ASK 2019, University of Hyogo, Kobe
December 13-15, 2019

Cube Attacks on KECCAK Keyed Modes

Willi Meier
Joint work with Zheng Li, Xiaoyang Dong, Wengquan Bi, Keting Jia and Xiaoyun Wang

n|w University of Applied Sciences Northwestern Switzerland
School of Engineering

Overview

- High-order differentials and cube attacks
- KECCAK Keyed Modes
- Conditional cube attack on KECCAK Keyed Modes
- Applications
- Conclusions

High-order differentials and cube attacks

Let V be a linear subspace of $\{0,1\}^n$ of dimension d.

Boolean function

$$f : \{0,1\}^n \mapsto \{0,1\}.$$

High-order differential: Derivative of order d of f with respect to V:

$$\Delta_V f(x) = \sum_{v \in V} f(x+v).$$

High-order differentials and cube attacks

Cube attack on $f(k, x)$, k secret, x public (Dinur-Shamir 2009):

For index set $I \subset \{1, .., n\}$ define t_I as monomial containing all public variables (cube variables), with index in I.

For fixed I, there is unique polynomial p such that ANF of $f(k, x)$ can be written as

$$f(k, x) = t_I p(k, x) + q(k, x),$$

where p does not contain any cube variable, and no monomial in q is divisible by monomial t_I.

p: superpoly of I in f.

Can compute superpoly by summing the outputs of f over all possible configurations of the cube variables.

Classical cube attack succeeds if superpoly is linear in key bits for a cube of size less than 50.

High-order differentials and cube attacks

Cube testers (2009). Test nonrandomness in high degree monomial distribution.

Dynamic cube attack on full round Grain-128 stream cipher by Dinur-Shamir.

Conditions in high-order differentials?

Impose conditions on ”basic differences” involved in summation, viewed as first-order difference.

Conditions assure that differential path is followed over many rounds.

May involve IV bits and key bits.

Many differences involved: Conditions for differences may contradict each other.

High-order differentials and cube attacks

Techniques first demonstrated for Trivium stream cipher.

Initialization (80-bit key, 80-bit IV):

$(s_1, s_2, ..., s_{93}) \leftarrow (k_0, ..., k_{79}, 0, 0, ..,)$
$(s_{94}, s_{95}, ..., s_{177}) \leftarrow (x_0, x_1, ..., x_{79}, 0., , , , 0)$
$(s_{178}, s_{179}, ..., s_{288}) \leftarrow (0, 0, ..., 0, 1, 1, 1)$
for $i = 1$ to $4 \cdot 288$ **do**

$t_1 \leftarrow s_{66} + s_{93}$
$t_2 \leftarrow s_{162} + s_{177}$
$t_3 \leftarrow s_{243} + s_{288}$
$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$
$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$
$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$
$(s_1, s_2, ..., s_{93}) \leftarrow (t_3, s_1, ..., s_{92})$
$(s_{94}, s_{95}, ..., s_{177}) \leftarrow (t_1, s_{94}, ..., s_{176})$
$(s_{178}, ..., s_{288}) \leftarrow (t_2, s_{178}, ..., s_{287})$

end for

High-order differentials and cube attacks

Output generation:

for $i = 1$ to ℓ **do**

$t_1 \leftarrow s_{66} + s_{93}$

$t_2 \leftarrow s_{162} + s_{177}$

$t_3 \leftarrow s_{243} + s_{288}$

$z_i \leftarrow t_1 + t_2 + t_3$

$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$

$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$

$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$

$(s_{178}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

end for

High-order differentials and cube attacks

High-order conditional differentials: Practical distinguisher for 961 out of 1152 rounds for subset of 2^{26} keys (KMN 2012).

Best cube distinguisher for Trivium initialization (for all keys) thus far (joint work with Kesarwani, Roy, S. Sarkar, 2019):

Cube of size 39. Reaches 850 rounds. Found by combination of degree evaluation method by Meicheng Liu (Crypto 2017) and GreedyBitSet algorithm.

