

On Some Variants of Cube-like Attacks

Mohammad Vaziri

End-of-ESCADA Workshop

26 August 2024

Table of Contents

- 1 Cube Attack
- 2 Keccak Sponge Function
- 3 Cube-Attack-Like Cryptanalysis
- 4 Improvement

Cube Attack(Simple Example and Terminology)

- Assume that f is:

$$f(x_1, x_2, x_3, k_1) = x_1 + x_1x_2x_3 + x_1x_2k_1 + x_3.$$

Cube Attack(Simple Example and Terminology)

- Assume that f is:

$$f(x_1, x_2, x_3, k_1) = x_1 + x_1x_2x_3 + x_1x_2k_1 + x_3.$$

- For recovering the k_1 , we choose x_1x_2 as cube, so the f can be represented as:

$$f(x_1, x_2, x_3, k_1) = x_1x_2(x_3 + k_1) + x_1 + x_3.$$

Cube Attack(Simple Example and Terminology)

- Assume that f is:

$$f(x_1, x_2, x_3, k_1) = x_1 + x_1x_2x_3 + x_1x_2k_1 + x_3.$$

- For recovering the k_1 , we choose x_1x_2 as cube, so the f can be represented as:

$$f(x_1, x_2, x_3, k_1) = x_1x_2(x_3 + k_1) + x_1 + x_3.$$

- By doing summation on all of the possible values of the cube variables x_1 and x_2 we have:

$$f(0, 0, x_3, k_1) + f(0, 1, x_3, k_1) + f(1, 0, x_3, k_1) + f(1, 1, x_3, k_1) = x_3 + k_1.$$

Cube Attack(Simple Example and Terminology)

- Assume that f is:

$$f(x_1, x_2, x_3, k_1) = x_1 + x_1x_2x_3 + x_1x_2k_1 + x_3.$$

- For recovering the k_1 , we choose x_1x_2 as cube, so the f can be represented as:

$$f(x_1, x_2, x_3, k_1) = x_1x_2(x_3 + k_1) + x_1 + x_3.$$

- By doing summation on all of the possible values of the cube variables x_1 and x_2 we have:

$$f(0, 0, x_3, k_1) + f(0, 1, x_3, k_1) + f(1, 0, x_3, k_1) + f(1, 1, x_3, k_1) = x_3 + k_1.$$

Cube Attack Terminology

$$f : X^n \rightarrow \{0, 1\} \longrightarrow f(x) = t.P_t(x) + Q(x), \quad t = x_0 \dots x_{k-1}$$

Then sum of f over all values of t is:

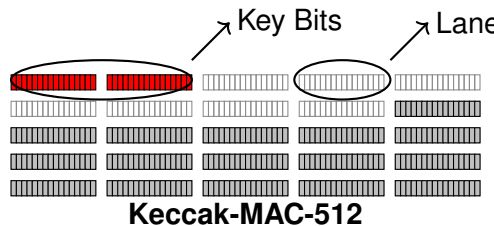
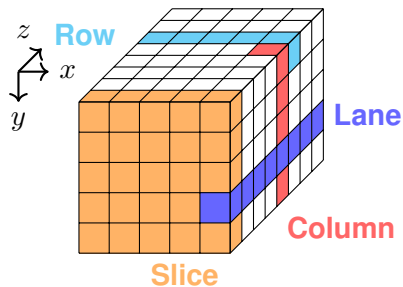
$$\sum_{x'=(x_0, \dots, x_{k-1}) \in C_t} f(x', x) = P_t(\underbrace{1, \dots, 1}_k, x_k, \dots, x_{n-1})$$

C_t contains all binary vectors of the length k and $P_t(x)$ is called **superpoly** of t .

