
```

Current Sum Arrays: Each Sum array (which is static) has 2^n elements of size 1-bit. We need at most **d+1** current Sum arrays.

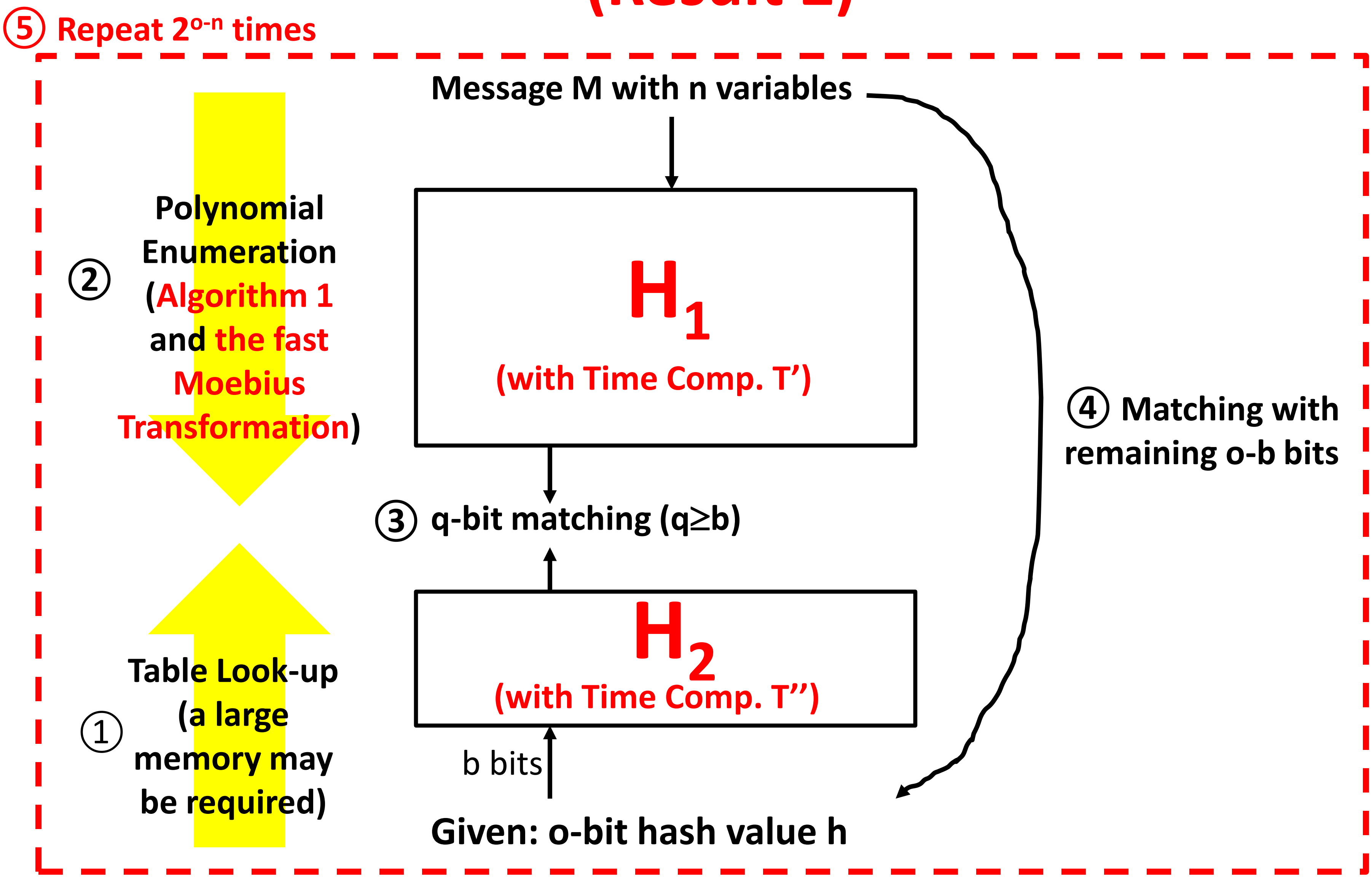
Previous Sum Arrays : Each Sum array (which is static) has 2^n elements of size 1-bit. We need at most **d** previous Sum arrays.

Coefficient Array (which is static) has 2^n elements of size 1-bit.