Key recovery ongoing subject based on division property.

KECCAK Keyed Modes

KECCAK-p permutation denoted as KECCAK[b, n_r], where we assume $b = 1600$, and n_r denotes number of rounds.

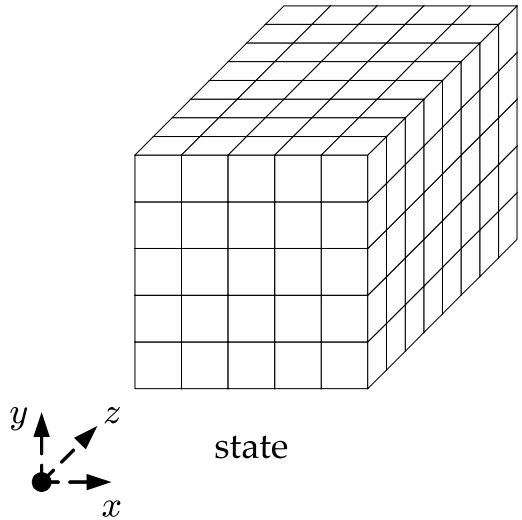
KECCAK[b, n_r] works on state A of b bits, represented as 5×5 lanes of 64 bits.

$A[x][y]$ denotes lane indexed by $A(x, y, *)$; indices x, y taken mod 5.

Round function built up as $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$.

$\theta : A[x][y] = A[x][y] \oplus \sum_{j=0}^{4} (A[x-1][j] \oplus (A[x+1][j] \lll 1))$.
$\rho : A[x][y] = A[x][y] \lll \rho[x, y]$.
$\pi : A[y][2x + 3y] = A[x][y]$.
$\chi : A[x][y] = A[x][y] \oplus ((\neg A[x+1][y]) \wedge A[x+2][y])$.
$\iota : A[0][0] = A[0][0] \oplus RC$.

KECCAK Keyed Modes



0, 0	1, 0	2, 0	3, 0	4, 0
0, 1	1, 1	2, 1	3, 1	4, 1
0, 2	1, 2	2, 2	3, 2	4, 2
0, 3	1, 3	2, 3	3, 3	4, 3
0, 4	1, 4	2, 4	3, 4	4, 4

Figure: (a) The KECCAK state, (b) A slice of state A

KECCAK Keyed Modes

KECCAK is based on the sponge construction.

Internal permutation is KECCAK-$p[1600, 24]$.

Sponge construction has parameters capacity c and rate r, with $c + r = 1600$. KECCAK$[c]$ denotes KECCAK with capacity c.

The state of KECCAK is initialized to 0. The input is a variable-length message M. Output is a 128-bit digest.

Message M is split into r-bit blocks. KECCAK processes blocks iteratively by absorbing them into the first r bits of the state and the permutation of KECCAK-$p[1600, 24]$.

In KECCAK-MAC, input is concatenation of key and message. Key size: 128 bits.

KECCAK-MAC-512 is a MAC based on KECCAK[1024].

KECCAK Keyed Modes

KMAC Message Authentication Code is a keyed hash function, whose output length is variable.

Two variants: KMAC128 and KMAC256 with capacities $c = 256$ and $c = 512$ bits.

Based on KECCAK[$c = 256$] and KECCAK[$c = 512$].

KMAC takes as parameters a key K, a variable length message M, output length L, name string N = "KMAC" and the optional customization bit string S of any length (including 0).