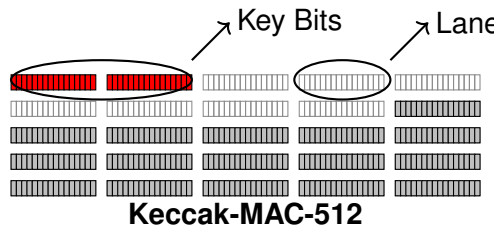
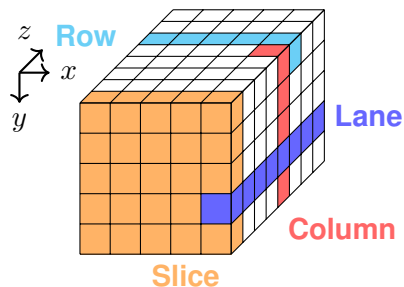
Cube Attack(Description)

- In Cube attack the attack is divided into two phases Preprocessing (Offline) Phase and Online Phase.
- The preprocessing phase is independent of the value of the secret key.
- In the preprocessing phase, $P_t(x)$ is built by interpolating the coefficients of $P_t(x)$, .
- The online phase is carried out after the secret key is set.
- For a given cube C_t , the expression $P_t(x)$ determined during the preprocessing phase equals a binary value that is evaluated in the online phase.

Keccak Sponge Function (Description)

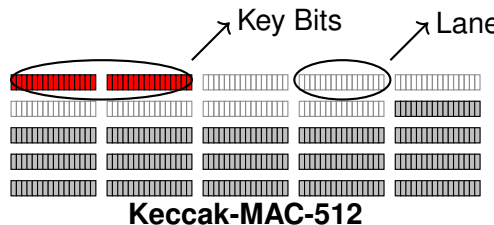
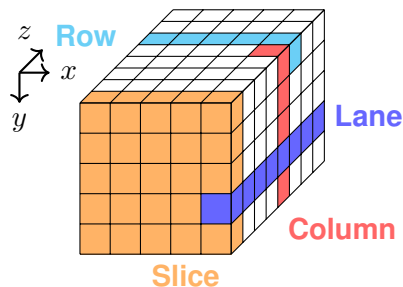


Keccak Sponge Function (Description)



$$\theta : A[x][y] = A[x][y] \oplus \sum_{j=0}^4 (A[x-1][j] \oplus (A[x+1][j] \lll 1)).$$

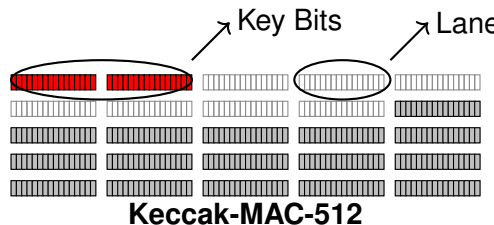
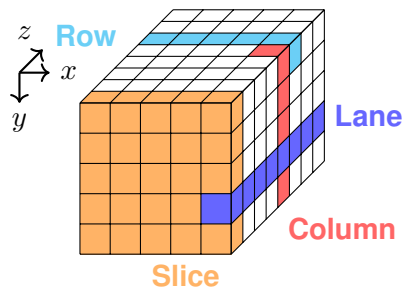
Keccak Sponge Function (Description)



$$\theta : A[x][y] = A[x][y] \oplus \sum_{j=0}^4 (A[x-1][j] \oplus (A[x+1][j] \lll 1)).$$

$$\rho : A[x][y] = A[x][y] \lll r[x, y].$$

Keccak Sponge Function (Description)

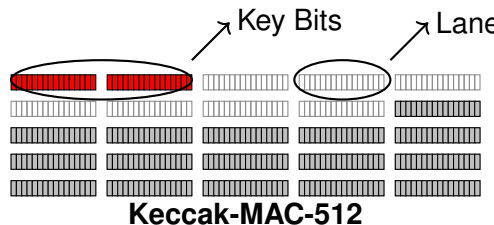
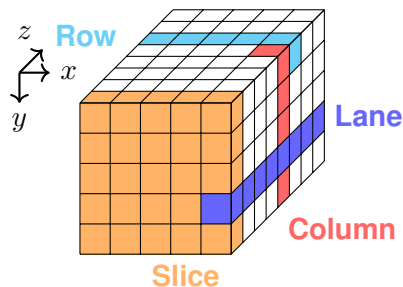


$$\theta : A[x][y] = A[x][y] \oplus \sum_{j=0}^4 (A[x-1][j] \oplus (A[x+1][j] \lll 1)).$$

$$\rho : A[x][y] = A[x][y] \lll r[x, y].$$

$$\Pi : A[y][2x + 3y] = A[x][y].$$

Keccak Sponge Function (Description)