Time Complexity: $5 \times \left(\sum_{l=0}^{d} l \times \binom{n}{l}\right) + T \times \sum_{l=0}^{d} \binom{n}{l}$

Memory Complexity: $(2d+1) \times 2^n + 2^n$

Our General Preimage Attack on $H=H_2 \circ H_1$ (Result 2)



Complexity of Our General Preimage Attack

(Result 2)

Time Complexity:

① Generating lookup Table for H_2

② Algorithm 1 (here, w=1)

$$
\begin{aligned}
&\boxed{b \times 2^q \times T'' + 2^{q-b} \times (q-b) \times q +} \\
&2^{o-n} \times \Big[\boxed{\Big(T' \times \sum_{j=0}^{d} \binom{n}{j}\Big) + \Big((2w+3) \times q \times \sum_{j=0}^{d} j \times \binom{n}{j}\Big)} + \boxed{(q \times n \times 2^{n-1})}\Big] + \\
&2^{o-n} \times \Big[\boxed{(T \times 2^{n-b})} + \boxed{(\max\{(q-b), 1\} \times 2^n \times q)}\Big],
\end{aligned}
$$

⑤

② the fast Moebius Transformation

④ Matching over remaining o-q bits (where T=T'+T'')

③ Matching over q-bit

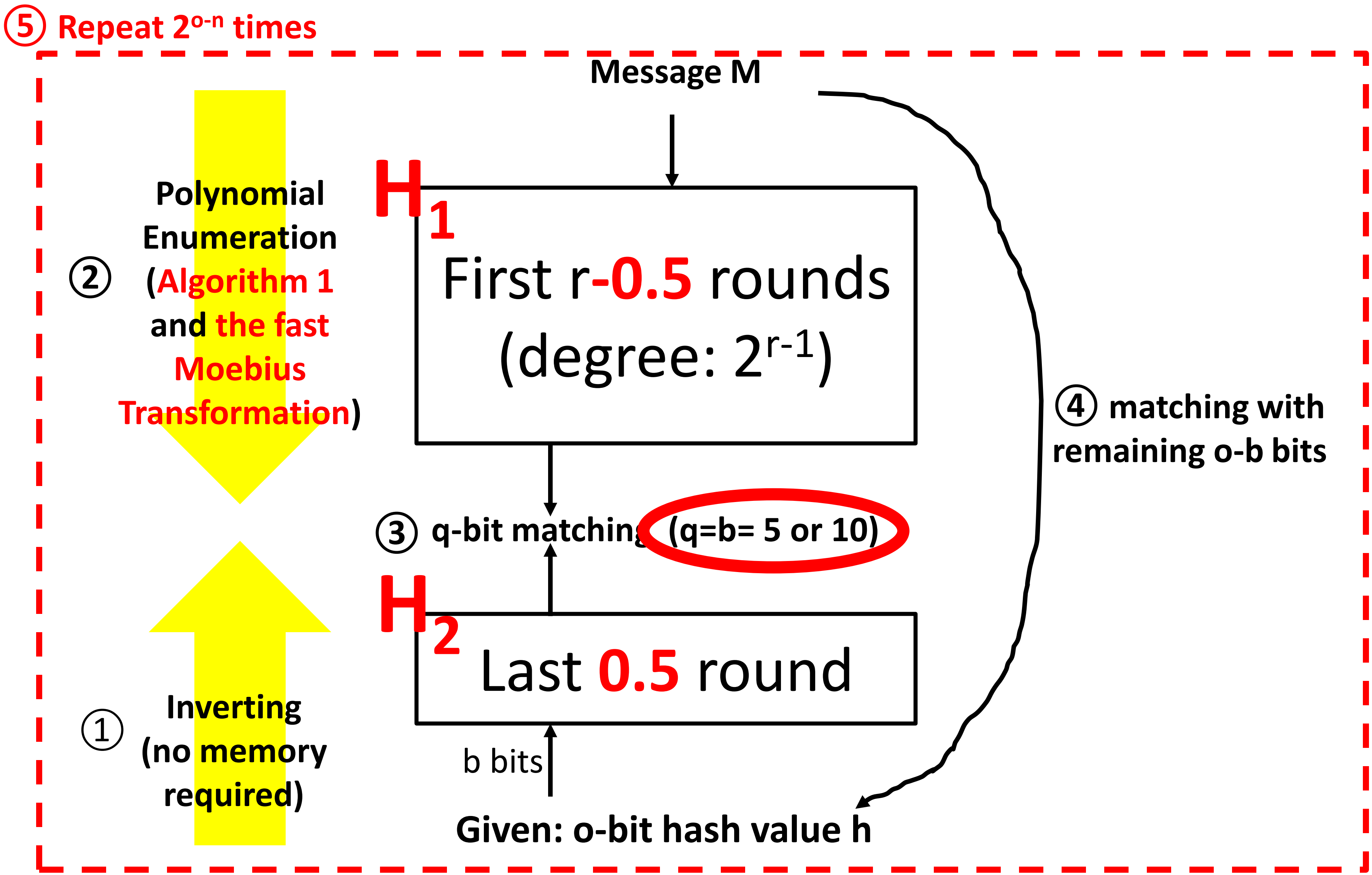
Memory Complexity:

$$\boxed{q \times 2^{q-b}} + \boxed{(2d+q+1) \times 2^n}.$$

Lookup Table for H_2

q Coefficient arrays and 2d+1 Sum arrays of size 2^n for Polynomial Enumeration

Application to Keccak (Result 3)



1^{st} and 2^{nd} Preimage Attacks on 6, 7, 8, 9 rounds of Keccak (Result 3)

Version	Reference	No. of Rounds	Type of attack	Time Complexity	Memory Complexity	Improvement Factor
Keccak-256	[18]	2	Preimage	2^{33}		2^{223}
Keccak-512	[17]	3	Preimage	2^{506}		64
Keccak-512	[illegible]	[illegible]	Preimage	2^{506}		64
Keccak-512	[12, 8]	6	2nd Preimage	2^{506}	2^{176}	50
	[12, 8, 14]	7	"	2^{507}	2^{320}	37
	[12, 8, 14]	8	"	$2^{511.4}$	2^{508}	1.44
	This work, § 7	6	Preimage/ 2nd Preimage	$2^{509.19}$	$2^{98.91}$	7.01
	This work, § 7	7	"	$2^{509.39}$	$2^{172.52}$	6.13
	[illegible]	8	"	$2^{509.73}$	$2^{315.29}$	4.81
Keccak-224	This work, § 8	7	"	$2^{218.11}$	$2^{180.12}$	58.66
Keccak-256	This work, § 8	8	"	$2^{255.64}$	$2^{254.03}$	1.29
Keccak-384	This work, § 8	8	"	$2^{378.74}$	$2^{324.06}$	38.36
Keccak-512	This work, § 8	6	"	$2^{505.58}$	$2^{104.23}$	85.70
	This work, § 8	7	"	$2^{506.11}$	$2^{180.12}$	59.34
	This work, § 8	8	"	$2^{506.74}$	$2^{324.07}$	38.36
	This work, § 8	9	"	$2^{511.70}$	$2^{510.02}$	1.23

Bernstein's results

Our results

1st and 2nd Preimage Attacks on 6, 7, 8, 9 rounds of Keccak (Result 3)

Bernstein's results

Our results

50 → 85.70

Version	Reference	No. of Rounds	Type of attack	Time Complexity	Memory Complexity	Improvement Factor