KECCAK Keyed Modes

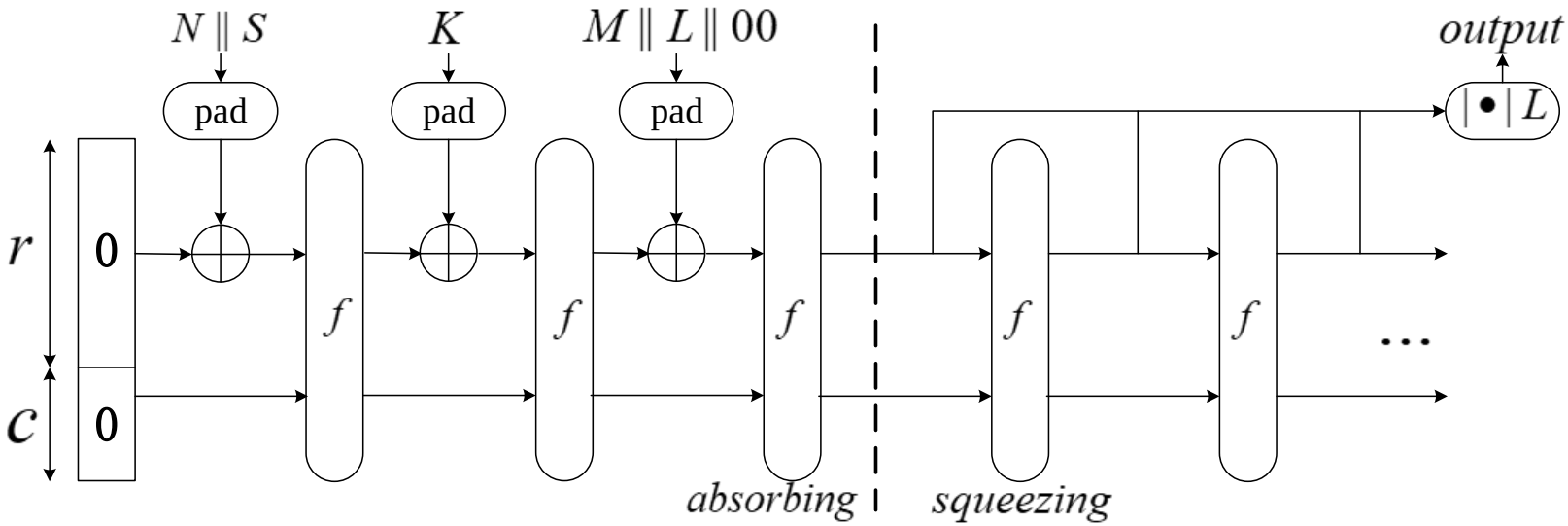


Figure: Construction of KMAC

Illustrates processing of one message block.

KECCAK Keyed Modes

Treat state before padding of message as initial state in attack.

Whole secret 1600-bit state treated as an equivalent key to be recovered.

Output length of `KMAC` is variable, while that of `KMAC128` is no less than 256 bits and `KMAC256` is no less than 512 bits.

Conditional cube attack on KECCAK Keyed Modes

In: New Conditional Cube Attack on Keccak Keyed Modes, ToSC 2019.

Quite some history on cryptanalysis of KECCAK keyed modes.

Conditional cube attack for KECCAK keyed modes introduced by Huang et. al. (Eurocrypt 2017).

Aims at key or state recovery (not just for distinguishing purpose).

Suitable conditions for IV-bits and key bits in order to reduce total degree (of reduced-round version).

Conditional cube attack on KECCAK Keyed Modes

Select a conditional cube variable. Bit conditions are added to reduce the diffusion of the conditional cube variable significantly.

A set of cube variables found that are not multiplied in the first round, while the conditional cube variable is not multiplied with other cube variables (called ordinary cube variables) in the first two rounds.

Key bit conditions lead to a key-recovery attack.

Different from Huang et al.'s work, remove the limitation that all the cube variables do not multiply in the 1st round.

Only two cube variables v_0, v_1 multiply in the 1st round, i.e. $v_0 v_1$ is the unique quadratic term in the output of the 1st round.

Conditional cube attack on KECCAK Keyed Modes

Under certain bit conditions, the unique quadratic term $v_0 v_1$ does not multiply with any other cube variable in the 2nd round.

In the output of the 2nd round, the bit conditions make any cubic term containing $v_0 v_1$ disappear.

For $(n+2)$-round attacks, the cube sum over $(2^{n+1}+1)$ dimensions is zero when the bit conditions are satisfied.