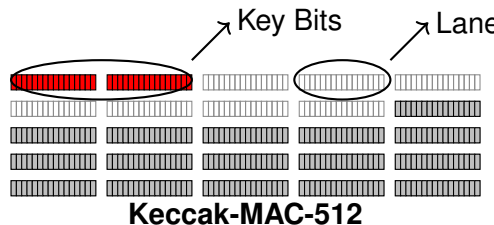
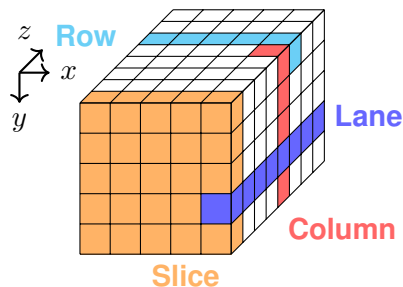
$$\theta : A[x][y] = A[x][y] \oplus \sum_{j=0}^4 (A[x-1][j] \oplus (A[x+1][j] \lll 1)).$$

$$\rho : A[x][y] = A[x][y] \lll r[x, y].$$

$$\Pi : A[y][2x + 3y] = A[x][y].$$

$$\chi : A[x][y] = A[x][y] \oplus ((\neg A[x+1][y]) \wedge A[x+2][y]).$$

Keccak Sponge Function (Description)



$$\theta : A[x][y] = A[x][y] \oplus \sum_{j=0}^4 (A[x-1][j] \oplus (A[x+1][j] \lll 1)).$$

$$\rho : A[x][y] = A[x][y] \lll r[x, y].$$

$$\Pi : A[y][2x + 3y] = A[x][y].$$

$$\chi : A[x][y] = A[x][y] \oplus ((\neg A[x+1][y]) \wedge A[x+2][y]).$$

Cube-Attack-Like Cryptanalysis

- The cube variables should be set in a way that prevents them from multiplying together in the first round of SHA3-Designs.

Cube-Attack-Like Cryptanalysis

- The cube variables should be set in a way that prevents them from multiplying together in the first round of SHA3-Designs.
- Since the cube variables do not multiply together in the first round, they instead multiply with some key bits (**related-key-bits**).

Cube-Attack-Like Cryptanalysis

- The cube variables should be set in a way that prevents them from multiplying together in the first round of SHA3-Designs.
- Since the cube variables do not multiply together in the first round, they instead multiply with some key bits (**related-key-bits**).
- By bypassing 1 round of multiplication, the degree of SHA3-like designs after r rounds is 2^{r-1} .

Cube-Attack-Like Cryptanalysis

- The cube variables should be set in a way that prevents them from multiplying together in the first round of SHA3-Designs.
- Since the cube variables do not multiply together in the first round, they instead multiply with some key bits (**related-key-bits**).
- By bypassing 1 round of multiplication, the degree of SHA3-like designs after r rounds is 2^{r-1} .
- Therefore, to attack r rounds, 2^{r-1} cube variables need to be chosen.

Cube-Attack-Like Cryptanalysis

- The cube variables should be set in a way that prevents them from multiplying together in the first round of SHA3-Designs.
- Since the cube variables do not multiply together in the first round, they instead multiply with some key bits (**related-key-bits**).
- By bypassing 1 round of multiplication, the degree of SHA3-like designs after r rounds is 2^{r-1} .
- Therefore, to attack r rounds, 2^{r-1} cube variables need to be chosen.
- The time and memory complexity of the attack, only depends on the **cube variables** and **related-key-bits**.

Cube-Attack-Like Cryptanalysis (Attack Process)

- Assume that the number of the required cube variables is s and the number of the related-key bits is t .

Cube-Attack-Like Cryptanalysis (Attack Process)

- Assume that the number of the required cube variables is s and the number of the related-key bits is t .

Preprocessing Phase

- For each of the 2^t possible values of related-key bits calculate the cube sum and store the amounts in a sorted list L
- The time complexity is 2^{s+t} and the memory complexity is 2^t .

Cube-Attack-Like Cryptanalysis (Attack Process)

- Assume that the number of the required cube variables is s and the number of the related-key bits is t .

Preprocessing Phase

- For each of the 2^t possible values of related-key bits calculate the cube sum and store the amounts in a sorted list L
- The time complexity is 2^{s+t} and the memory complexity is 2^t .