Keccak-256	[18]	2	Preimage	2^{33}		2^{223}
Keccak-512	[17]	3	Preimage	2^{506}		64
Keccak-512	[illegible]	[illegible]	Preimage	2^{506}		64
Keccak-512	[12, 8]	6	2nd Preimage	2^{506}	2^{176}	50
	[12, 8, 14]	7	"	2^{507}	2^{320}	37
	[12, 8, 14]	8	"	$2^{511.4}$	2^{508}	1.44
	This work, § 7	6	Preimage/ 2nd Preimage	$2^{509.19}$	$2^{98.91}$	7.01
	This work, § 7	7	"	$2^{509.39}$	$2^{172.52}$	6.13
	[illegible]	8	"	$2^{509.73}$	$2^{315.29}$	4.81
Keccak-224	This work, § 8	7	"	$2^{218.11}$	[illegible]	58.66
Keccak-256	This work, § 8	8	"	$2^{255.64}$	$2^{254.03}$	1.29
Keccak-384	This work, § 8	8	"	$2^{378.74}$	$2^{324.06}$	38.36
Keccak-512	This work, § 8	6	"	$2^{505.58}$	$2^{104.23}$	85.70
	This work, § 8	7	"	$2^{506.11}$	$2^{180.12}$	59.34
	This work, § 8	8	"	$2^{506.74}$	$2^{324.07}$	38.36
	This work, § 8	9	"	$2^{511.70}$	$2^{510.02}$	1.23

1^{st} and 2^{nd} Preimage Attacks on 6, 7, 8, 9 rounds of Keccak (Result 3)

Version	Reference	No. of Rounds	Type of attack	Time Complexity	Memory Complexity	Improvement Factor
Keccak-256	[18]	2	Preimage	2^{33}		2^{223}
Keccak-512	[17]	3	Preimage	2^{506}		64
Keccak-512	[illegible]	[illegible]	Preimage	2^{506}		64
Keccak-512	[12, 8]	6	2nd Preimage	2^{506}	2^{176}	50
	[12, 8, 14]	7	"	2^{507}	2^{320}	37
	[12, 8, 14]	8	"	$2^{511.4}$	2^{508}	1.44
	This work, § 7	6	Preimage/ 2nd Preimage	$2^{509.19}$	$2^{98.91}$	7.01
	This work, § 7	7	"	$2^{509.39}$	$2^{172.52}$	6.13
	[illegible]	8	"	$2^{509.73}$	$2^{315.29}$	4.81
Keccak-224	This work, § 8	7	"	$2^{218.11}$	$2^{180.12}$	58.66
Keccak-256	This work, § 8	8	"	$2^{255.64}$	[illegible]	1.29
Keccak-384	This work, § 8	8	"	$2^{378.74}$	[illegible]	38.36
Keccak-512	This work, § 8	6	"	$2^{505.58}$	$2^{104.23}$	85.70
	This work, § 8	7	"	$2^{506.11}$	$2^{180.12}$	59.34
	This work, § 8	8	"	$2^{506.74}$	$2^{324.07}$	38.36
	This work, § 8	9	"	$2^{511.70}$	$2^{510.02}$	1.23

Bernstein's results

Our results

37 → 59.34

1st and 2nd Preimage Attacks on 6, 7, 8, 9 rounds of Keccak (Result 3)

Version	Reference	No. of Rounds	Type of attack	Time Complexity	Memory Complexity	Improvement Factor
Keccak-256	[18]	2	Preimage	2^{33}		2^{223}
Keccak-512	[17]	3	Preimage	2^{506}		64
Keccak-512			Preimage	2^{506}		64
Keccak-512	[12, 8]	6	2nd Preimage	2^{506}	2^{176}	50
	[12, 8, 14]	7	"	2^{507}	2^{320}	37
	[12, 8, 14]	8	"	$2^{511.4}$	2^{508}	1.44
	This work, § 7	6	Preimage/ 2nd Preimage	$2^{509.19}$	$2^{98.91}$	7.01
	This work, § 7	7	"	$2^{509.39}$	$2^{172.52}$	6.13