Conditional cube attack on KECCAK Keyed Modes

Definition

Suppose the $(q+2)$ cube variables are $v_0, v_1, u_0, ...u_{q-1}$, and the constraints are as follows:

- In the output of the first round, $v_0 v_1$ is the only quadratic term;
- In the second round, if the bit conditions are satisfied, $v_0 v_1$ does not multiply with any of $u_0, ...u_{q-1}$, i.e. no cubic term occurs.
- In the second round, if the bit conditions are not satisfied, $v_0 v_1$ multiplies with some of $u_0, ...u_{q-1}$, i.e. some cubic terms like $v_0 v_1 u_i$ $(i = 0, ..., q-1)$ occur.

Then $v_0 v_1$ is called **kernel quadratic term**. The remaining cube variables except v_0 and v_1, i.e. $u_0, ...u_{q-1}$, are called **ordinary cube variables**.

Conditional cube attack on KECCAK Keyed Modes

Describe the new conditional cube attack using the **kernel quadratic term** in the following corollary.

Corollary

For the $(n+2)$*-round* KECCAK *sponge function* $(n > 0)$*, if there is one kernel quadratic term* $v_0 v_1$*, and* $q = 2^{n+1} - 1$ *ordinary cube variables,* $u_0, u_1, ..., u_{q-1}$*, the term* $v_0 v_1 u_0 u_1 ... u_{q-1}$ *will not appear in the output polynomials of the* $(n+2)$*-round* KECCAK *sponge function under certain bit conditions.*

The distinguisher works as follows:

- under the right bit conditions, the degree of the output of $(n+2)$-round KECCAK is no more than 2^{n+1};
- under wrong bit conditions, the degree of the output of $(n+2)$-round KECCAK is $q + 2 = 2^{n+1} + 1$.

Conditional cube attack on KECCAK Keyed Modes

CP-kernel: if all columns in a state have even parity, θ is the identity.

Does also hold for individual columns, if sum of two other suitable columns is 0.

In the state $S_{1,\pi}$ of the 2nd round, if the conditional cube variables occupy fewer bit positions, better chance to find the ordinary cube variables , which have to be not multiplied with the conditional cube variables after 2 rounds.

$S_{i-1,\pi}$: internal state after π in the i-th round, $i \geq 1$.

Conditional cube attack on KECCAK Keyed Modes

In Huang et al.'s work, as the left part in figure shows, one conditional cube variable v_0 is placed in two black bits of S_0, i.e. $S_0[2][0][0] = S_0[2][1][0] = v_0$, which is exactly set in the CP-kernel.

After adding some conditions, the conditional cube variable v_0 maintains only 2 active bits in S_1 and diffuses to 22 bits after the linear part θ, ρ, π of the 2nd round as shown in state $S_{1,\pi}$.

The diffusion pattern is denoted as 2-2-22. This pattern provides a nice restriction on the diffusion of the conditional cube variable v_0. Actually, the 2-2-22 patten is chosen by all the previous conditional cube attacks for KECCAK keyed modes.

Conditional cube attack on KECCAK Keyed Modes

New conditional cube attack: Only care that the kernel quadratic term (i.e. $v_0 v_1$) should not be multiplied with ordinary cube variables in the second round.

Therefore, in order to get more degrees of freedom to find ordinary cube variables, have to reduce the diffusion of $v_0 v_1$.

Hence, introduce a so-called 6-2-2 pattern shown in the right part.

In the initial state S_0, v_0 occupies 4 black bits of S_0, i.e. $S_0[2][0][0] = S_0[2][1][0] = S_0[3][0][34] = S_0[3][1][34] = v_0$, which are also set in the CP-kernel, and v_1 occupies the other 2 grey bits, i.e. $S_0[0][1][60] = S_0[1][1][1] = v_1$.