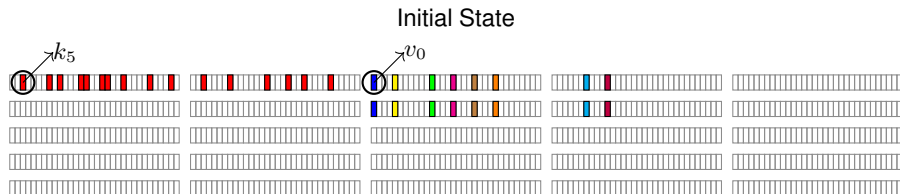
Online Phase

- Calculate the cube sums for the output bits and search them in L
- The time complexity is 2^s .

The total time and memory complexity for recovering t bits is
 $2^{s+t} + 2^s$ and 2^t .

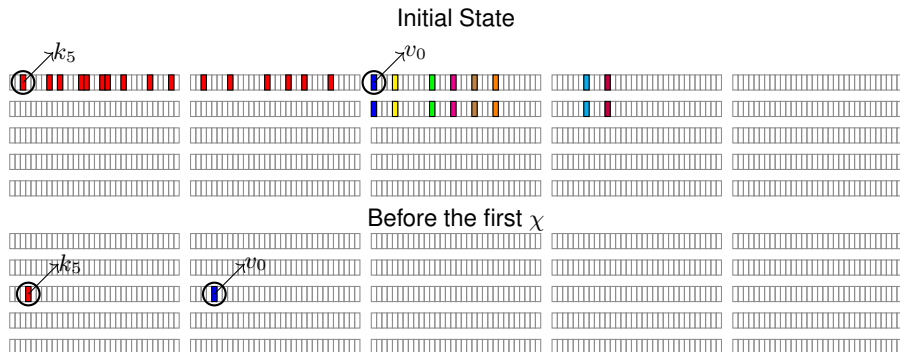
Cube-Attack-Like Cryptanalysis (Simple Example)

- Let's assume that we want to attack 4 rounds of Keccak-Mac-512.



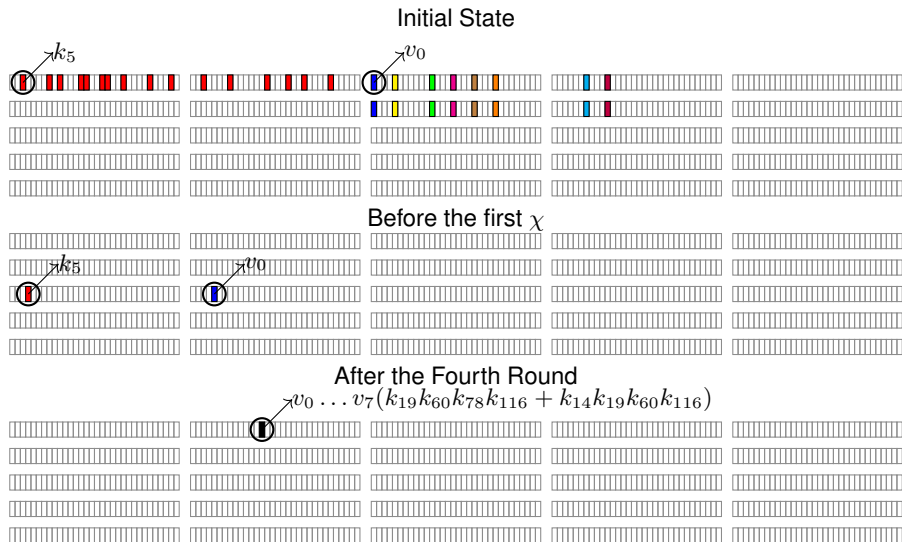
Cube-Attack-Like Cryptanalysis (Simple Example)

- Let's assume that we want to attack 4 rounds of Keccak-Mac-512.



Cube-Attack-Like Cryptanalysis (Simple Example)

- Let's assume that we want to attack 4 rounds of Keccak-Mac-512.