		8	"	$2^{509.73}$	$2^{315.29}$	4.81
Keccak-224	This work, § 8	7	"	$2^{218.11}$	$2^{180.12}$	58.66
Keccak-256	This work, § 8	8	"	$2^{255.64}$	$2^{254.03}$	1.29
Keccak-384	This work, § 8	8	"	$2^{378.74}$		38.36
Keccak-512	This work, § 8	6	"	$2^{505.58}$	$2^{104.23}$	85.70
	This work, § 8	7	"	$2^{506.11}$	$2^{180.12}$	59.34
	This work, § 8	8	"	$2^{506.74}$	$2^{324.07}$	38.36
	This work, § 8	9	"	$2^{511.70}$	$2^{510.02}$	1.23

Bernstein's results

Our results

1.44 → 38.36

1st and 2nd Preimage Attacks on 6, 7, 8, 9 rounds of Keccak (Result 3)

Bernstein's results

Our results

New : 1.23

Version	Reference	No. of Rounds	Type of attack	Time Complexity	Memory Complexity	Improvement Factor
Keccak-256	[18]	2	Preimage	2^{33}		2^{223}
Keccak-512	[17]	3	Preimage	2^{506}		64
Keccak-512			Preimage	2^{506}		64
Keccak-512	[12, 8]	6	2nd Preimage	2^{506}	2^{176}	50
	[12, 8, 14]	7	"	2^{507}	2^{320}	37
	[12, 8, 14]	8	"	$2^{511.4}$	2^{508}	1.44
	This work, § 7	6	Preimage/ 2nd Preimage	$2^{509.19}$	$2^{98.91}$	7.01
	This work, § 7	7	"	$2^{509.39}$	$2^{172.52}$	6.13
		8	"	$2^{509.73}$	$2^{315.29}$	4.81
Keccak-224	This work, § 8	7	"	$2^{218.11}$	$2^{180.12}$	58.66
Keccak-256	This work, § 8	8	"	$2^{255.64}$	$2^{254.03}$	1.29
Keccak-384	This work, § 8	8	"	$2^{378.74}$	$2^{324.06}$	38.36
Keccak-512	This work, § 8	6	"	$2^{505.58}$	$2^{104.23}$	85.70
	This work, § 8	7	"	$2^{506.11}$		59.34
	This work, § 8	8	"	$2^{506.74}$	$2^{324.07}$	38.36
	This work, § 8	9	"	$2^{511.70}$	$2^{510.02}$	1.23

Work in Progress

- **Message Modification**: Good selection of position of message lanes will not double the degree by bypassing chi step (χ) of the round function of Keccak.
- Very careful memory and time complexity analysis required (at the complexities close to exhaustive search)
- Our preliminary analysis shows
  - 1^{st} and 2^{nd} preimage attacks on **9 rounds** of Keccak-256 with improvement factor 1.14
  - 1^{st} and 2^{nd} preimage attacks on **10 rounds** of Keccak-512 with improvement factor 1.05

Conclusion

- None of the attacks threatens the security of Keccak as the attack complexities are already close to brute force by the time we cross 9 rounds of Keccak.
- In fact, this work shows the limits of polynomial enumeration method-based preimage attacks against Keccak.
- Our Attack on reduced rounds of Keccak can be applied to reduced rounds of SHA3 with the same complexity and same number of rounds.