Conditional cube attack on KECCAK Keyed Modes

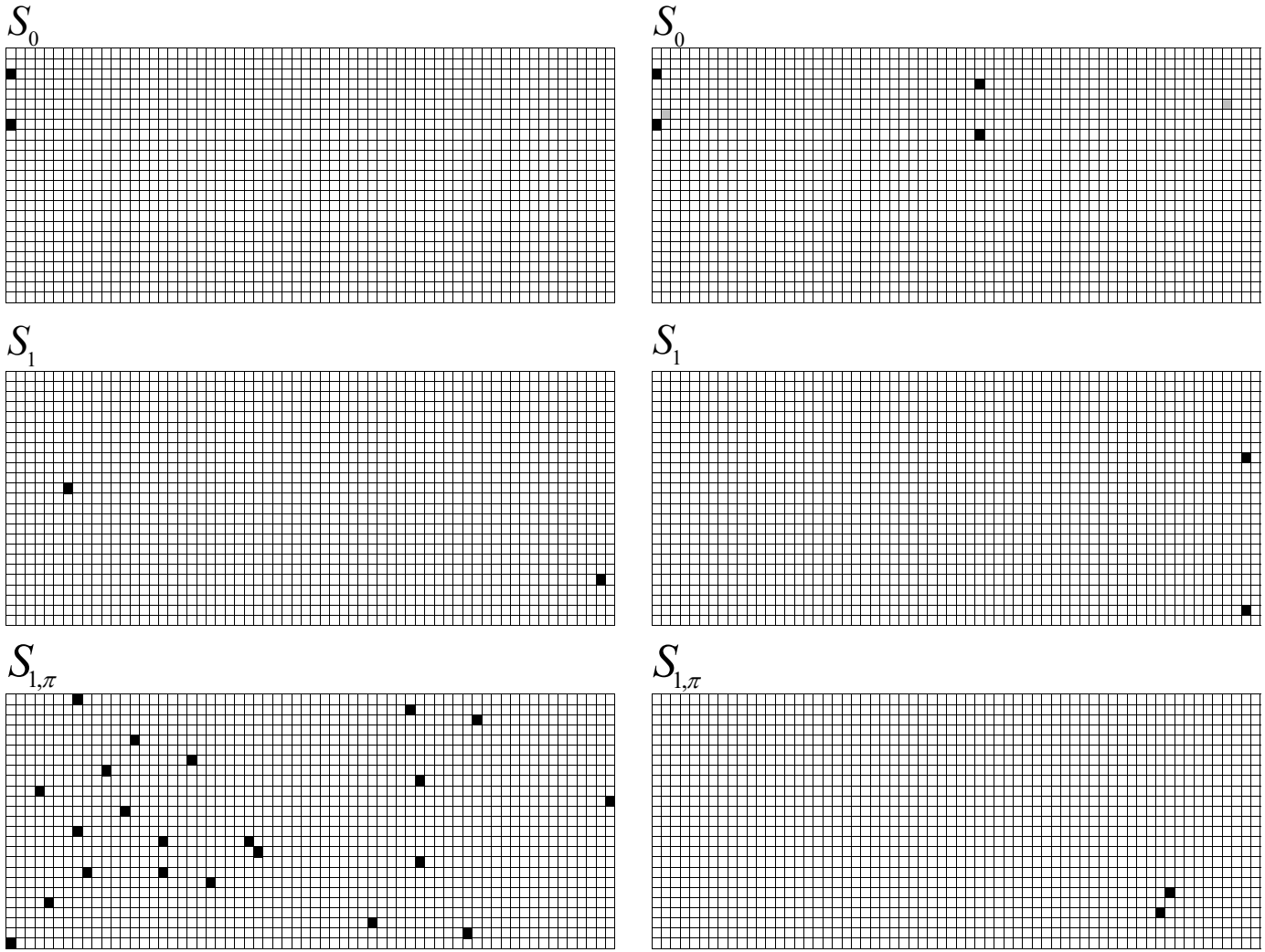


Figure: Diffusions of the conditional cube variable in 2-2-22 and 6-2-2 pattern in KECCAK-MAC-512