Cube-Attack-Like Cryptanalysis (Some Results)

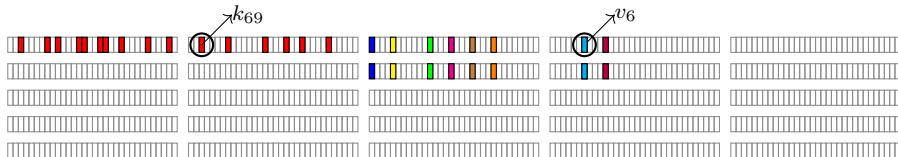
- Cube attack-like cryptanalysis aims to minimize the involvement of related key bits by carefully selecting cube variables.
- Works like [Song, 2019] and [Bi, 2019] proposed using MILP techniques to minimize the number of related key bits.
- In this work, we build on previous results to further reduce the number of involved related keys.

Target	Key	Rounds	Time	memory	Reference
Keccak-Mac-512	128	7/24	2^{111}	2^{111}	[Song 2019, 2018]
Keccak-Mac-512	128	6/24	2^{59}	2^{26}	[Bi, 2018]
Keccak-Mac-512	128	7/24	$2^{112.6}$	2^{47}	[Bi, 2018]

Improvement (First Approach)

- To attack 4 rounds of Keccak-MAC-512, consider the following two states.

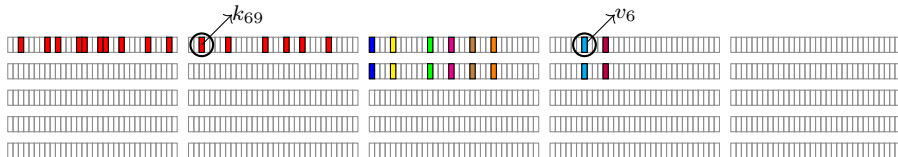
State 1



Improvement (First Approach)

- To attack 4 rounds of Keccak-MAC-512, consider the following two states.

State 1



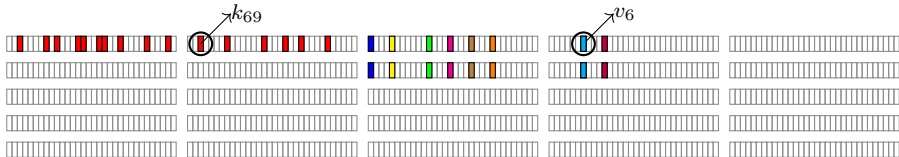
State 2



Improvement (First Approach)

- To attack 4 rounds of Keccak-MAC-512, consider the following two states.

State 1



State 2



The i -th bit of State 1 after 4 rounds: $v_0 \dots v_7 P_i(k_5, \dots, k_{116}) + Q_i$.

The i -th bit of State 2 after 4 rounds: $v_0 \dots v_7 P'_i(k_5, \dots, k_{116}) + Q'_i$.

$$\sum_{(v_0, \dots, v_7) \in C} f(State1) + \sum_{(v_0, \dots, v_7) \in C} f(State2) = P_i(k_5, \dots, k_{116}) + P'_i(k_5, \dots, k_{116})$$

f is 4 rounds of Keccak, C contains all binary vectors of the length 8

Improvement (First Approach)

- The differences of State 1 and State 2 after some operations.

Initial difference between State 1 and State 2



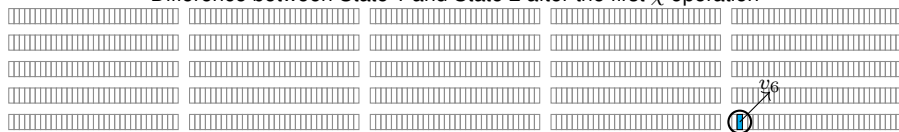
Improvement (First Approach)

- The differences of State 1 and State 2 after some operations.

Initial difference between State 1 and State 2



Difference between State 1 and State 2 after the first χ operation



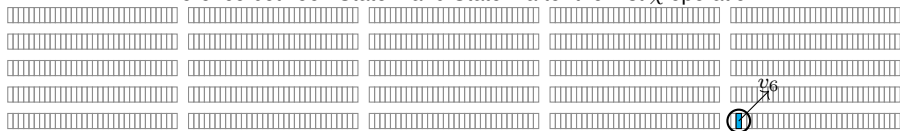
Improvement (First Approach)

- The differences of State 1 and State 2 after some operations.

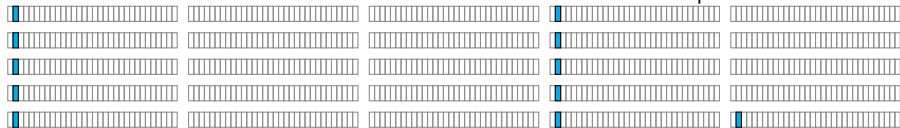
Initial difference between State 1 and State 2



Difference between State 1 and State 2 after the first χ operation

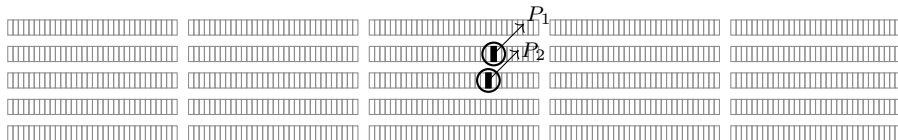


Difference between State 1 and State 2 after the second θ operation



Improvement (First Approach)

Difference between State 1 and State 2 after the fourth χ operation



$$P1 = v_0 \dots v_7(k_{19}k_{27}k_{28}k_{34}k_{52} + k_{19}k_{27}k_{28}k_{116}).$$

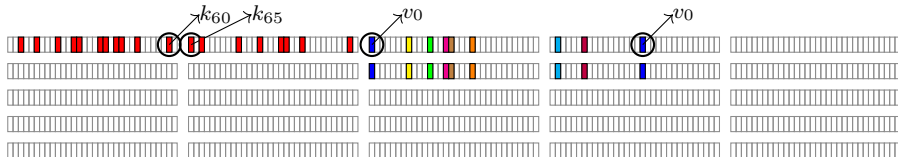
$$P2 = v_0 \dots v_7(k_5k_{19}k_{100} + k_{19}k_{36}k_{69} + k_{19}k_{69}k_{100} + k_5k_{19}k_{36}).$$

- The original attack depends on **16** related key bits, while the proposed approach depends on **9** related key bits.

Improvement (Second Approach)

- To attack 4 rounds of Keccak-MAC-512, consider the following two states.

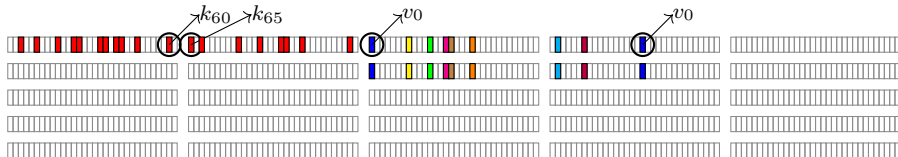
State 1



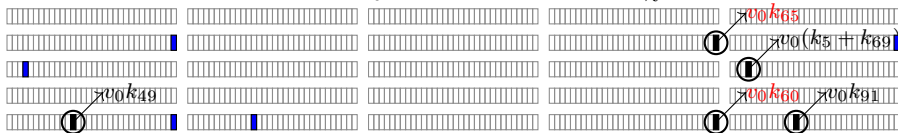
Improvement (Second Approach)

- To attack 4 rounds of Keccak-MAC-512, consider the following two states.

State 1



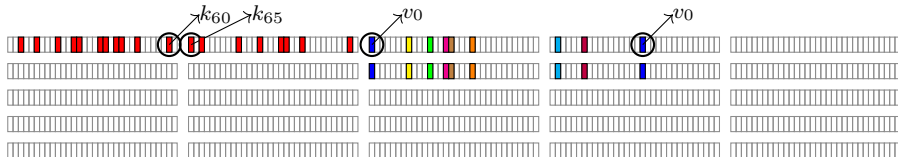
Diffusion of v_0 in the State 1 after the first χ



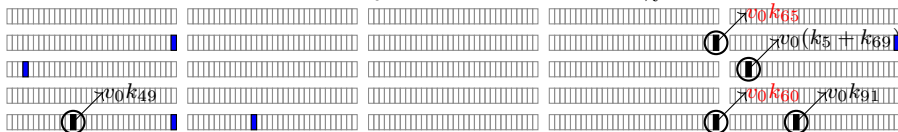
Improvement (Second Approach)

- To attack 4 rounds of Keccak-MAC-512, consider the following two states.