Conditional cube attack on KECCAK Keyed Modes

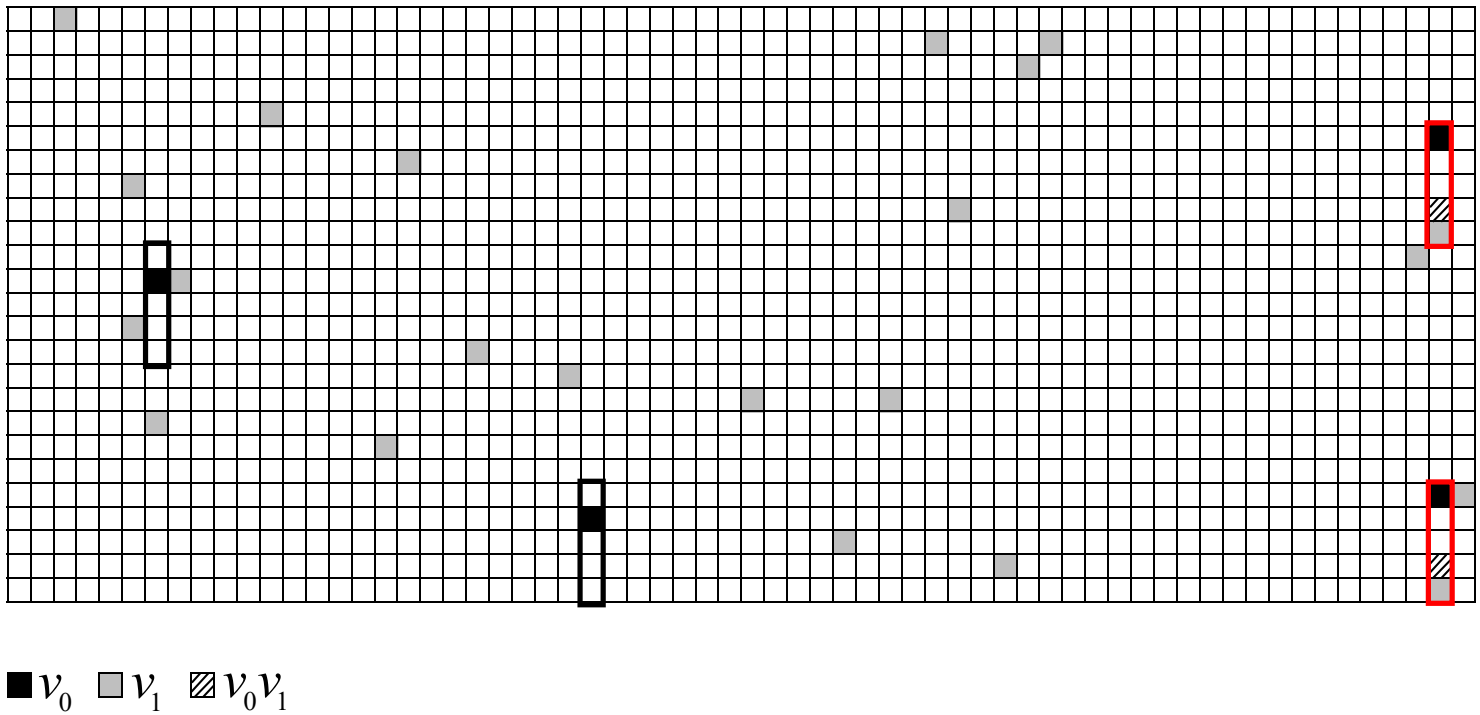


Figure: 6-2-2 pattern: generation of kernel quadratic terms in the first χ

Conditional cube attack on KECCAK Keyed Modes

In the χ operation of the 1st round, the kernel quadratic term $v_0 v_1$ is generated.

Before the χ operation, v_0 (initialized in the CP-kernel) only appears in the 4 black bits, while v_1 appears in 22 grey bits. Only 2 S-boxes highlighted by red rectangles are related to v_0 and v_1 simultaneously.