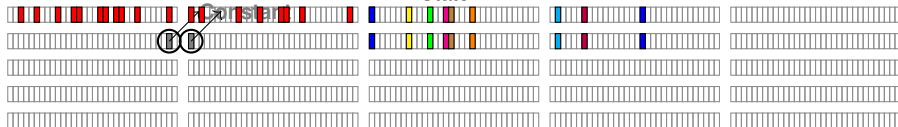
State 1



Diffusion of v_0 in the State 1 after the first χ



State 2



Improvement (Second Approach)

- The differences of State 1 and State 2 after some operations.

Initial difference between State 1 and State 2



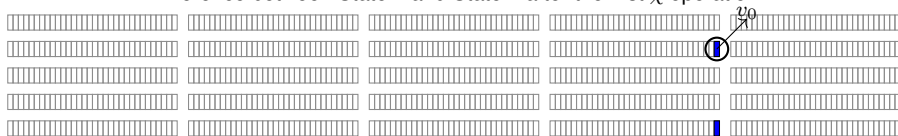
Improvement (Second Approach)

- The differences of State 1 and State 2 after some operations.

Initial difference between State 1 and State 2



Difference between State 1 and State 2 after the first χ operation



Improvement (Second Approach)

- The differences of State 1 and State 2 after some operations.

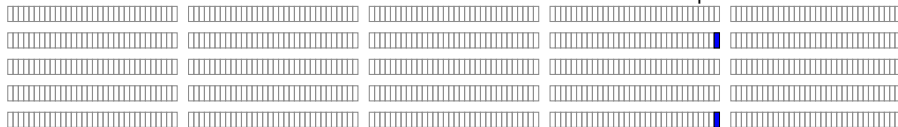
Initial difference between State 1 and State 2



Difference between State 1 and State 2 after the first χ operation

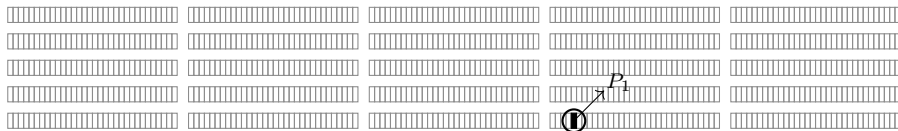


Difference between State 1 and State 2 after the second θ operation



Improvement (Second Approach)

Difference between State 1 and State 2 after the fourth χ operation



$$P_1 = v_0 \dots v_7 (k_{10}k_{25}k_{27}k_{34}k_{100}k_{107} + k_{10}k_{25}k_{34}k_{36}k_{91}k_{107} + k_{10}k_{25}k_{27}k_{91}k_{100}k_{108} + k_{10}k_{25}k_{27}k_{34}k_{36}k_{108}).$$

- The original attack depends on **16** related key bits, while the proposed approach depends on **8** related key bits.

Improvement(Interpolating the Superpolys)

- To identify which related keys are canceled, the generated superpolys in the bits must be interpolated.
- Since the generated superpolys in the bits are not linear, interpolating them is complicated.
- To address the challenge, a **tool** can be developed to symbolically track the diffusion of cube variables and secret key bits.
- To lighten the tool, only the monomials containing 2^{r-1} cube variables need to be considered in round r after χ .

Improvement(Interpolating the Superpolys)

- To clarify, consider the following example:
- Let's assume that the following 3 consecutive bits X_0 , X_1 , and X_2 are the input to the χ operation in round 3:

$$X_0 = k_3v_0v_3 + k_3v_0$$

$$X_1 = k_5k_{12}v_4v_5 + v_1v_3 + k_{14}$$

$$X_2 = v_0v_2 + k_{15}v_3$$

- To compute the corresponding output y_0 for the input bit x_0 , we have

$$Y_0 = X_0 + X_1X_2 + X_2$$

$$Y_0 = (k_3v_0v_3 + k_3v_0) + k_5k_{12}v_0v_2v_4v_5 + k_5k_{12}k_{15}v_3v_4v_5 + v_0v_1v_2v_3 + k_{15}v_1v_3 + k_{14}v_0v_2 + k_{14}k_{15}v_3$$

In Keccak function for the rounds greater than 1, the χ operation can be reformed as follows:

$$Y_0 = X_0 + X_1X_2 + X_2 \quad \equiv \quad Y_0 = X_0X_2$$

Thank you for your attention!
Questions?