According to the expression of the χ operation: $A[x][y] = A[x][y] \oplus ((\neg A[x+1][y]) \wedge A[x+2][y]$, the quadratic term $v_0 v_1$ is just generated in two bits in black slashes after the χ operation.

Conditional cube attack on KECCAK Keyed Modes

Method to find 6-2-2 patterns:

As shown in Figure, $v_0 v_1$ is in the CP-kernel in S_1. Denote the bit positions of $v_0 v_1$ as (x, y_0, z), (x, y_1, z).

According to the expression of χ, $v_0 v_1$ in $S_1[x][y_0][z]$ is generated by multiplying v_0 in $S_{0,\pi}[x+1][y_0][z]$ and v_1 in $S_{0,\pi}[x+2][y_0][z]$, or v_0 in $S_{0,\pi}[x+2][y_0][z]$ and v_1 in $S_{0,\pi}[x+1][y_0][z]$.

The same happens to $v_0 v_1$ in $S_1[x][y_1][z]$. There are 4 cases to determine the bit positions for v_0 and v_1 in reverse.

Conditional cube attack on KECCAK Keyed Modes

MILP Model of the New Conditional Cube Attack:

The 6-2-2 pattern has already determined the bit positions of v_0 and v_1 in the initial state, as well as the bit positions of the kernel quadratic term $v_0 v_1$ in the output of the first round.

To perform an attack on a $(n+2)$-round KECCAK sponge function ($n > 0$), the remaining $2^{n+1} - 1$ ordinary cube variables and related conditions are found by a MILP model.

Applications

Attack on 7-round KECCAK-MAC-512

For KECCAK-MAC-512, a 7-round attack can be performed with 65 cube variables.

In the whole 1600-bit state, the rate occupies 576 bits, and the capacity 1024 bits.

Suppose $v_0 v_1$ is the kernel quadratic term. As Figure shows, the 128-bit key K is located at the first 2 red lanes, and v_0 is set in the CP-kernel as $S_0[2][0][0] = S_0[2][1][0] = S_0[3][0][34] = S_0[3][1][34] = v_0$ in 4 black bits, while v_1 is located at 2 grey bits, i.e. $S_0[0][1][60] = S_0[1][1][1] = v_1$. The white part represents free space for ordinary cube variables, i.e. like nonce bits which can be selected as ordinary cube variables, while the padding is in blue.

Applications

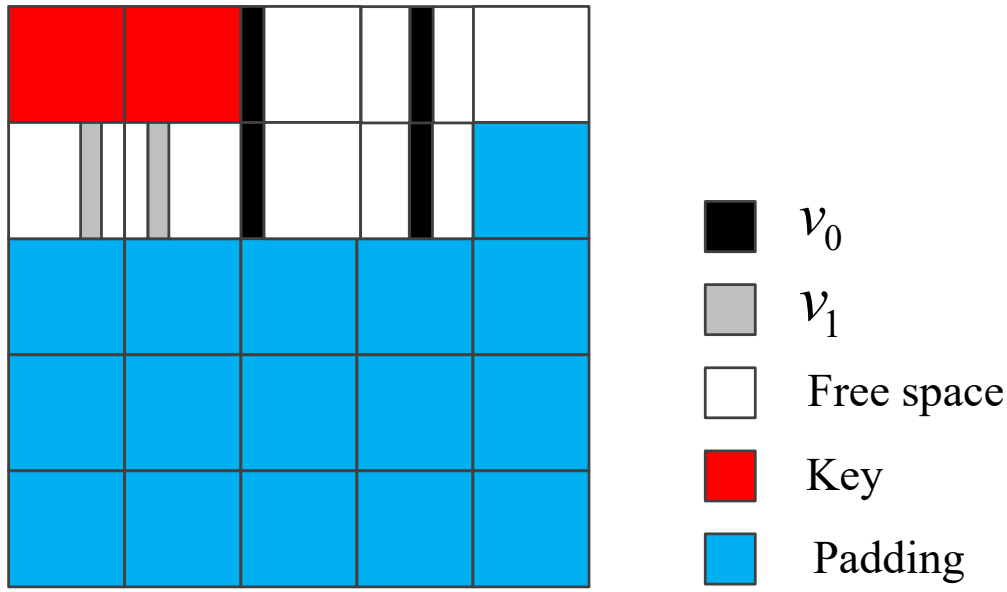


Figure: The initial state of KECCAK-MAC-512

According to new conditional cube attack, need to search with MILP program for the minimal number of key bit conditions and ordinary cube variables satisfying the corresponding rules.

Applications

In the 7-round attack on KECCAK-MAC-512, $2^6 + 1 = 65$ cube variables are optimal, denoted by $v_0, v_1, u_0, u_1 ... u_{62}$, where v_0 and v_1 are elements of the kernel quadratic term fixed in the beginning and $u_0, u_1 ... u_{62}$ are ordinary cube variables found by the MILP search strategy.

Summarize the requirements as follows:

(1) Only v_0 and v_1 multiply with each other in the first round;

(2) Under some conditions on key and nonce, $v_0 v_1$ does not multiply with any of $u_0, u_1 ... u_{62}$ in the second round;

(3) Under the conditions of the nonce in (2), if the key conditions are not satisfied, $v_0 v_1$ multiplies with some u_i ($i = 0, 1, ..., 62$) in the second round.

Applications

According to requirements, the kernel quadratic term $v_0 v_1$ is the unique quadratic term in the output of the first round (i.e. S_1).

So in the output of the second round, the available degree is at most 3 and the corresponding term should contain $v_0 v_1$ as a factor.

As in the above requirements (2) and (3), under the conditions on the nonce, all cubic terms disappear with the key conditions satisfied, while some cubic terms like $v_0 v_1 u_i$ $(i = 0, 1, ..., 62)$ appear without the key conditions.

Applications

The algebraic degree of the round function in KECCAK-p permutation is 2.

Thus the highest degree of terms in the output of the 7-round KECCAK-MAC-512 (i.e. S_7) is 2^6 with the bit conditions satisfied, while the highest degree of S_7 is $2^6 + 1 = 65$ in the case that the key bit conditions are not satisfied.

Therefore, if the 65-dimensional cube sums of 7-round output bits are 0, we conjecture that the key guess is correct with very high probability; if the term $v_0 v_1 u_0 \ldots u_{62}$ appears, the key guess is wrong.

Applications

Time and data complexity of the 7-round attack on KECCAK-MAC-512:

With suitable parameter sets, one guessed key bit can be recovered. The time complexity of one recovery is $2^1 \times 2^{65}$.

Due to symmetry of permutation in direction of the z-axis, can obtain corresponding parameter sets with any rotation index i ($0 \leq i < 64$) in the z-axis.

Therefore, the guessed key bits rotated by i bits can be recovered.

Applications

Through simple count, for $0 \leq i < 64$, 64 independent key bits (i.e. the whole first lane) out of 128 key bits can be recovered, 64 iterations consume $64 \times 2^1 \times 2^{65}$ and the remaining 64 key bits are left to exhaustive search consuming 2^{64}.

Combining the two parts, the procedure consumes $64 \times 2^1 \times 2^{65} + 2^{64} = 2^{72}$ computations of 7-round KECCAK-MAC-512, and each one corresponds to a unique value of the input.

After the procedure above, all the 128 bits in K can be recovered. Therefore, both time and data complexity of the attack are 2^{72}.

Conclusions

With methods as described, and related methods, the following reduced-round keyed primitives are cryptanalized:

- 7-round KECCAK-MAC-512 in 2^{72} time complexity (best previous result: 2^{111})
- 9-round KMAC256 in 2^{139} complexity (best previous result: 2^{147})
- 6-round Xoodyak authenticated encryption (NIST LWC 2nd round candidate) in nonce misuse scenario in practical time
- Cryptanalysis of a round 2 candidate of NIST lwc project: Cube-Based Cryptanalysis of Subterranean-SAE (Fukang Liu and Takanori Isobe)